UCLA Samueli Computer Science

CS M146 Discussion: Week 1

Course logistics, Math Review Exercise

Junheng Hao

Friday, 1/8/2021

Roadmap

- Course Logistics
- Math Prep: Calculus, linear algebra, probability and optimization
- Q & A

Course Information & Resources

- **Course schedule and logistics**
  - Syllabus (tentative) on CCLE
  - Logistics discussed in the first lecture (Week 1, Monday)
- **CCLE**
  - Slides, lecture recordings, and other private course materials
- **Online forum (Campuswire)** → ***Invitation sent!***
  - Link: https://campuswire.com/p/GB5E561C3 Passcode: 3428
  - Slides, QA and chat rooms.
  - If you have any questions, you may DM me on Campuswire or write me emails.
- **GradeScope** → ***Invitation sent!***
  - Submissions of problem set (total 4) and quizzes
  - Final exam

Course logistics

- Office hours & Zoom links (time in PST)
  - Sriram Sankararaman (sriram@cs.ucla.edu) Wednesday 3:00-4:00pm @Zoom
  - **Junheng Hao (haojh.ucla@gmail.com) Mondays 3:00-5:00 pm @Zoom**
  - Danfeng Guo (lyleguo@ucla.edu) Tuesdays 4:00pm-6:00pm @Zoom
  - Andrei Storozhenko (storozhenko@cs.ucla.edu) Tuesdays & Thursdays 11:00-12:00 am @Zoom
- Discussion 1C by Junheng Hao:
  - Time: 12-1:50 pm, Fridays.
  - Slides are posted on: https://www.haojunheng.com/teaching/cs146-winter21/
  - Recordings are posted on CCLE.
- Junheng's Zoom Link:
  - https://ucla.zoom.us/j/96240702917?pwd=QzFyWDZlYWpjNy9BSHl5OFMyNU1jdz09

Note: You can attend any discussion session (honestly they are at the same time).

Course Grading

- **Problem Sets: 50%**
  - Total 4 problem sets: Math/conceptual questions + programming tasks
  - No late submissions
- **Weekly Quizzes: 30%**
  - Math quiz on Week 1, weekly quizzes on Week 2-9
  - Lowest quiz score dropped
  - One-hour time for completion once the quiz starts
- **Final Exam: 20%**
  - Scheduled on **March 15**
  - All material covered, open book
  - Email to inform and confirm with both Prof. Sankararaman and your TA, for any accommodation approved by CAE

*Reminder: Daylight saving time 2021 in California will begin at 2:00 AM on **Sunday, March 14**!*

- **Default grade cutoff**

> 96	93	90	86	83	80	76	73	70	< 70
A +	A	A-	B+	B	B-	C+	C	C-	D

Yes, all online!

- As required by UCLA chancellor office, CS M146 is entirely online this quarter.
- All teaching activities (lectures, discussion sessions, office hours) will be held virtually through Zoom.
- Please **DO NOT** share the zoom links outside!
- Please **DO NOT** enter the meeting room outside the regular lectures, office hours and discussion time!

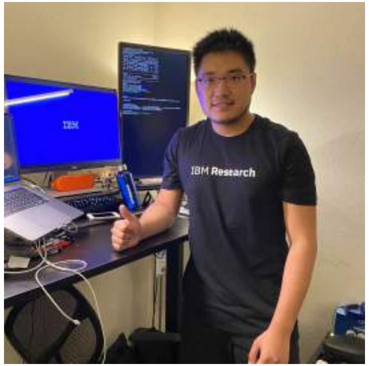
Other Questions?

- Enrollment problems on myUCLA
- PTE
- Grading option
- CS145 and/or CS146

Need some help? Check here!
https://www.studentincrisis.ucla.edu/Portals/36/Documents/redfolder.pdf

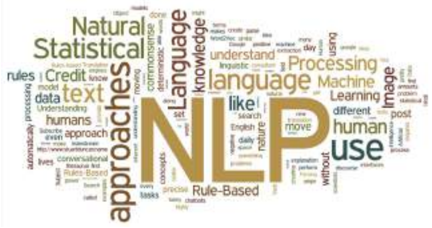
About TA (Myself)

- Fourth-year CS Ph.D. candidate
- UCLA Advisors: Yizhou Sun, Wei Wang (UCLA ScAi Institute, UCLA Data Mining Group)
- Past work experiences: @NEC Labs, @Amazon, @IBM Research AI
- Research interests: Knowledge Graphs, Graph mining, NLP, Bioinformatics, etc.
- Hobbies: Languages (beginner for Spanish and German), tennis, ...
- More about myself: https://www.haojunheng.com/



About My Research

- Foundational to knowledge-driven AI systems
- Enable many downstream applications (NLP tasks, Recommender, Bioinformatics, etc)

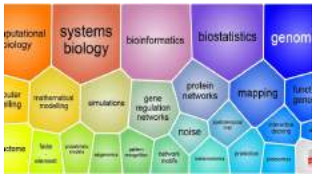


Natural Language Processing

QA & Dialogue systems

Knowledge Graphs

Recommender Systems



Computational Biology

Announcements

- 5:00pm, Jan. 8 ~~12:00am, Jan. 9~~: Math quiz released on Gradescope.
- **11:59pm, Jan. 10 (Sunday):** Math quiz closed on Gradescope!
- Jan. 15 (expected): Problem set 1 released on campuswire/CCLE, submission on Gradescope.

Other deadline reminders (Problem sets, Quizzes, etc) will be announced in class and Campuswire, as well as my discussion webpage.

"Machine learning is part of both ***statistics*** *and* ***computer science****."*

-- I don't know who said that.

Math Review: Probability

- **Checklist:**
  - Properties of probability
  - Probability spaces (discrete/continuous)
  - Probability distributions (discrete/continuous)
  - Random variables
  - Multivariate probability distributions
  - Marginal probability and conditional probability
  - Expectation, variance, covariance
  - Rules of probability
  - Independence and Bayes rule

Gaussian Distrib

Math Review: ~~Linear Algebra~~ Calculus

- Checklist:
  - Basic calculus $y = x^2, \frac{\delta y}{\delta x} = ?$
  - Gradient calculation in matrix format

$x \in R^{1 \times n}$

$f(\vec{x}) = \beta^T \vec{x}$

$\frac{\partial f(x)}{\partial \vec{x}} = \beta$

Math Review: Linear Algebra

- **Checklist:**
  - Vector, matrix
  - Norm
  - Multiplication
  - Useful (special) matrices
  - Rank of a matrix
  - Matrix inverse
  - Eigenvalues and eigenvectors

Math Review: Optimization *

- Checklist:
  - Convex set and convex functions
  - Gradients
  - Gradient descent
  - *(For SVM)* Quadratic problem and dual problem, duality, KKT condition. *

Math Review: Reading List

From the website: http://web.cs.ucla.edu/~sriram/courses/cm146.winter-2019/html/index.html (Details in the links below)

- Review of probability
  - Link 1: http://cs229.stanford.edu/section/cs229-prob.pdf
  - Link 2: https://www.cs.princeton.edu/courses/archive/spring07/cos424/scribe_notes/0208.pdf
- Linear Algebra
  - Link 2: http://cs229.stanford.edu/section/cs229-linalg.pdf
- Optimization
  - Link 1: http://cs229.stanford.edu/section/cs229-cvxopt.pdf
  - Link 2: http://cs229.stanford.edu/section/cs229-cvxopt2.pdf
- Machine Learning Math Essentials by Jeff Howbert from Washington U.
  - Link: http://courses.washington.edu/css490/2012.Winter/lecture_slides/02_math_essentials.pdf

About Math Quiz (Week 1)

- Notification: Campuswire Post#12.
- You will have up to **60 minutes** to take this exam.
- You can find the exam entry named "Week 1 Math Quiz" on GradeScope.
- There are in total **10 questions** with types of true/false and multiple choices. Note that for multiple-choice questions, it is possible to have one single correct answer and multiple correct answers (select all that apply).
- Quiz release date and time: **Jan 08, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Jan 10, 2021 (Sunday) 11:59 PM PST**

Practice Exercise 1

- Bayes Theorem
  - A patient goes to see a doctor. The doctor performs a test with 99 percent reliability, that is, 99 percent of people who are sick test positive and 99 percent of the healthy people test negative.
  - The doctor knows that only 1 percent of the people in the country are sick (well, obviously this is not COVID-19 in US)
  - If the patient tests positive, what are the chances the patient is sick? Is it 99%?

A – has the disease

B – has a positive test

$P(A|B)$

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} = \frac{0.99 \times 0.01}{0.0198} = 50\%$$

Practice Exercise 1 (Answer)

- Bayes Theorem
  - 99 percent of people who are sick test positive and 99 percent of the healthy people test negative. Only 1 percent of the people in the country are sick.
  - If the patient tests positive, what are the chances the patient is sick?
  - Further question: What is the chance of a false positive result?

	Diseased	Not Diseased	
Test +	99	99	198
Test -	1	9,801	9,802
	100	9,900	10,000

$$P(B) = \frac{198}{10000}$$

Practice Exercise 2

- Matrix Rank
- What is the rank of following matrix? Are they non-singular?

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 0 & 1 \\ 4 & 1 & 4 \end{bmatrix}$$

3

$$B = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 0 \\ 4 & 1 & 0 \end{bmatrix}$$

2

Practice Exercise 2: Answer

- Matrix Rank
- What is the rank of following matrix? Are they non-singular?

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 2 \\ 1 & 0 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 0 & 1 \\ 4 & 1 & 4 \end{bmatrix}$$

- rank(A)=3, rank(B)=2
- What about the rank of the matrix B+mI (I is identity matrix)?

Practice Exercise 3

- Hessian matrix

$$H = \begin{bmatrix} \frac{\partial f}{\partial x_1^2} & & \\ & & \\ & & \end{bmatrix} = \begin{bmatrix} 6x & -2 \\ -2 & \end{bmatrix}$$

$$\mathbf{H}f = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} & \cdots \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} & \cdots \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$f(x, y) = x^3 - 2xy - y^6$$

$3x^2 \rightarrow 6x$

$6y^5 \rightarrow 30y^4$

Practice Exercise 3: Answer

- Hessian matrix

$$\mathbf{H}f = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} & \cdots \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} & \cdots \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$f(x,y) = x^3 - 2xy - y^6$$

$$\mathbf{H}f(x,y) = \begin{bmatrix} f_{xx}(x,y) & f_{yx}(x,y) \\ f_{xy}(x,y) & f_{yy}(x,y) \end{bmatrix} = \begin{bmatrix} 6x & -2 \\ -2 & -30y^4 \end{bmatrix}$$

*Note: The Hessian matrix is a **symmetric** matrix, since the hypothesis of continuity of the second derivatives implies that the order of differentiation does not matter (Schwarz's theorem).*

Practice Exercise 4 (Link)

- Calculus: Chain rule → Later used in Neural Networks!

$f(x) = \cos^3(x)$

$[\cos(x)]^3$

$f'(x) = ?$

$f(x) = f_1(y) \cdot y_1(x)$

$\frac{\partial f(x)}{\partial x} = \frac{\partial f}{\partial y} \cdot \frac{\partial y}{\partial x}$

Choose 1 answer:

(A) $3\cos^2(x)$

(B) $-\sin^3(x)$

(C) $[1 - \sin^2(x)]\cos(x)$

(D) $-3\sin(x)\cos^2(x)$

$3\cos^2(x) \cdot (-\sin x)$

What’s next?

- In next week’s discussion, we will discuss:
  - Materials in the first 2 weeks: Decision tree, kNN and linear classification
  - Programming prep: Python, Google Colab, some useful packages (numpy, scikit-learn, matplotlib, etc)
- Useful resources for programming resources
  - Python/Numpy/Matplotlib tutorial: https://cs231n.github.io/python-numpy-tutorial/
  - Scikit-learn: https://scikit-learn.org/stable/tutorial/index.html

Thank you!

Q & A

CS M146 Discussion: Week 2
Decision Tree, Nearest Neighbors, ML Pipeline, Programming Prep

Junheng Hao

Friday, 01/15/2021

Roadmap

- Announcement
- Lecture Review
- Programming Prep for Problem Sets

Announcements

- 5:00pm PST, Jan. 15: Weekly quiz 2 released on Gradescope.
- **11:59pm PST, Jan. 17 (Sunday):** Weekly quiz 2 closed on Gradescope!
    - Start the quiz before **11:00pm PST, Jan. 17** to have the full 60-minute time
- 5:00pm, Jan. 15: Problem set 1 released on campuswire/CCLE, submission on Gradescope.
    - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
    - You do not need to submit code, only the results required by the problem set
    - Due on **11:59pm PST, Jan. 29 (Friday)**
- There is no class on **Jan. 18 (Monday)**, in observance of Martin Luther King Jr. Day.

About Quiz 2

- Quiz release date and time: Jan 15 **~~Jan 08~~, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **~~Jan 10~~, 2021 (Sunday) 11:59 PM PST** Jan 17
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week 2 Quiz" on GradeScope.
- Topics: Decision Tree, Nearest Neighbors, General machine learning basics and pipeline
- Question Types
  - True/false, multiple choices, and auto-graded short answers (fill blanks)
  - Some questions may include several subquestions.
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.
- More Info: https://campuswire.com/c/GB5E561C3/feed/57

Lecture Review

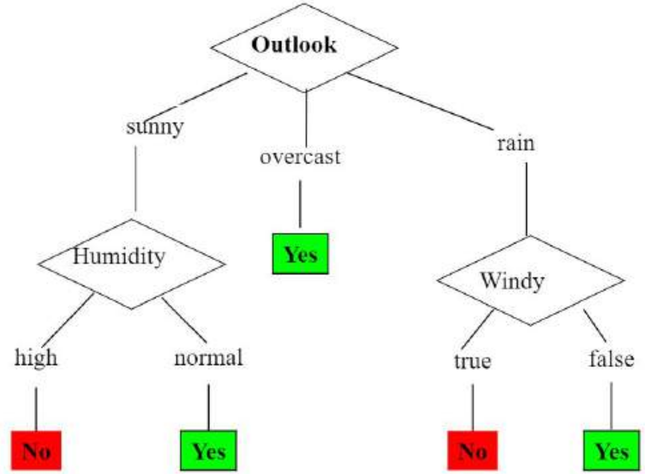
Decision Tree, Nearest Neighbors, ML Pipelines

Decision Tree

- Decision Tree Classification: From data to model

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

Info [9, 5] —



Decision Tree: Takeaway

- Choosing the Splitting Attribute
- At each node, available attributes are evaluated on the basis of separating the classes of the training examples.
- A goodness function (information measurement) is used for this purpose:
  - **Information Gain**
  - Gain Ratio*
  - Gini Index*

Decision Tree: Attribute Selection

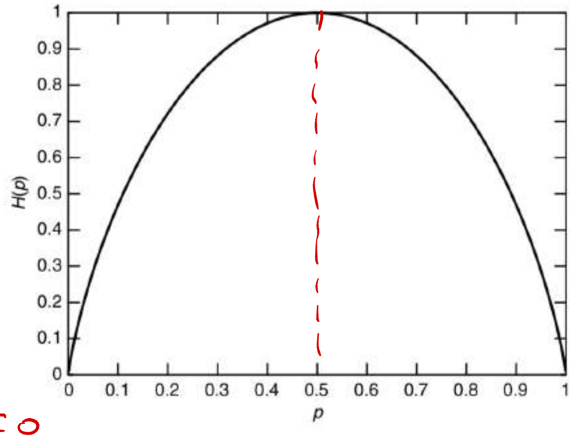
- Which is the best attribute?
  - The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the “purest” nodes
- Popular *impurity criterion*: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces
- Strategy: choose attribute that results in greatest information gain

Decision Tree: Entropy of Random Variable

$$X = \begin{cases} 1 & \text{with probability} \quad p \\ 0 & \text{with probability} \quad 1-p \end{cases}$$

$p=0.5$

$$H(X) = -p\log p - (1-p)\log(1-p) \stackrel{\text{def}}{=} H(p)$$



$0 \log 0 = 0$

Decision Tree: Attribute Selection

- Information in a split with ***x*** items of one class, ***y*** items of the second class

$$\text{info}([x, y]) = \text{entropy}\left(\frac{x}{x+y}, \frac{y}{x+y}\right)$$

$$= -\frac{x}{x+y}\log\left(\frac{x}{x+y}\right) - \frac{y}{x+y}\log\left(\frac{y}{x+y}\right)$$

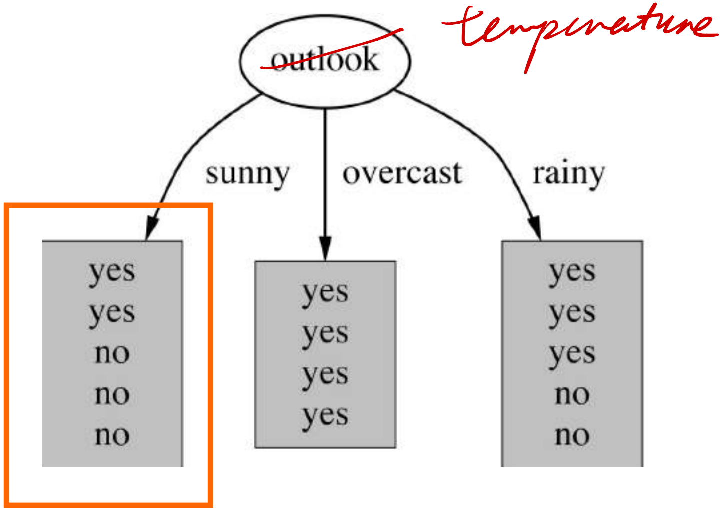
$4C_1$ $3C_2$

$$= -\frac{3}{7}\log\frac{3}{7} - \frac{4}{7}\log\frac{4}{7}$$

Attribute: “Outlook” = “Sunny”

- “Outlook” = “Sunny”: 2 and 3 split

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -\frac{2}{5}\log(\frac{2}{5}) - \frac{3}{5}\log(\frac{3}{5}) = 0.971\ \text{bits}$$

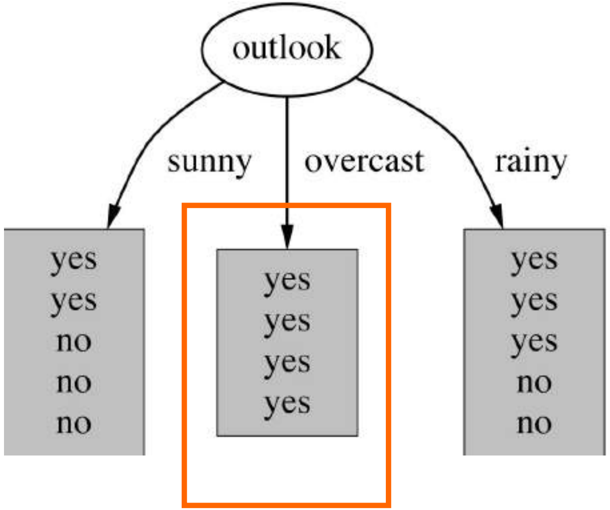


Attribute: “Outlook” = “Overcast”

- “Outlook” = “Overcast”: 4/0 split

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) \boxed{-0\log(0)} = 0 \text{ bits}$$

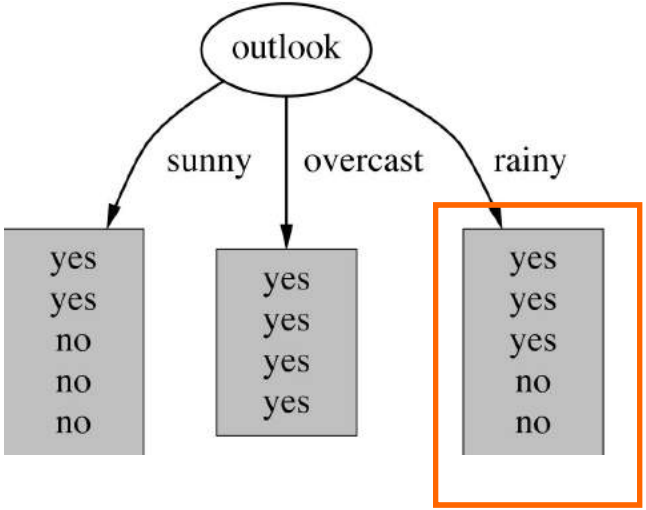
Note: log(0) is not defined, but we evaluate 0*log(0) as zero.



Attribute: “Outlook” = “Rainy”

- “Outlook” = “Rainy”:

$$\text{info}([3,2]) = \text{entropy}(3/5,2/5) = -\frac{3}{5}\log(\frac{3}{5}) - \frac{2}{5}\log(\frac{2}{5}) = 0.971 \text{ bits}$$



Expected Information of Attribute "Outlook"

Expected information for attribute:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$

$$= 0.693 \text{ bits}$$

Compute Information Gain

Information gain:

(information before split) – (information after split)

$$\text{gain("Outlook")} = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

Information gain for attributes from all weather data:

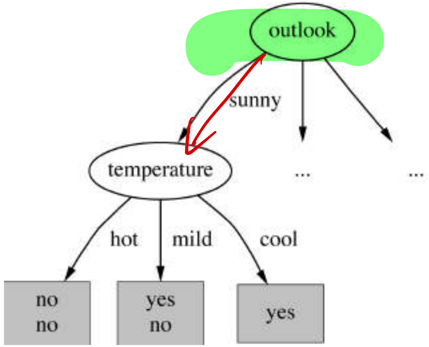
$$\text{gain("Outlook")} = 0.247 \text{ bits}$$

$$\text{gain("Temperature")} = 0.029 \text{ bits}$$

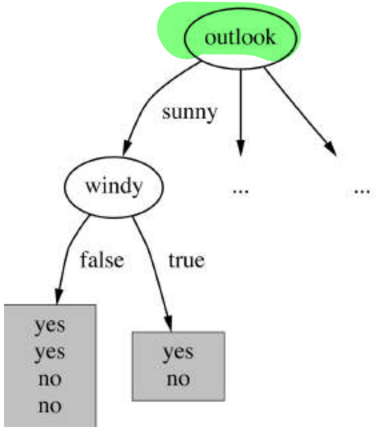
$$\text{gain("Humidity")} = 0.152 \text{ bits}$$

$$\text{gain("Windy")} = 0.048 \text{ bits}$$

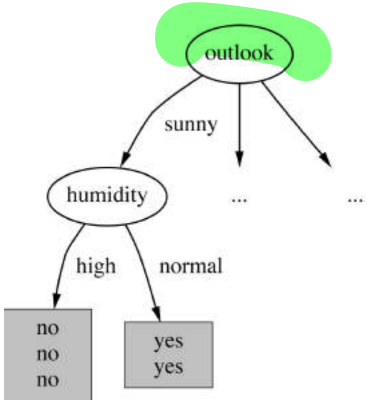
Continue to Split



gain("Temperature") = 0.571 bits



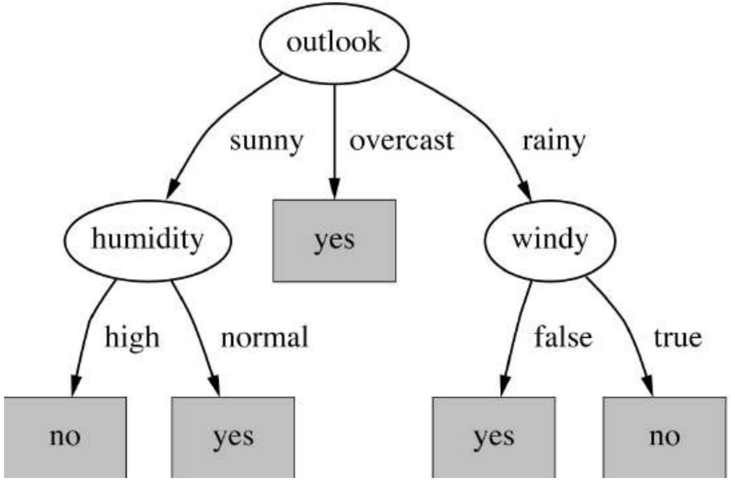
gain("Windy") = 0.020 bits



gain("Humidity") = 0.971 bits

Decision Tree: Example for Practice

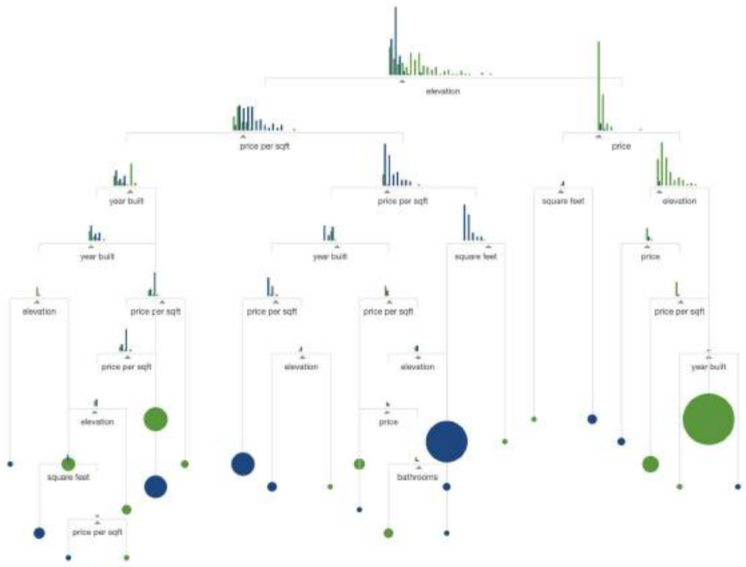
Final Tree



- Note: Not all leaves need to be pure. Sometimes identical instances have different classes.
- Splitting can stop when data can’t be split any further.

Decision Tree: Visual Tutorials

- Demo links
  - http://www.r2d3.us/visual-intro-to-machine-learning-part-1/
  - http://explained.ai/decision-tree-viz/

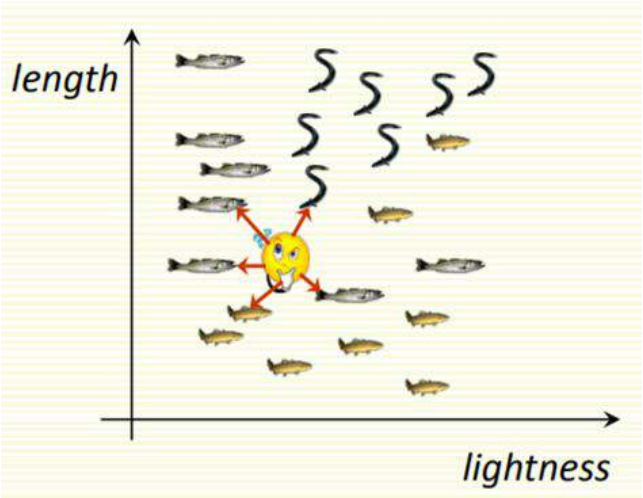


KNN

- Classify an unknown example with the most common class among K nearest examples
  - “Tell me who your neighbors are, and I’ll tell you who you are”
- Example
  - K = 3
  - 2 sea bass, 1 salmon
  - Classify as sea bass

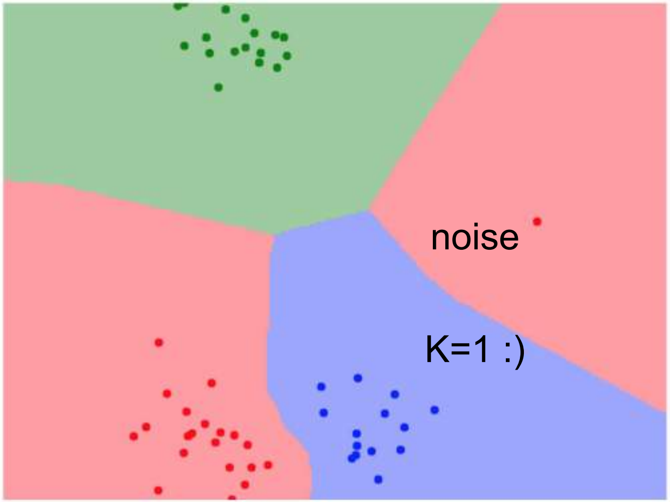
KNN: Multiple Classes

- Easy to implement for multiple classes
- Example for K = 5
  - 3 fish species: salmon, sea bass, eel
  - 3 sea bass, 1 eel, 1 salmon → classify as sea bass

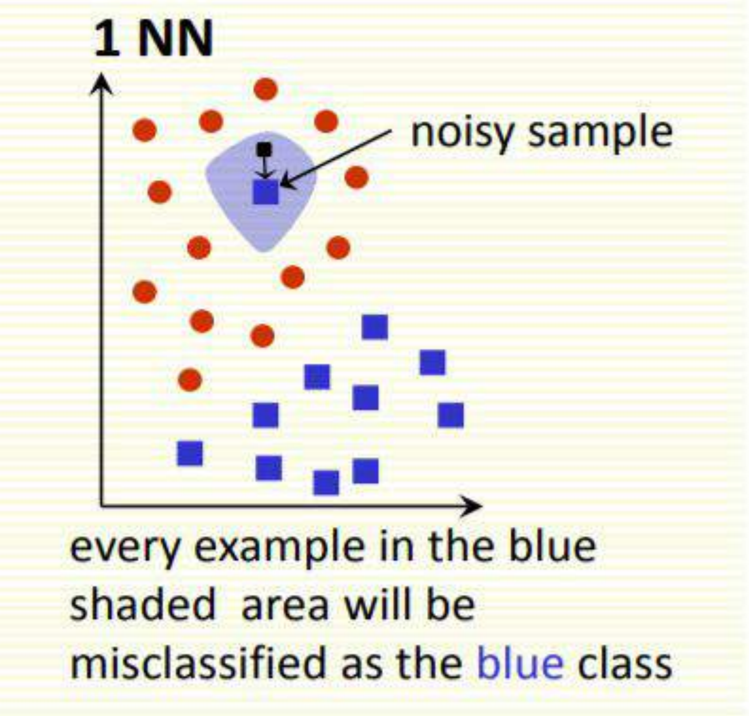


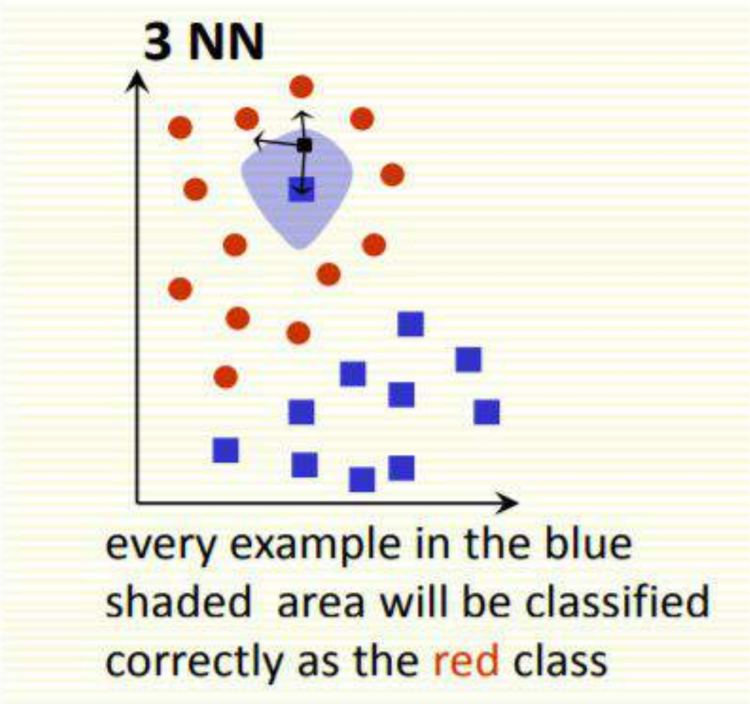
KNN: How to Choose K?

- In theory, if infinite number of samples available, the larger K, the better classification result you'll get.
- Caveat: all K neighbors have to be close
  - Possible when infinite # samples available
  - Impossible in practice since # samples if finite
- Should we "tune" K on training data?
  - Underfitting → Overfitting
- K = 1 → sensitive to "noise" (e.g. see right)



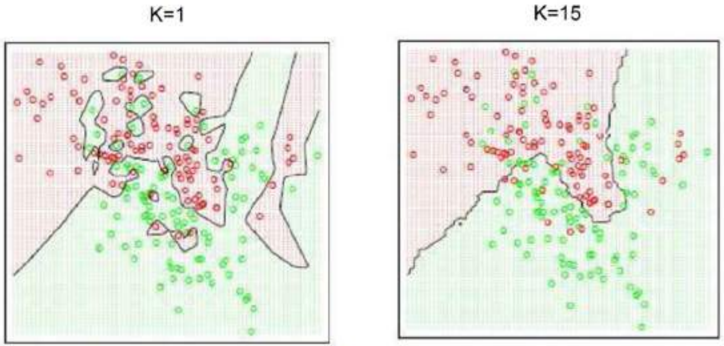
KNN: How to Choose K?



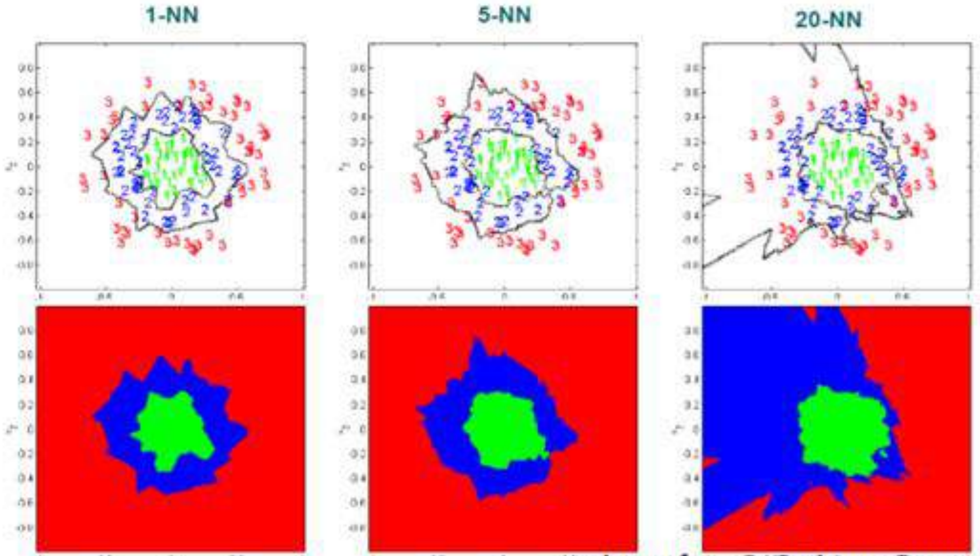


KNN: How to Choose K?

- Larger K gives smoother boundaries, better for generalization
  - Only if locality is preserved
  - K too large → looking at samples too far away that are not from the same class
- Can choose K through cross-validation



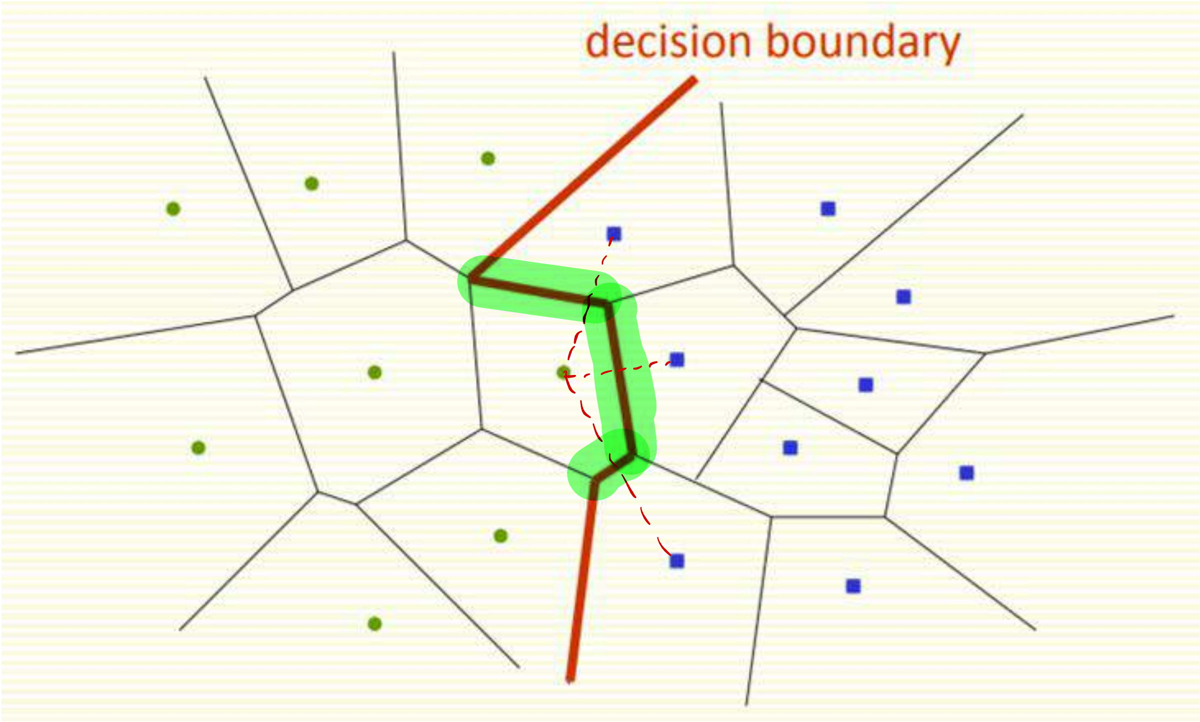
Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)



picture from R. Gutierrez-Osuna

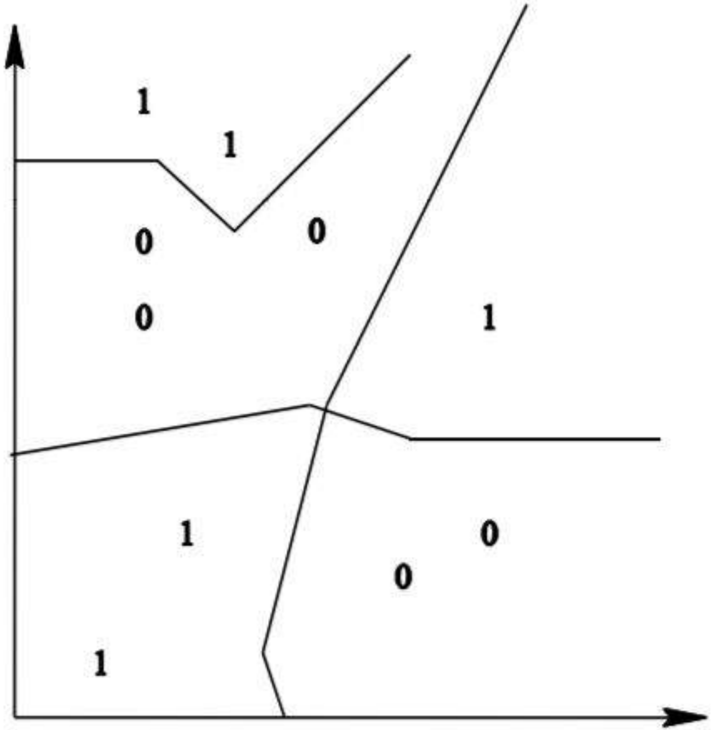
KNN: Decision Boundary

- Voronoi diagram



KNN: Decision Boundary

- Decision boundaries are formed by a subset of the Voronoi Diagram of the training data
- Each line segment is equidistant between two points of opposite class
- The more examples that are stored, the more fragmented and complex the decision boundaries can be.



KNN: Distance

- If we use Euclidean Distance to find the nearest neighbor:

$$D(a,b) = \sqrt{\sum_k (a_k - b_k)^2}$$

- Euclidean distance treats each feature as equally important
- Sometimes, some features (or dimensions) may be much more discriminative than other features

KNN: Distance

- Feature 1 gives the correct class: 1 or 2
- Feature 2 gives irrelevant number from 100 to 200
- Dataset: [1, 150], [2, 110]
- Classify [1, 100]

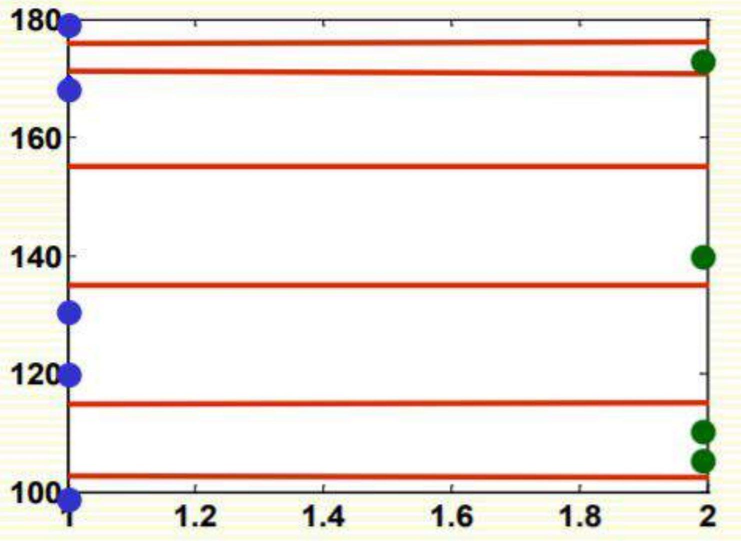
$$D\left(\begin{bmatrix}1\\100\end{bmatrix}, \begin{bmatrix}1\\150\end{bmatrix}\right) = \sqrt{(1-1)^2 + (100-150)^2} = 50$$

$$D\left(\begin{bmatrix}1\\100\end{bmatrix}, \begin{bmatrix}2\\110\end{bmatrix}\right) = \sqrt{(1-2)^2 + (100-110)^2} = 10.5$$

- Use Euclidean distance can result in wrong classification
- Dense Example can help solve this problem

KNN: Distance

- Decision boundary is in red, and is really wrong because:
  - Feature 1 is discriminative, but its scale is small
  - Feature gives no class information but its scale is large, which dominates distance calculation



KNN: Feature Normalization

- Normalize features that makes them be in the same scale
- Different normalization approaches may reflect the result
- Linear scale the feature in range [0,1]:

$$f_{new} = \frac{f_{\text{old}} - f_{\text{old}}^{\text{min}}}{f_{\text{old}}^{max} - f_{\text{old}}^{\text{min}}}$$

- Linear scale to 0 mean standard deviation 1(Z-score):

$$f_{new} = \frac{f_{\text{old}} - \mu}{\sigma}$$

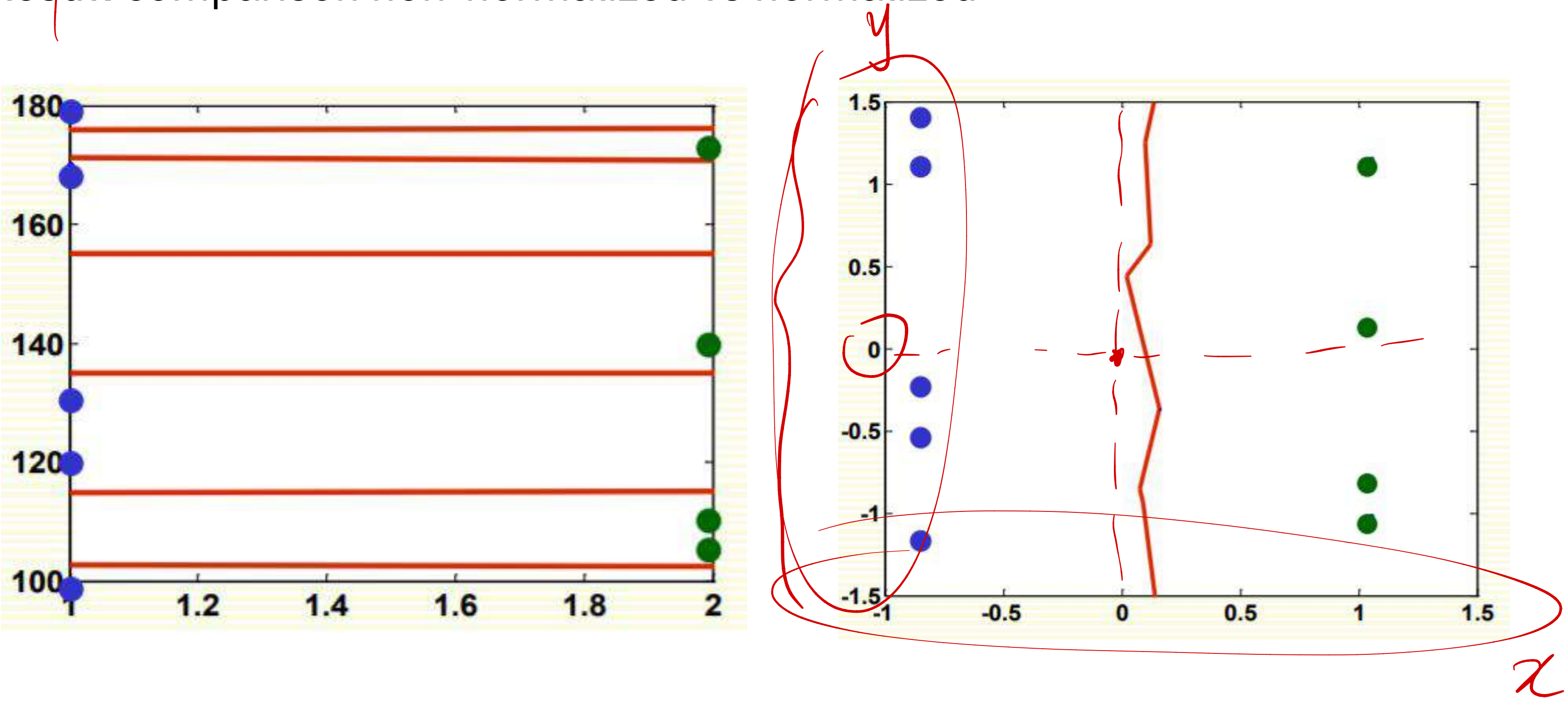
$$\mu = \frac{1}{N}\sum_i f_i$$

$$\sigma^2 = \frac{1}{N-1}\sum (f_i - \mu)^2$$

KNN: Feature Normalization

(1, 10M)

- Result comparison non-normalized vs normalized



KNN: Feature Weighting

- Scale each feature by its importance for classification

$$D(a,b) = \sqrt{\sum_k w_k (a_k - b_k)^2}$$

- Use prior/domain knowledge to set the weight w
- Use cross-validation to learn the weight w

KNN: Computational Complexity

- Suppose ***n*** examples with dimension ***d***
- Complexity for kNN training?
- Complexity for kNN ~~training~~? testing / inference?
  - For each point to be classified:
  - Complexity for computing distance to one example
  - Complexity for computing distances to all examples
  - Find ***k*** closest examples
- Is it expensive for a large number of queries?

KNN: Summary

10M

- Advantages:
  - Can be applied to the data from any distribution
  - The decision boundary is not necessarily to be linear
  - Simple and Intuitive
  - Good Classification with large number of samples

100 100 100 100²

10K

- Disadvantages:
  - Choosing k may be tricky
  - Test stage is computationally expensive
    - No training stage, time-consuming test stage
    - Usually we can afford long training step but fast testing speed
  - Need large number of examples for accuracy

100 100 new 10K

UCLA Engineer Change.

ML Pipeline: Evaluation Your Models

kNN

1-NN
3-NN
:
9-NN

train valid

best k = 5

5-NN

test

accuy:

Training data (set)

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{N}}, y_{\mathsf{N}})\}$
- They are used for learning $h(\cdot)$

Test (evaluation) data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{M}}, y_{\mathsf{M}})\}$
- They are used for assessing how well $h(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Development (or validation) data

- L samples/instances: $\mathcal{D}^{\text{DEV}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathsf{L}}, y_{\mathsf{L}})\}$
- They are used to optimize hyperparameter(s).

Training data, validation and test data should *not* overlap!

1 2 3 4 5

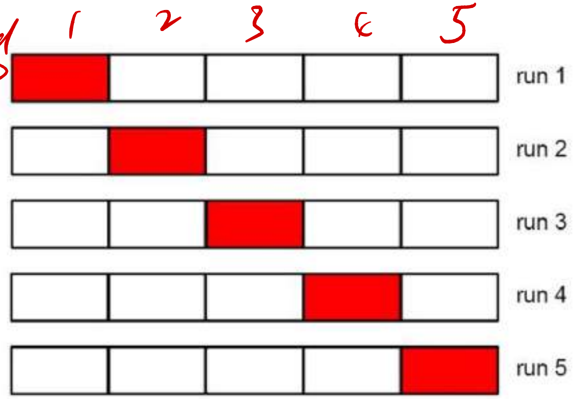
1 4

Cross Validation

- We split the training data into K equal parts (termed folds or splits).
- We use each part *in turn* as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that *on average*, the model performing the best

$K = 5$: 5-fold cross validation

train/valid



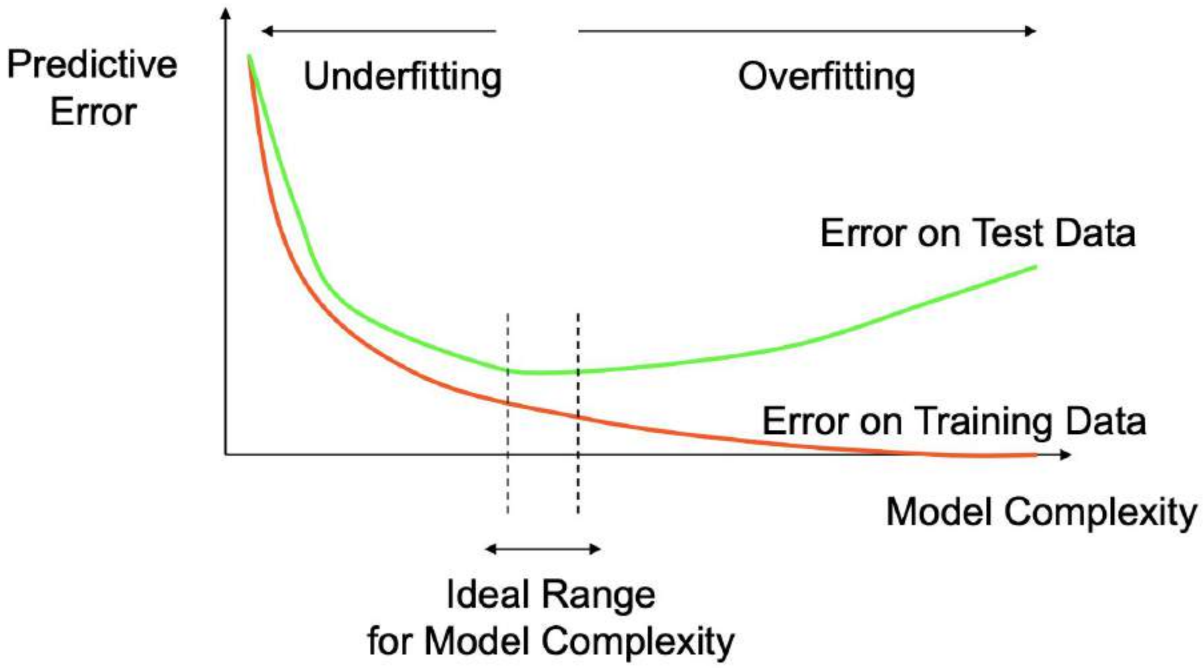
Special case: when $K = N$, this will be leave-one-out (LOO).

KNN=1

KNN(K=3)

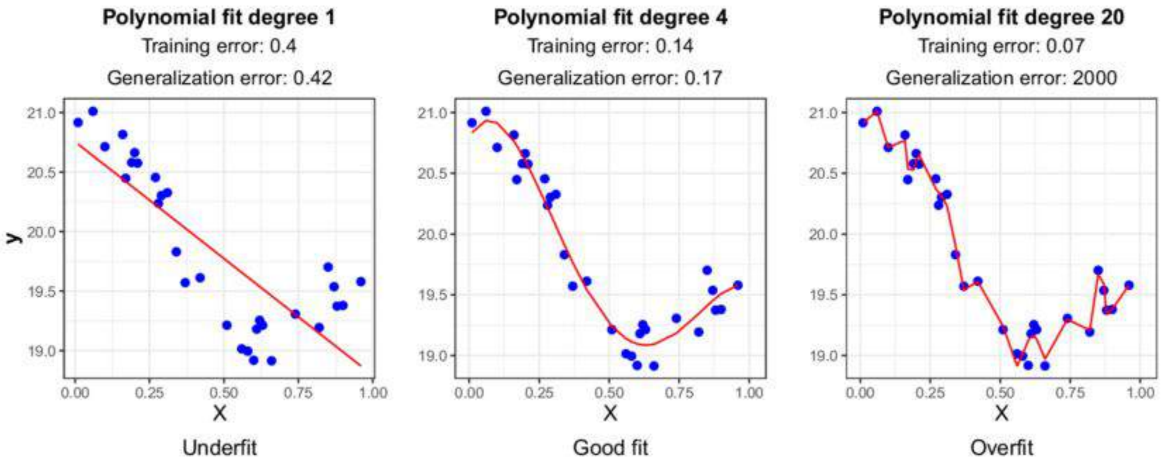
test

Analyze Your Model: Underfit or Overfit?



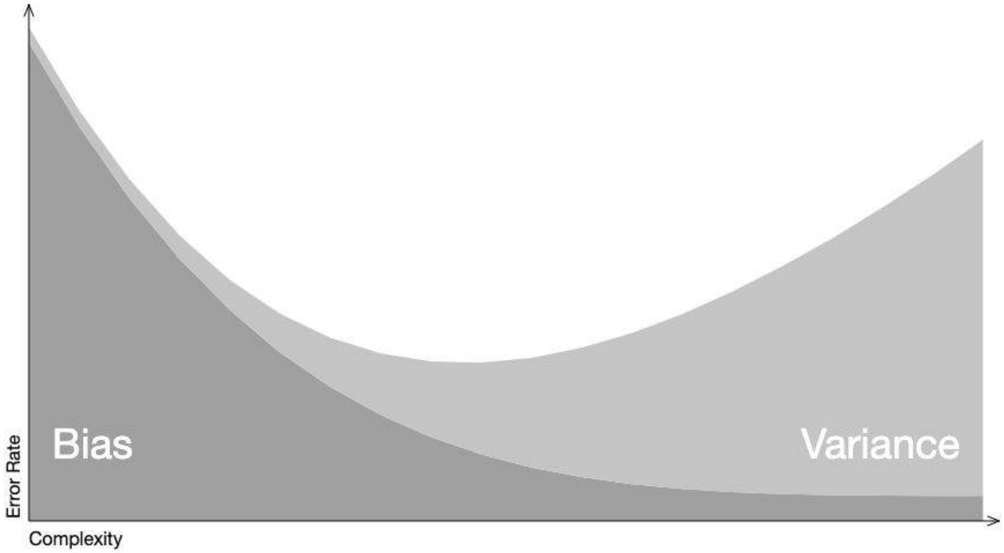
Analyze Your Model: Underfit or Overfit?

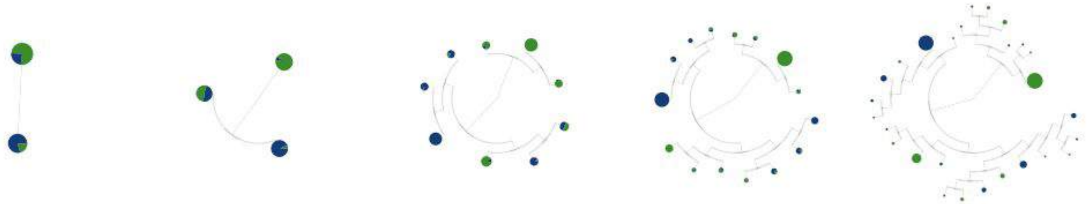
- Another example on regression



Analyze Decision Tree: Too simple or too complex?

- Examples on Decision Tree
- Another two concepts: Model Bias & Variance
- Demo: [Link]





Programming Prep Guide

“Do it local”: Python & Jupyter Notebook

- **Step 1:** Install Anaconda (with Python 3.X and Jupyter Notebooks)
- **Step 2:** Try out Python in command line and open Jupyter Notebooks
- **Step 3:** Familiarize yourself with Python 3
- **Step 4:** Use Jupyter Notebooks for coding and writing together
- **Step 5:** Customize your Python environment and install Python packages
  - Example packages: Numpy, Pandas, Matplotlib

Note: This slide is only intended for students who want to program on local desktop instead of Google Colab.

Where is your Python?

- Install Conda/Anaconda
  - Conda: https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html
  - Anaconda: https://docs.anaconda.com/anaconda/install/mac-os/
- Install Jupyter Notebook from anaconda (this step may be skipped once Anaconda is installed)
  - Link: https://jupyter.org/install
  - Command Line: conda install -c conda-forge notebook
- Check out Python and Jupyter notebook
  - Command Line: python or ipython
  - Version/Source: python --version or which python
  - Open Jupyter Notebook: jupyter notebook (automatically into something URL like: http://localhost:8888/tree)

Note: This slide is only intended for students who want to program on local desktop instead of Google Colab.

Create customized Python environment

- Checklist:
  - Create a customized virtual environment
  - Activate/Deactivate your environment
  - Install packages for your virtual environment
- Helpful links:
  - Managing conda environment: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html

A Program Notebook: Write and Code

- Apply both on Jupyter Notebook and Google Colab!

- Checklist:
  - Identify **Markdown** cell and **Code** cell
  - Learn how to use markdown and latex to input math formula
  - Run Python code

- Markdown tutorial → *It is a notebook interface!*
  - Checklist: paragraph, bold, italic, list, code (courier), math formula (in latex)
  - Link: https://www.markdowntutorial.com/

- Latex → *It is for typing math symbols and equations!*
  - No need to install Tex or Mactex
  - Cheatsheet: http://tug.ctan.org/info/undergradmath/undergradmath.pdf

Checklist: Python, Numpy, Pandas, Matplotlib

- Shown in the demo
- Python
  - Data types and control flow
- Numpy
  - Array and matrix
  - Matrix operation
  - Broadcasting
- Pandas
  - Data load and export
  - Dataframe operations
- Matplotlib
  - Plot types, settings and output figure files
- Scikit-learn
  - ML pipeline (data prep, model selection, train and development, evaluation)

Demo [Link]

- **Google Colab: A starter guide**
  - Create and connect online codebook
  - Run code and commands
  - Save and output results
- **Text cell**
  - Markdown and Latex
- **Code cell**
  - Python
  - Numpy
  - Pandas
  - Matplotlib

Thank you!

Q & A

UCLA Samueli Computer Science

CS M146 Discussion: Week 3

Perceptron, Linear Models, Optimization

Junheng Hao

Friday, 01/22/2021

Roadmap

- Announcement
- Perceptron & Linear Models
- Optimization, MLE

Announcements

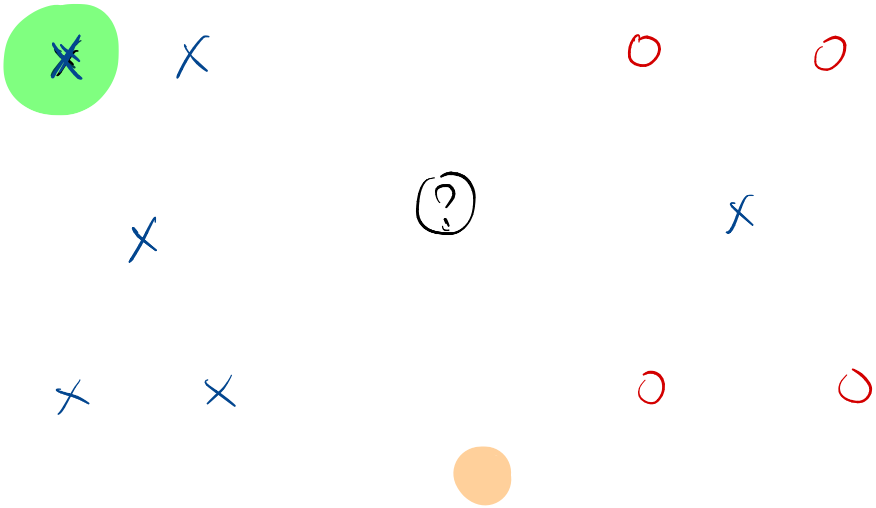
- 5:00 pm PST, Jan. 22: Weekly Quiz 3 released on Gradescope.
- **11:59 pm PST, Jan. 24 (Sunday):** Weekly quiz 3 closed on Gradescope!
  - Start the quiz before **11:00 pm PST, Jan. 24** to have the full 60-minute time
- Problem set 1 released on campuswire/CCLE, submission on Gradescope.
  - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
  - You do not need to submit code, only the results required by the problem set
  - Due on **11:59pm PST, Jan. 29 (Friday)**

About Quiz 3

- Quiz release date and time: **Jan 22, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Jan 24, 2021 (Sunday) 11:59 PM PST**
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week 3 Quiz" on GradeScope.
- Topics: **Perceptron, Linear Models**
- Question Types
  - True/false, multiple choices, and ~~auto-graded short answers (fill blanks)~~
  - Some questions may include several subquestions.
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.

*One more quiz of K-NN

- True/False: The training error of K-NN will be zero when K = 1, irrespective of the dataset.



Math Reminder: Normal vector and plane

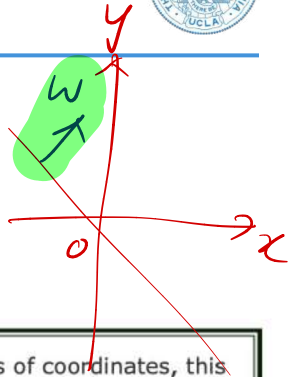
UCLA Engineer Change.

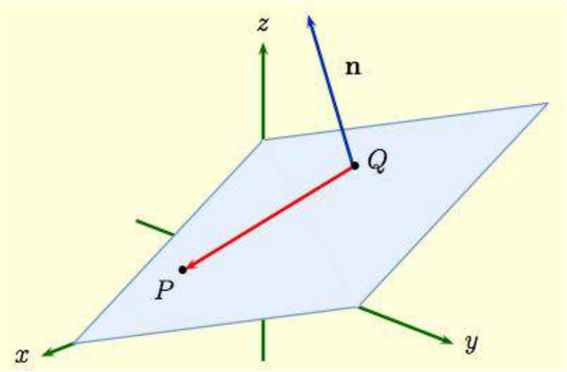
- A normal vector is a vector perpendicular to another object, such as a surface or plane.

$\Rightarrow Ax + By - m = 0$

$m = 0$ $(x, y) \neq (0, 0)$

$\vec{w} = (A, B)$





Let $Q(a, b, c)$ be a fixed point in the plane, $P(x, y, z)$ an arbitrary point in the plane, and $\mathbf{n} = \langle A, B, C \rangle$ the normal to the plane. If

$$\mathbf{b} = \langle a, b, c \rangle, \qquad \mathbf{r} = \langle x, y, z \rangle,$$

the vector

$$\overrightarrow{QP} = \mathbf{r} - \mathbf{b} = \langle x - a, y - b, z - c \rangle$$

lies in the plane, and is **perpendicular** to $\mathbf{n}$.

Thus $\mathbf{n} \cdot (\mathbf{r} - \mathbf{b}) = 0$. In terms of coordinates, this becomes

$$\boxed{\langle A, B, C \rangle \cdot \langle x - a, y - b, z - c \rangle = 0,}$$

where $\mathbf{n} = \langle A, B, C \rangle$. In other words, we get the **point-normal** equation

$$A(x - a) + B(y - b) + C(z - c) = 0.$$

for a plane.

$\Rightarrow Ax + By + Cz - m = 0$

$n = (A, B, C)$

Credit: https://web.ma.utexas.edu/users/m408m/Display12-5-4.shtml

Math Reminder: Normal vector and plane

- A normal vector is a vector perpendicular to another object, such as a surface or plane.

As promised, we return the the question of finding the equation for a plane from the location of three points, say

$$Q(x_1, y_1, z_1), \qquad R(x_2, y_2, z_2), \qquad S(x_3, y_3, z_3)$$

The fact that the cross-product $\mathbf{a} \times \mathbf{b}$ is perpendicular to both $\mathbf{a}$ and $\mathbf{b}$ makes it very useful when dealing with normals to planes.

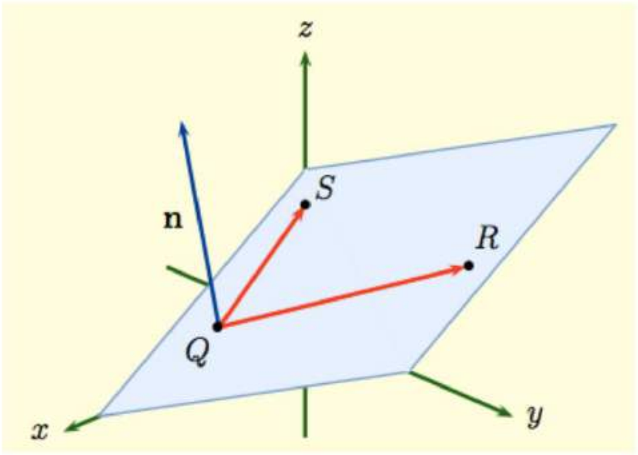
Let

$$\mathbf{b} = \langle x_1, y_1, z_1 \rangle, \quad \mathbf{r} = \langle x_2, y_2, z_2 \rangle, \quad \mathbf{s} = \langle x_3, y_3, z_3 \rangle.$$

The vectors

$$\overrightarrow{QR} = \mathbf{r} - \mathbf{b}, \qquad \overrightarrow{QS} = \mathbf{s} - \mathbf{b},$$

then lie in the plane. The normal to the plane is given by the cross product $\mathbf{n} = (\mathbf{r} - \mathbf{b}) \times (\mathbf{s} - \mathbf{b})$. Once this normal has been calculated, we can then use the point-normal form to get the equation of the plane passing through Q, R, and S.



Math Reminder: Normal vector and plane

- Demo Calculation Example

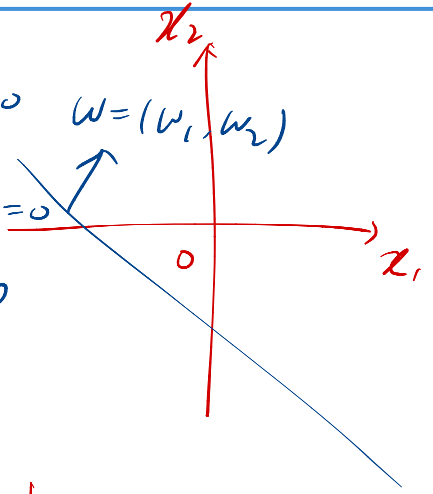
$Q(-1, 1, 2)$ $\quad R(-4, 2, 2)$

$w^T x + b = 0$

$w_1 x_1 + w_2 x_2 + b = 0$

$Ax + By + b = 0$

$w = (w_1, w_2)$



$\vec{QR} = (-3, 1, 0)$

$\vec{QS} = (-1, 0, 3)$

$\vec{n} = (x, y, 1)$

$$\begin{cases} \vec{n} \cdot \vec{QR} = 0 \\ \vec{n} \cdot \vec{QS} = 0 \end{cases}$$

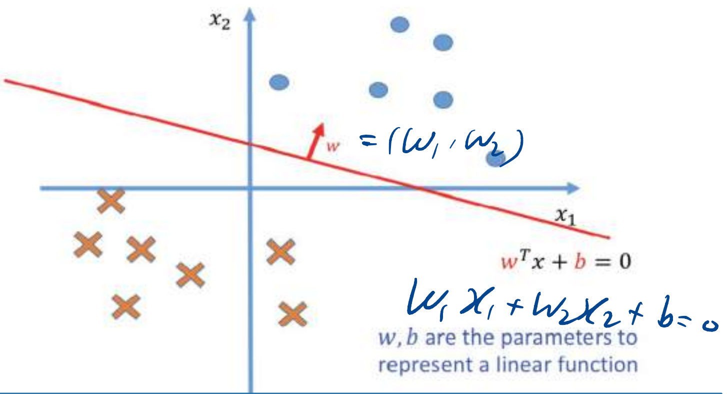
$$n = \begin{vmatrix} i & j & k \\ -3 & 1 & 0 \\ -1 & 0 & 3 \end{vmatrix} = 3i + 9j + k$$

$\vec{n} = (3, 9, 1) \Rightarrow 3x + 9y + z - 8 = 0$

Perceptron: Overview

$X \in R^{N \times D}$

- Instance (feature vectors): $\boldsymbol{x} \in \mathbb{R}^D$
- Label: $y \in \{-1, +1\}$
- Model/Hypotheses:
  $H = \{h | h : \mathbb{X} \rightarrow \{-1, +1\}, h(\boldsymbol{x}) = sign(\sum_{d=1}^{D} w_d x_d + b)\}$.
- Learning goal: $y = h(\boldsymbol{x})$
  - Learn $w_1, \ldots, w_D, b$.
  - Parameters: $w_1, \ldots, w_D, b$.
  - $\boldsymbol{w}$: weights, b: bias



$a = h(x)$

$1 \quad 1$

$-1 \quad -1$

Iteratively solving one case at a time

- REPEAT
- Pick a data point $\boldsymbol{x}_n$
- Compute $a = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n$ using the *current* $\boldsymbol{w}$
- If $a y_n > 0$, do nothing. Else,

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n \boldsymbol{x}_n$$

- UNTIL converged.

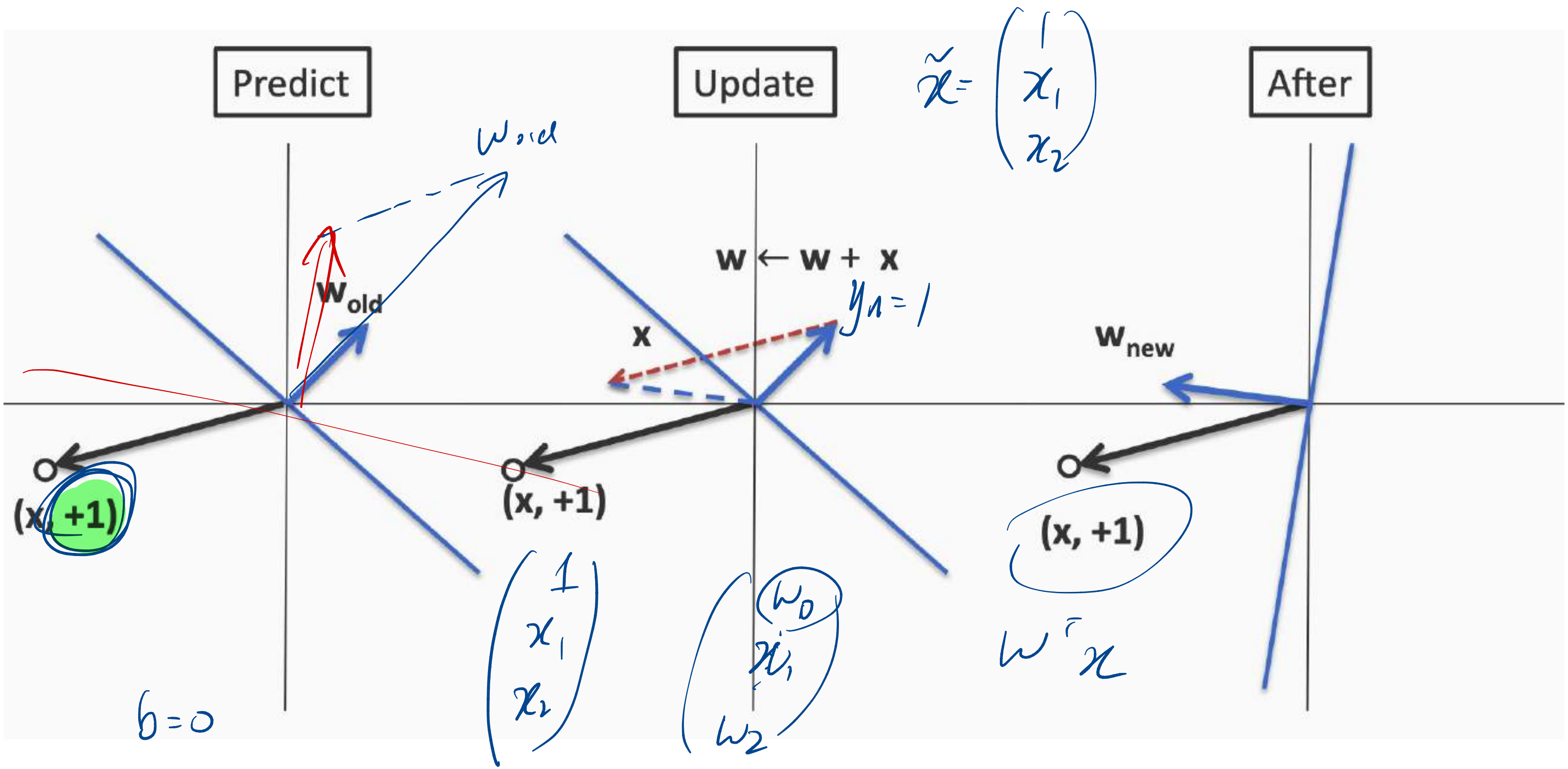
Perceptron: Convergence

XOR

- If training data is **not linearly separable**, the algorithm does not converge.
- If the training data is **linearly separable**, the algorithm stops in a finite number of steps (converges).

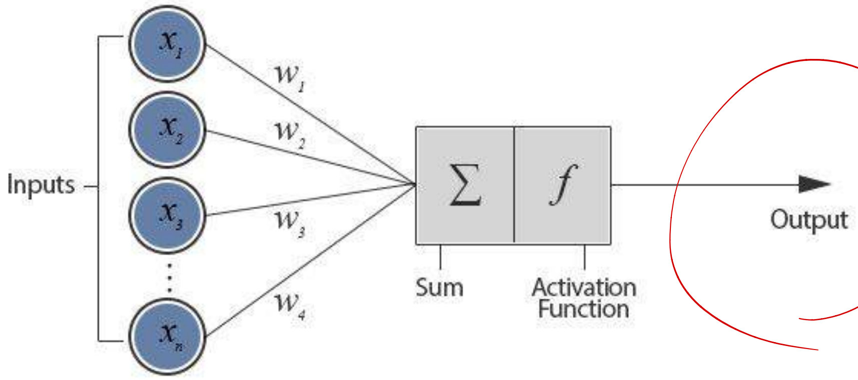
- Let $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$ be a sequence of training examples such that $\|\boldsymbol{x}_n\|_2 \leq R$ and label $y_n \in \{-1, +1\}$.
- Suppose there exists a unit vector $\boldsymbol{u} \in \mathbb{R}^D$ such that for some $\gamma > 0$, we have $y_n \boldsymbol{u}^{\mathrm{T}} \boldsymbol{x}_n \geq \gamma$.
- Then the Perceptron algorithm will make at most $\frac{R^2}{\gamma^2}$ mistakes on the training sequence.

Perceptron: Update (Geometry)



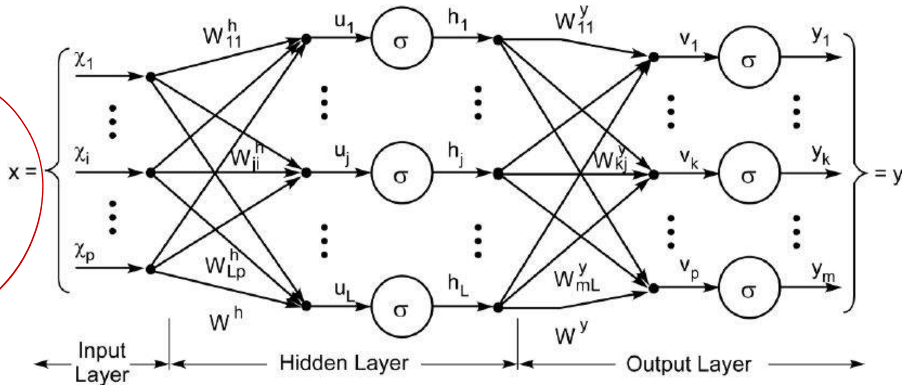
Perceptron: Connect to Neural Network

A Single Perceptron



(0, 1)

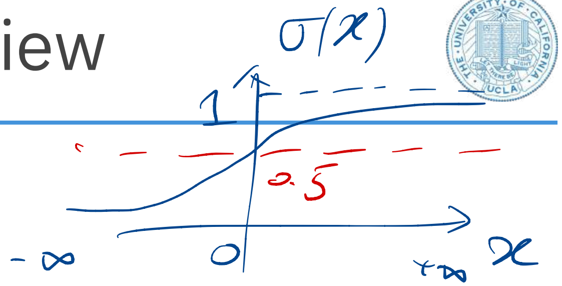
Multi-layer Neural Network



Question: Can a single perceptron classify XOR data? How about 2-layer perceptrons?

UCLA Engineer Change.

Logistic Regression: Overview



$b+w^Tx=0$

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$

$\sigma(0)=0.5$

$$p(y_n|\boldsymbol{x}_n; b, \boldsymbol{w}) = \begin{cases} h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = \sigma(b+\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ = 1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n) = 1 - \sigma(b+\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

Compact expression, exploring that y_n is either 1 or 0

$$\log p(y_n|\boldsymbol{x}_n; b; \boldsymbol{w}) = h_{\boldsymbol{w},b}(\boldsymbol{x}_n)^{y_n}[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]^{1-y_n}$$

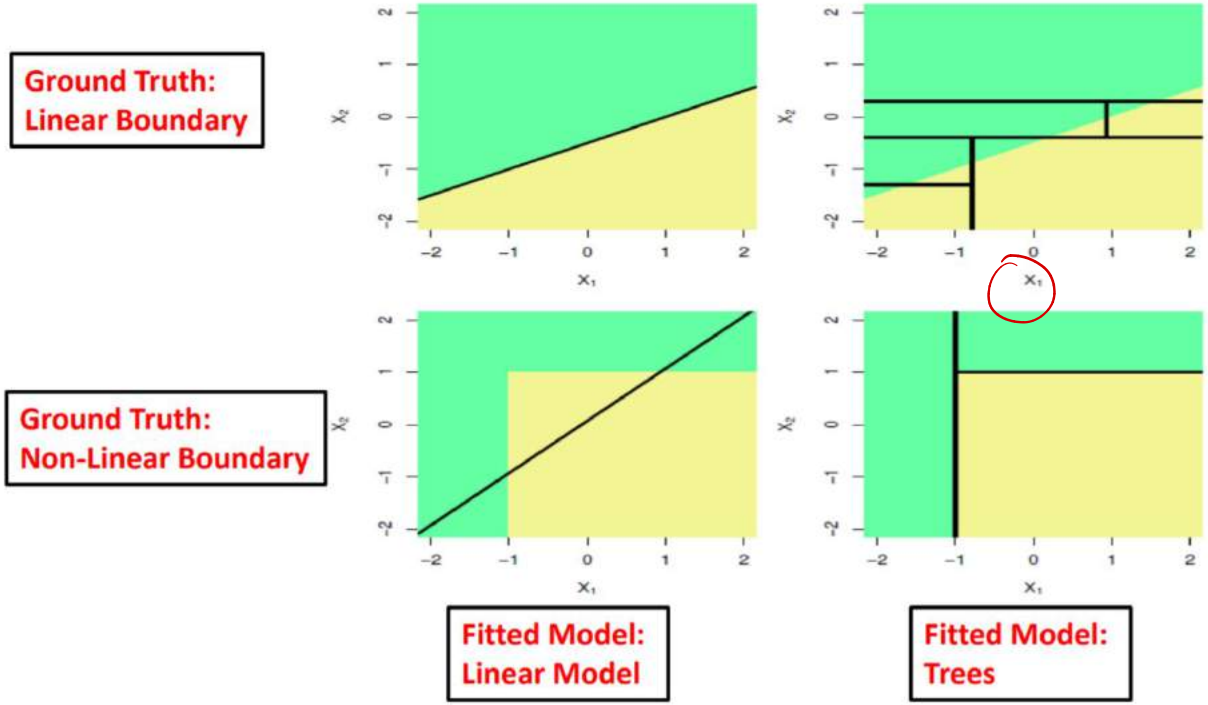
$y_n = \begin{cases} 1 \\ 0 \end{cases}$

Log-likelihood of the whole training data $\mathcal{D}$

$$l(\boldsymbol{w}, b) = \sum_n \{y_n \log h_{\boldsymbol{w},\boldsymbol{b}}(\boldsymbol{x}_n) + (1-y_n)\log[1 - h_{\boldsymbol{w},b}(\boldsymbol{x}_n)]\}$$

Linear Models

- Compare: Decision Tree, Nearest Neighbors, Perceptron

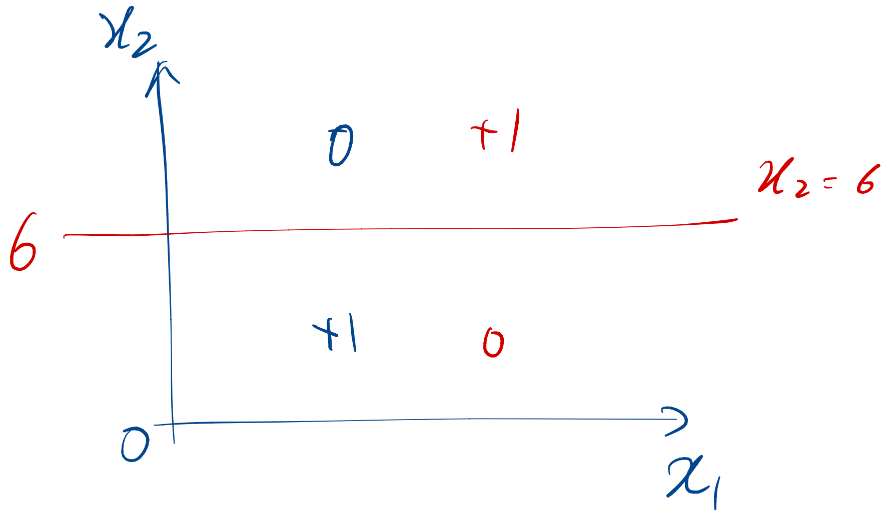


UCLA Engineer Change.

Decision Boundary: Quiz

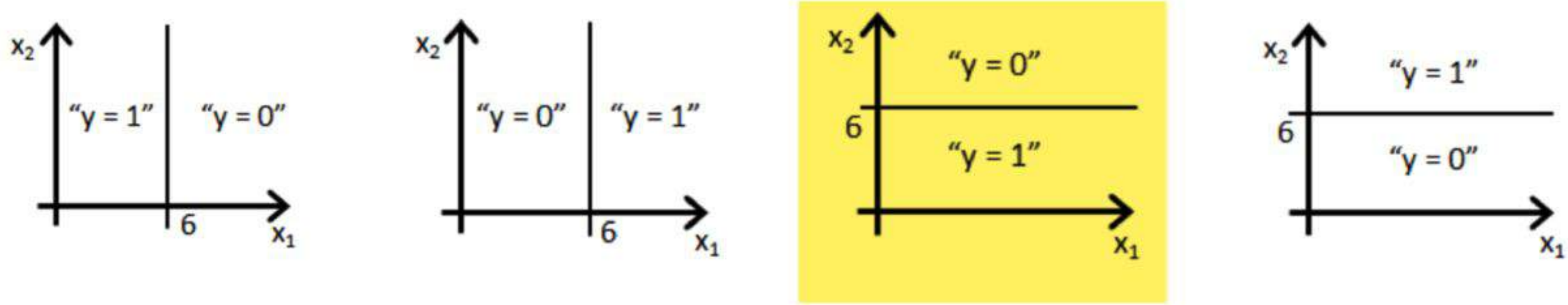
Suppose you train a logistic classifier $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$. Suppose $\theta_0 = 6, \theta_1 = 0, \theta_2 = -1$. Which of the following figures represents the decision boundary found by your classifier?

$$\theta_0 + \underbrace{\theta_1}_{0} x_1 + \theta_2 x_2 = 0$$



Decision Boundary: Quiz

Suppose you train a logistic classifier $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$. Suppose $\theta_0 = 6, \theta_1 = 0, \theta_2 = -1$. Which of the following figures represents the decision boundary found by your classifier?



Unconstrained Optimization

- Convex Function and Convexity
- Closed-form solution
- Gradient Descent
- Newton's methods

$\min \; f(x)$ $\quad x^*$

$$\frac{\partial f(x)}{\partial x} = \underline{\qquad\qquad} = 0$$

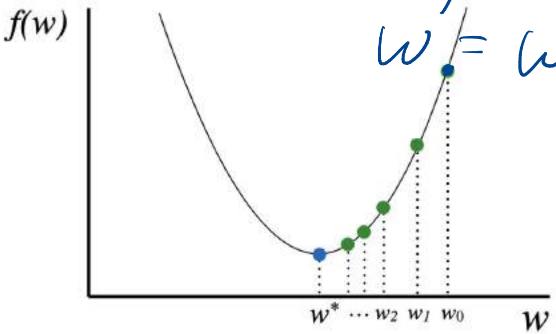
$$f(x) = x^2 - 3x$$

Gradient Descent

Start at a random point
Repeat
1. Determine a descent direction
2. Choose a step size
   Update

Until stopping criterion is satisfied

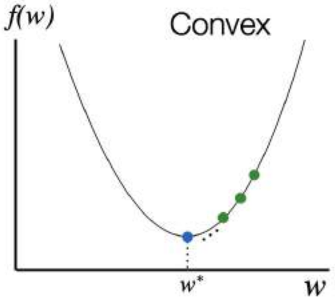
$$w' = w - \eta \frac{\partial f(w)}{\partial w}$$



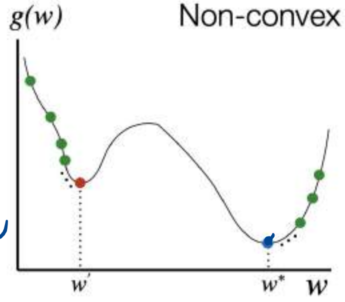
$f(x) = x$

$-\log(x)$

Where Will We Converge?



Any local minimum is a global minimum



Multiple local minima may exist

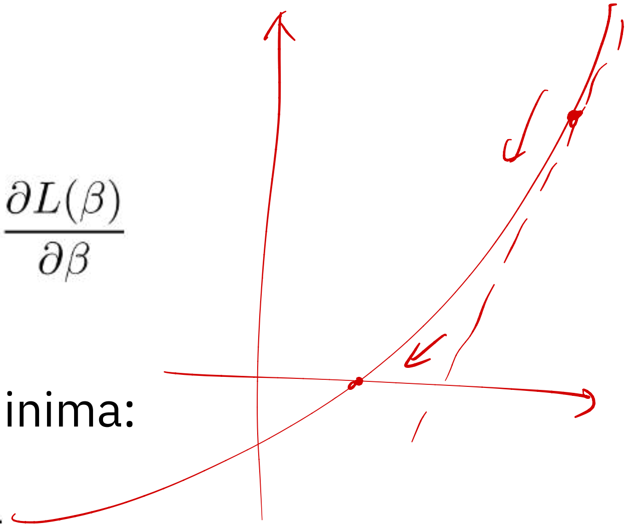
Newton's Method (Optional)

Definition:

$$\beta^{\text{new}} = \beta^{\text{old}} - \left(\frac{\partial^2 L(\beta)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial L(\beta)}{\partial \beta}$$

Apply Newton's methods on single variable to find minima:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$



From single variable to Multivariate Newton-Raphson Method

Newton's Method: Steps (Optional)

1. Initialize $x^{(0)}$
2. Calculate $\nabla f(x)$
3. Calculate $F(x)$
4. Initialize step $n = 0$ and start loops
   a. Calculate $\nabla f(x^{(n)})$
   b. Calculate $F(x^{(n)})$
   c. Calculate $[F(x^{(n)})]^{-1}$
   d. Update: $x^{(n+1)} = x^{(n)} - [F(x^{(n)})]^{-1} \cdot \nabla f(x^{(n)})$
   e. Update: $n = n + 1$
5. Exit Loop

Newton's Method: Example (Optional)

$$x^{(0)} = [3, -1, 0]$$
$$f(x_1, x_2, x_3) = (x_1 + 10x_2)^2 + 5(x_1 - x_3)^2 + (x_2 - 2x_3)^4$$

Newton's Method Example in one step:

- ➔ Calculate $\nabla f(x^{(n)})$
- ➔ Calculate $F(x^{(n)})$
- ➔ Calculate $[F(x^{(n)})]^{-1}$
- ➔ Update: $x^{(n+1)}$
- ➔ Update: $n = n + 1$

$$\nabla f(x^{(0)}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3}\right] = [16, -144, 22]$$

$$F(x^{(0)}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix} = \begin{bmatrix} 12 & 20 & -10 \\ 20 & 22 & -24 \\ -10 & -24 & 48 \end{bmatrix}$$

$$[F(x^{(0)})]^{-1} = \begin{bmatrix} -0.079 & 0.119 & 0.043 \\ 0.119 & -0.079 & -0.015 \\ 0.043 & -0.015 & 0.023 \end{bmatrix}$$

$$x^{(1)} = x^{(0)} - [F(x^{(0)})]^{-1} \cdot \nabla f(x^{(0)})$$
$$n = 1$$

Maximum Likelihood Estimation

Definition: The maximum likelihood estimator (MLE) $\hat{\theta}$, is the value of θ that maximizes $L(\theta)$.

min $-L(\theta)$

The log-likelihood function is defined by $l(\theta) = \log L(\theta)$. Its maximum occurs at the same place as that of the likelihood function.

- Using logs simplifies mathemetical expressions (converts exponents to products and products to sums)
- Using logs helps with numerical stabilitity

The same is true of the likelihood function times any constant. Thus we shall often drop constants in the likelihood function.

MLE: Logistic Regression

- Model

$$y = \sigma(X) = \frac{1}{1 + e^{-X^T \beta}}$$

- Original Objective

$$J(\beta) = -\frac{1}{n} \sum_i \left(y_i x_i^T \beta - \log\left(1 + \exp\{x_i^T \beta\}\right)\right)$$

N_1 $+$ y_i $\hat{y}_i$

N

N_2 $-$

$N_1 + N_2 = N$

MLE: Logistic Regression

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

where

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\begin{aligned}
g'(z) &= \frac{d}{dz} \frac{1}{1 + e^{-z}} \\
&= \frac{1}{(1 + e^{-z})^2} \left(e^{-z}\right) \\
&= \frac{1}{(1 + e^{-z})} \cdot \left(1 - \frac{1}{(1 + e^{-z})}\right) \\
&= g(z)(1 - g(z)).
\end{aligned}$$

MLE: Logistic Regression

$\theta^* = \frac{k}{n}$

Assuming that the m training examples were generated independently, we can then write down the likelihood of the parameters as

$$
\begin{aligned}
P(y=1 \mid x;\theta) &= h_\theta(x) \\
P(y=0 \mid x;\theta) &= 1-h_\theta(x)
\end{aligned}
$$

$$p(y \mid x;\theta) = (h_\theta(x))^y \, (1-h_\theta(x))^{1-y}$$

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X;\theta) \\
&= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\
&= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}}
\end{aligned}
$$

$\theta^* =$

$\frac{\partial l(\theta)}{\partial \theta}$

As before, it will be easier to maximize the log likelihood:

$$\boxed{
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log(1-h(x^{(i)}))
\end{aligned}
}$$

$p(C \mid B, A)$

Constrained Optimization

- Lagrange Multiplier

$f(x, y) = \$900 \cdot x + \$5 \cdot y$

profit

$$\boxed{\begin{aligned} &\text{maximize } f(x, y) \\ &\text{subject to: } g(x, y) = 0 \end{aligned}}$$

2

$2+1 \longrightarrow$

$$\underline{\mathcal{L}(x, y, \lambda)} = f(x, y) - \underline{\lambda g(x, y)}$$

$$\nabla_{x,y,\lambda} \mathcal{L}(x, y, \lambda) = 0 \iff \begin{cases} \nabla_{x,y} f(x, y) = \lambda \nabla_{x,y} g(x, y) \\ g(x, y) = 0 \end{cases}$$

$$\nabla f(\mathbf{x}) = \sum_{k=1}^{M} \lambda_k \nabla g_k(\mathbf{x}) \iff \nabla f(\mathbf{x}) - \sum_{k=1}^{M} \lambda_k \nabla g_k(\mathbf{x}) = 0.$$

- Considering multiple constraints

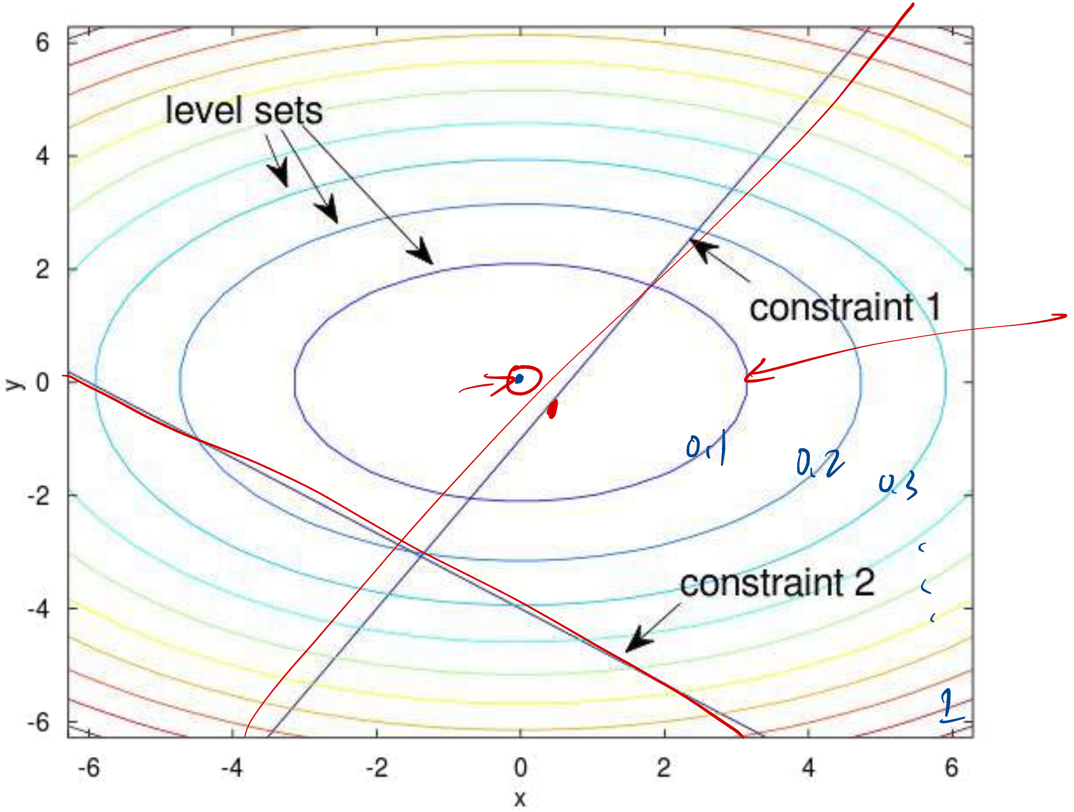
n $\quad M \quad$ λ_k

$$\mathcal{L}(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_M) = f(x_1, \ldots, x_n) - \underline{\sum_{k=1}^{M} \lambda_k g_k(x_1, \ldots, x_n)}$$

$n + M \longrightarrow L$

$$\nabla_{x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_M} \mathcal{L}(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_M) = 0 \iff \begin{cases} \nabla f(\mathbf{x}) - \sum_{k=1}^{M} \lambda_k \nabla g_k(\mathbf{x}) = 0 \\ g_1(\mathbf{x}) = \cdots = g_M(\mathbf{x}) = 0 \end{cases}$$

Constrained Optimization



Lagrange Multiplier

- Example:

$f(x)_{max} = \sqrt{2}$ $f(x)_{min} = -\sqrt{2}$ $x = y = \frac{1}{2\lambda}$

max $$f(x, y) = x + y$$

$$\text{Constraint} : g(x, y) = x^2 + y^2 = 1$$

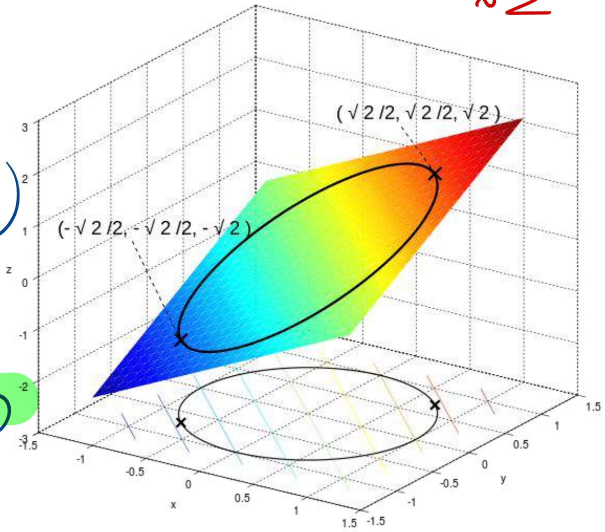
$\lambda = \pm\frac{1}{\sqrt{2}}$

$g(x) = x^2 + y^2 - 1 = 0$

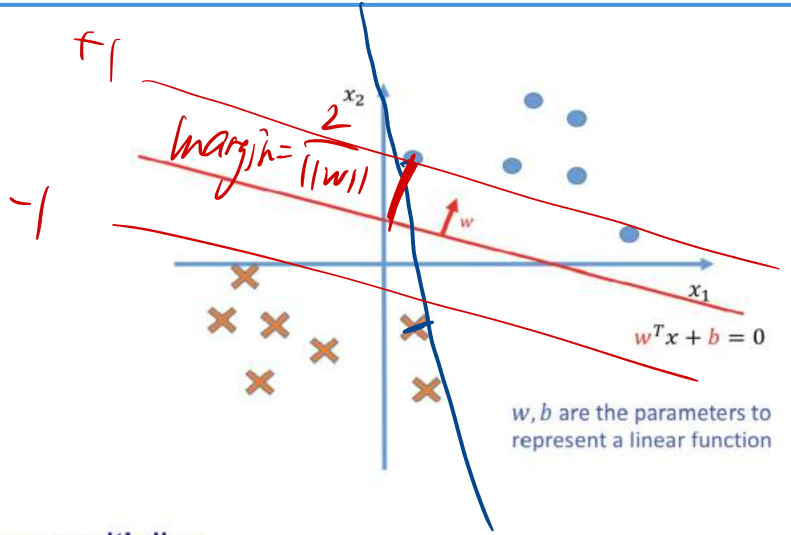
$L(x, y, \lambda) = (x+y) - \lambda(x^2+y^2-1)$

$\frac{\partial L}{\partial x} = 1 - 2\lambda x = 0$ $\frac{\partial L}{\partial y} = 1 - 2\lambda y = 0$

$x^2 + y^2 - 1 = 0$



Lagrange Multiplier: Connect to SVM *



w, b are the parameters to represent a linear function

Original optimization problem:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j$$

Rewrite constraints

One Lagrange multiplier per example

Lagrangian:

$$L(\mathbf{w}, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$
$$\alpha_j \geq 0, \ \forall j$$

What's next?

- In next week's discussion, we will discuss:
    - Logistic Regression (Continued)
    - Naive Bayes, Linear Regression (Planned)

Thank you!

Q & A

UCLA Samueli Computer Science

CS M146 Discussion: Week 4

Logistic Regression & Linear Regression

Junheng Hao

Friday, 01/29/2021

Roadmap

- Announcement
- Logistic Regression
- Linear Regression

Announcements

- **5:00 pm PST, Jan. 29:** Weekly Quiz 4 released on Gradescope.
- **11:59 pm PST, Jan. 31 (Sunday):** Weekly quiz 4 closed on Gradescope!
  - Start the quiz before **11:00 pm PST, Jan. 31** to have the full 60-minute time
- Problem set 1 released on CCLE, submission on Gradescope.
  - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
  - You do not need to submit code, only the results required by the problem set
  - Due on **TODAY 11:59pm PST, Jan. 29 (Friday)**
- Problem set 2 expected to be released on CCLE, submission on Gradescope.
  - Due on two week later, **11:59pm PST, Feb. 12 (Friday)**

About Quiz 4

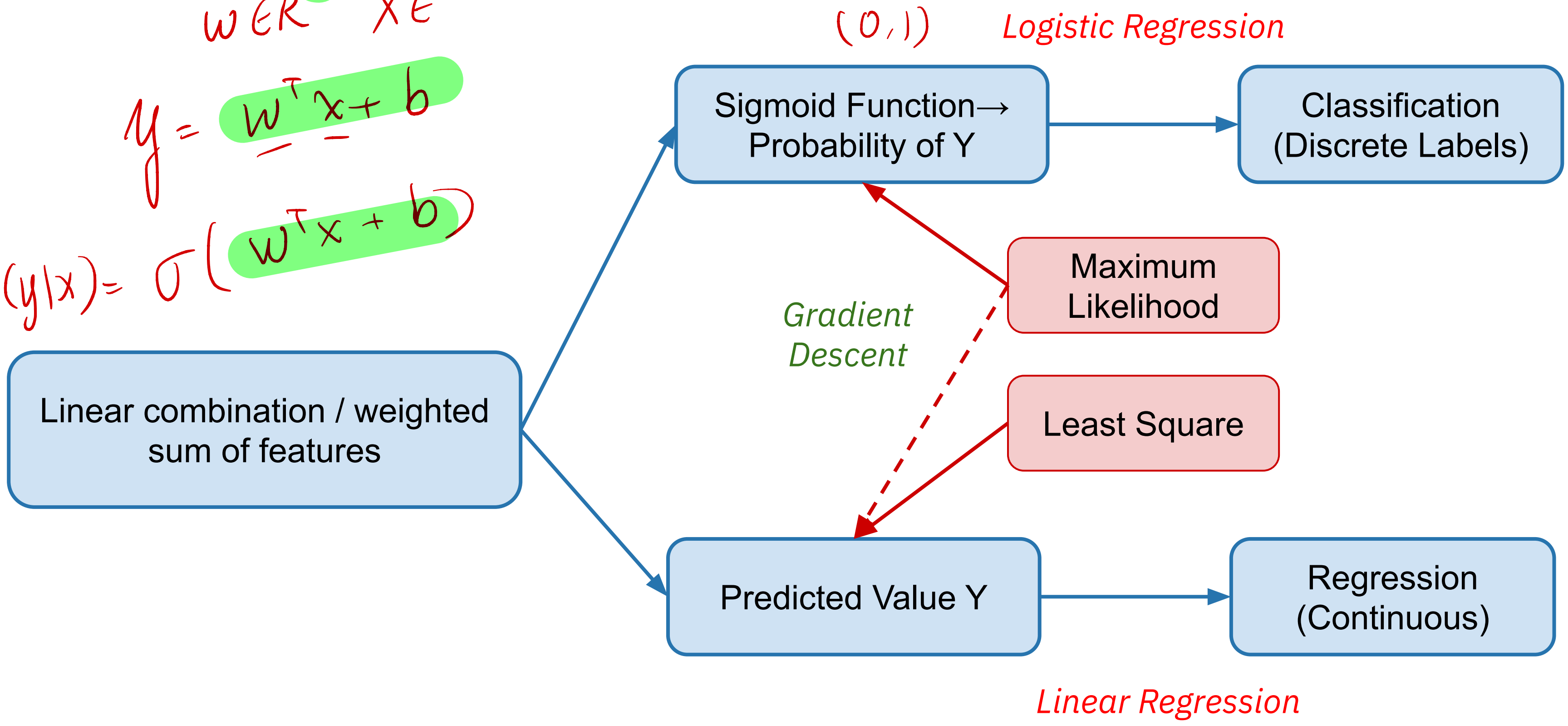
- Quiz release date and time: **Jan 29, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Jan 31, 2021 (Sunday) 11:59 PM PST**
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week 4 Quiz" on GradeScope.
- Topics: **Logistic Regression, Linear Regression, Gradient Descent**
- Question Types
    - True/false, multiple choices
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.

Today's topic

$w \in R^d \quad X \in R^{N\times d}$

$y = w^T x + b$

$P(y|x) = \sigma(w^T x + b)$



Logistic Regression: Example Question

We are given a data set consisting of the following experiment. Well, the dataset is a little bit small. (O_o)

The height and weight of 3 people were recorded at the beginning of each person's 65th birthday. At exactly one year after each person's 65th birthday the vital status was recorded to be either alive or deceased.

Our end goal is to use logistic regression to predict the probability that a person's life expectancy is at least 66 years given their age of 65, initial vital status of alive, height, and weight (but we won't go that far here).

The data is given in the following table on the right.

Height (inches)	Weight (lbs)	Vital Status
60	155	Deceased
64	135	Alive
73	170	Alive

Logistic Regression: Example Question

Step 1: State the log-likelihood function.

Height (inches)	Weight (lbs)	Vital Status
60	155	Deceased
64	135	Alive
73	170	Alive

Logistic Regression: Example Question

Step 1: State the log-likelihood function.

$y = w_1 x_1 + w_2 x_2 + b$

Answer:

$\frac{\partial L}{\partial \theta}$

$\theta = [w_1, w_2, b]^T$

$$\alpha_1 = -b - 155w_1 - 60w_2$$

$$\alpha_2 = -b - 135w_1 - 64w_2$$

$$\alpha_3 = -b - 170w_1 - 73w_2$$

Height (inches)	Weight (lbs)	Vital Status
60	155	Deceased 0
64	135	Alive +1
73	170	Alive +1

$\sigma(\alpha_1)$ $\sigma(\alpha_2)$ $\sigma(\alpha_3)$

$$L = log\left(1 - \frac{1}{1+e^{\alpha_1}}\right) + log\left(\frac{1}{1+e^{\alpha_2}}\right) + log\left(\frac{1}{1+e^{\alpha_3}}\right)$$

$\frac{\partial L}{\partial w_1}$ $\frac{\partial L}{\partial w_2}$ $\frac{\partial L}{\partial b}$

Logistic Regression: Example Question

Step 2: State the gradients for each parameter.

Height (inches)	Weight (lbs)	Vital Status
60	155	Deceased
64	135	Alive
73	170	Alive

Logistic Regression: Example Question

$X^T\beta$ $\quad X^T w = x_1 w_1 + x_2 \cdot w_2 + \cdots + x_n w_n$

$$\frac{\partial L}{\partial b} = \boxed{\frac{\partial L}{\partial \alpha_1} \cdot \frac{\partial \alpha_1}{\partial b}} + \frac{\partial L}{\partial \alpha_2} \cdot \frac{\partial \alpha_2}{\partial b} + \frac{\partial L}{\partial \alpha_3} \cdot \frac{\partial \alpha_3}{\partial b}$$

$\frac{\partial L}{\partial \theta}$ ← learned parameters

Step 2: State the gradients for each parameter.

$P(y=1|x) = \frac{1}{1+e^{-\beta^T x}}$ $\quad \alpha_i$

Answer:

Height (inches)	Weight (lbs)	Vital Status
60	155	Deceased
64	135	Alive
73	170	Alive

1

$$L = -log\left(1 - \frac{1}{1+e^{\alpha_1}}\right) - log\left(\frac{1}{1+e^{\alpha_2}}\right) - log\left(\frac{1}{1+e^{\alpha_3}}\right)$$

positive $\quad$ positive

$\sigma(\alpha_i) = \frac{1}{1+e^{\alpha_i}}$

$$\nabla_b = \boxed{-1.0 \cdot \frac{1}{1+e^{\alpha_1}}} + -1.0 \cdot -\frac{e^{\alpha_2}}{1+e^{\alpha_2}} + -1.0 \cdot -\frac{e^{\alpha_3}}{1+e^{\alpha_3}}$$

$$\nabla_{w_1} = -155.0 \cdot \frac{1}{1+e^{\alpha_1}} + -135.0 \cdot -\frac{e^{\alpha_2}}{1+e^{\alpha_2}} + -170.0 \cdot -\frac{e^{\alpha_3}}{1+e^{\alpha_3}}$$

$$\nabla_{w_2} = -60 \cdot \frac{1}{1+e^{\alpha_1}} + -64.0 \cdot -\frac{e^{\alpha_2}}{1+e^{\alpha_2}} + -73.0 \cdot -\frac{e^{\alpha_3}}{1+e^{\alpha_3}}$$

$x_1 \quad x_2$

$$\alpha_1 = -b - 155w_1 - 60w_2 = -X^T w$$

$$\alpha_2 = -b - 135w_1 - 64w_2$$

$$\alpha_3 = -b - 170w_1 - 73w_2$$

Logistic Regression: Example Question

Step 3: Give the Hessian Matrix (Optional)

$$1-\left(\frac{1}{1+e^{\alpha_1}}\right)(e^{\alpha_1})$$

$$(e^x)' = e^x \qquad \left(\frac{1}{1+x}\right)' = -\frac{1}{(1+x)^2}$$

$$= \frac{1+e^{\alpha_1}-e^{\alpha_1}}{1+e^{\alpha_1}} = \frac{1}{1+e^{\alpha_1}}$$

Height (inches)	Weight (lbs)	Vital Status
60	155	Deceased
64	135	Alive
73	170	Alive

$$\log\left(1-\frac{1}{1+e^{\alpha_1}}\right)$$

$$=\log\left(\frac{e^{\alpha_1}}{1+e^{\alpha_1}}\right) = \log e^{\alpha_1} - \log(1+e^{\alpha_1})$$

$$= \alpha_1 - \log(1+e^{\alpha_1}) \qquad \alpha_1$$

Logistic Regression: Example Question

Step 3: Give the Hessian Matrix

$$\nabla_b = -1.0 \cdot \frac{1}{1+e^{\alpha_1}} + -1.0 \cdot -\frac{e^{\alpha_2}}{1+e^{\alpha_2}} + -1.0 \cdot -\frac{e^{\alpha_3}}{1+e^{\alpha_3}}$$

$$\nabla_{w_1} = -155.0 \cdot \frac{1}{1+e^{\alpha_1}} + -135.0 \cdot -\frac{e^{\alpha_2}}{1+e^{\alpha_2}} + -170.0 \cdot -\frac{e^{\alpha_3}}{1+e^{\alpha_3}}$$

$$\nabla_{w_2} = -60 \cdot \frac{1}{1+e^{\alpha_1}} + -64.0 \cdot -\frac{e^{\alpha_2}}{1+e^{\alpha_2}} + -73.0 \cdot -\frac{e^{\alpha_3}}{1+e^{\alpha_3}}$$

$$\Rightarrow$$

$$H_b^T = \begin{bmatrix} -1.0 \cdot -1.0 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -1.0 \cdot -1.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -1.0 \cdot -1.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \\ -155.0 \cdot -1.0 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -135.0 \cdot -1.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -170.0 \cdot -1.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \\ -60 \cdot -1.0 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -64.0 \cdot -1.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -73.0 \cdot -1.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \end{bmatrix}$$

$$H_{w_1}^T = \begin{bmatrix} -1.0 \cdot -155.0 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -1.0 \cdot -135.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -1.0 \cdot -170.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \\ -155.0 \cdot -155.0 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -135.0 \cdot -135.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -170.0 \cdot -170.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \\ -60 \cdot -155.0 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -64.0 \cdot -135.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -73.0 \cdot -170.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \end{bmatrix}$$

$$H_{w_2}^T = \begin{bmatrix} -1.0 \cdot -60 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -1.0 \cdot -64.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -1.0 \cdot -73.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \\ -155.0 \cdot -60 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -135.0 \cdot -64.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -170.0 \cdot -73.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \\ -60 \cdot -60 \cdot -\frac{e^{\alpha_1}}{(1+e^{\alpha_1})^2} + -64.0 \cdot -64.0 \cdot -\frac{e^{\alpha_2}}{(1+e^{\alpha_2})^2} + -73.0 \cdot -73.0 \cdot -\frac{e^{\alpha_3}}{(1+e^{\alpha_3})^2} \end{bmatrix}$$

Logistic Regression: Example Question

Step 4: Assuming an initial guess of 0.25 for each parameter, write python code for finding the values of the parameters after 2 iterations using the Newton Raphson method.

$$b = 1.1346728128592689$$

$$w_1 = -2.4878423877892759$$

$$w_2 = 3.8192554544178936$$

Height (inches)	Weight (lbs)	Vital Status
60	155	Deceased
64	135	Alive
73	170	Alive

Closed form: LR + Regularization

- Model

$P(y=1|X)$

$$y = \sigma(X) = \frac{1}{1+e^{-X^T\beta}}$$

$-(w_0 \cdot 1 + w_1 x_1 + w_2 x_2 \cdots + w_n x_n)$

- Original Objective

$$J(\beta) = -\frac{1}{n}\sum_i \left(y_i x_i^T \beta - \log\left(1 + \exp\{x_i^T \beta\}\right)\right)$$

- L2-Regularized Objective

$$J(\beta) = -\frac{1}{n}\sum_i \left(y_i x_i^T \beta - \log\left(1 + \exp\{x_i^T \beta\}\right)\right) + \lambda \sum_j \beta_j^2$$

Sigmoid: Calculus Cheatsheet

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

$w^T x + b = 0.5$

where

$$g(z) = \frac{1}{1 + e^{-z}}$$

$g'(0.5) = g(0.5)(1 - g(0.5))$

$$\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1 + e^{-z}} \\
&= \frac{1}{(1 + e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1 + e^{-z})} \cdot \left(1 - \frac{1}{(1 + e^{-z})}\right) \\
&= g(z)(1 - g(z)).
\end{aligned}$$

Logistic Regression: Likelihood

$$
\begin{aligned}
P(y=1 \mid x;\theta) &= h_\theta(x) \\
P(y=0 \mid x;\theta) &= 1 - h_\theta(x)
\end{aligned}
$$

$$p(y \mid x;\theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

Assuming that the m training examples were generated independently, we can then write down the likelihood of the parameters as

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X;\theta) \\
&= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)};\theta) \\
&= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}
\end{aligned}
$$

As before, it will be easier to maximize the log likelihood:

$$
\boxed{
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))
\end{aligned}
}
$$

Logistic Regression: Multiclass Case*

cat [1, 0, 0, 0]
dog [0, 1, 0, 0]

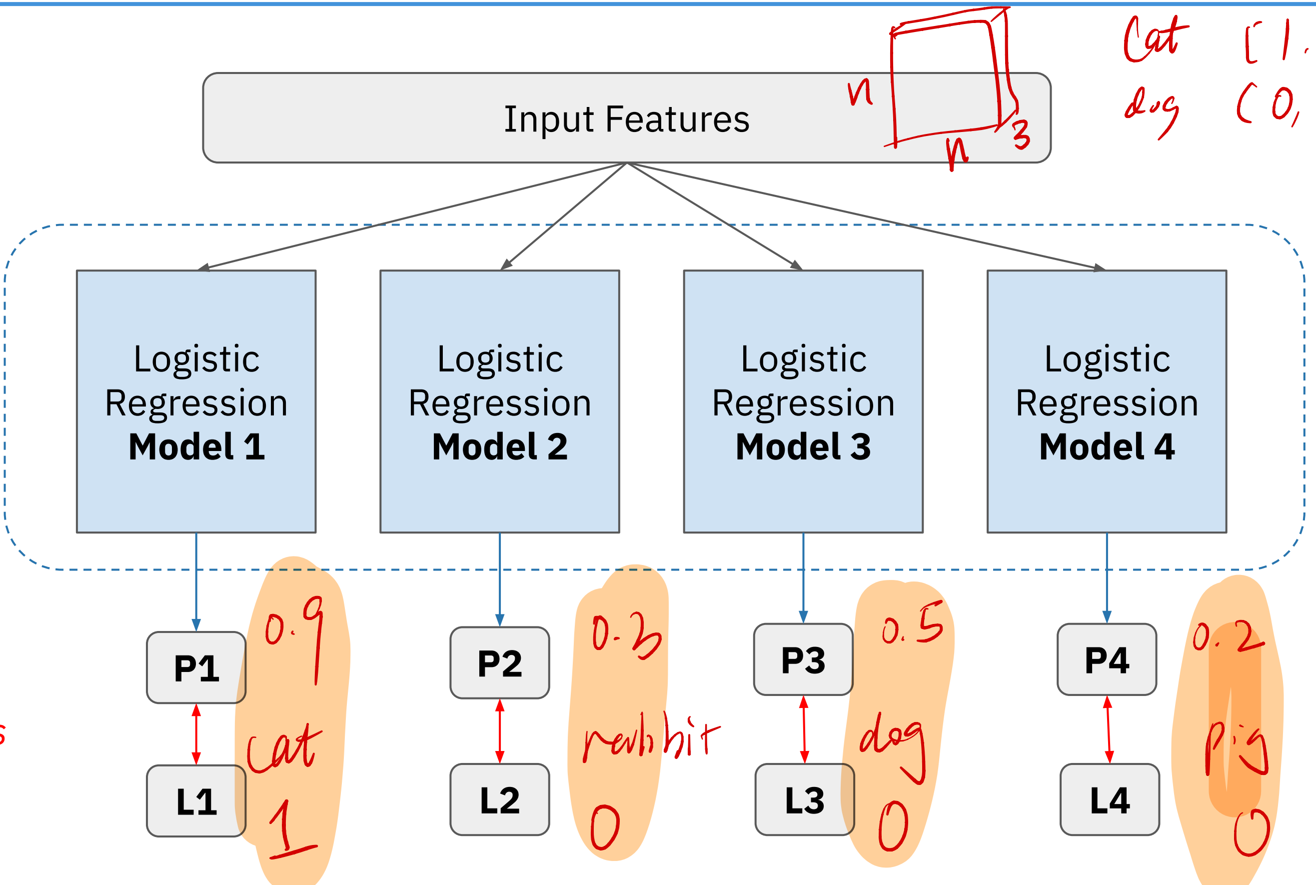
Data

Model Collection

Predictions

Cross-entropy Loss

Label (truth)



Linear Regression: Model

$x^* = (x, 1)^T$

$\beta = (\beta, \beta_0)$

- Linear model to predict value of a variable *y* using features ***x***

$$y = \boldsymbol{x}^T \boldsymbol{\beta} = x_1\beta_1 + x_2\beta_2 + \cdots + x_p\beta_p + \beta_0$$

- Least Square Estimation

$x_i^T \beta = \hat{y}_i \quad y_i$

$$J(\boldsymbol{\beta}) = \frac{1}{2n}(X\boldsymbol{\beta} - y)^T(X\boldsymbol{\beta} - y)$$

$\sum_{i=1}^{n} L_i = (\hat{y}_i - y_i)^2$

- Closed form solution

$\frac{J(\vec{\beta})}{\partial \vec{\beta}} = 0$

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T y$$

$X \in R^{1M \times 1000}$

$\beta^T X^T X \beta \longrightarrow 2X\beta$

Linear Regression: Close-Form

Least Square Estimation $$J(\boldsymbol{\beta}) = \frac{1}{2n}(X\boldsymbol{\beta} - y)^T(X\boldsymbol{\beta} - y)$$

Closed form solution $$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T y$$

Linear Regression: Example

- A ball is rolled down a hallway and its position is recorded at five different times. Use the table shown below to calculate
  - Weights
  - Predicted position at each given time and at time 12 seconds

Time (s)	Position (m)
1	9
2	12
4	17
6	21
8	26

Linear Regression: Example

Step 1: Question

- What are X and Y variables?

- What are the parameters for our problem?

- Calculating parameters

β_0 X Y

Time (s)	Position (m)
1	9
2	12
4	17
6	21
8	26

Linear Regression: Example

Step 1: Calculate Weights

- What are X and Y variables?
    - Time (X) and Position(Y)
- What are the parameters for our problem?
    - $\hat{\beta}_1$:Time $\hat{\beta}_0$:Intercept
      X bias
- Calculating parameters
    - $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T y$

Time (s)	Position (m)
1	9
2	12
4	17
6	21
8	26

Linear Regression: Example

Let's calculate on BOARD!

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

bias

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 4 \\ 1 & 6 \\ 1 & 8 \end{bmatrix} \in R^{5\times 2} \qquad y = \begin{bmatrix} 9 \\ 12 \\ 17 \\ 21 \\ 26 \end{bmatrix} \in R^{5\times 1}$$

$X^{1M \times 1k}$

$(X^T X)^{-1} \in R^{1k \times 1k}$

Time (s)	Position (m)
1	9
2	12
4	17
6	21
8	26

$$X^T X = ?\, R^{2\times 2} \qquad (X^T X)^{-1} = ? \quad R^{2\times 2}$$

$$X^T y = ?\, R^{2\times 1} \quad \hat{\beta} = (X^T X)^{-1} X^T y = ?\, R^{2\times 1}$$

Linear Regression: Example

Step 2: Apply your model and predict

- Plug time values into linear regression equation

$$\hat{y} = 2.378x + 7.012 \quad (\beta_0 \ \beta_1)$$

- Predicted value at time = 12 secs

$$\hat{y}(x = 12) = 2.378 \times 12 + 7.012 = 35.548$$

- Matrix form to predict all other positions

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\beta}$$

Time (s)	Position (m)
1	9
2	12
4	17
6	21
8	26
12	35.55

Linear Regression: Example

Plot: Check your model

$$\hat{y} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 4 \\ 1 & 6 \\ 1 & 8 \end{bmatrix} \begin{bmatrix} 7.012 \\ 2.378 \end{bmatrix} = \begin{bmatrix} 9.390 \\ 11.768 \\ 16.524 \\ 21.280 \\ 26.036 \end{bmatrix}$$

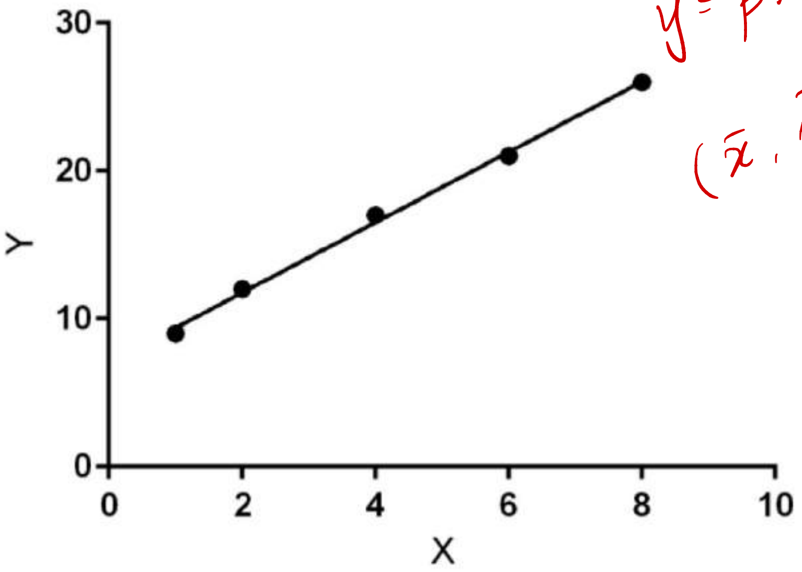
Time (s)	Position (m)
1	9
2	12
4	17
6	21
8	26

Linear Regression: Example

Plot: Check your model

$\hat{y} = \beta^T x$

$(\bar{x}, \bar{y})$



Time (s)	Position (m)
1	9
2	12
4	17
6	21
8	26

Linear Regression: Underfit & Overfit

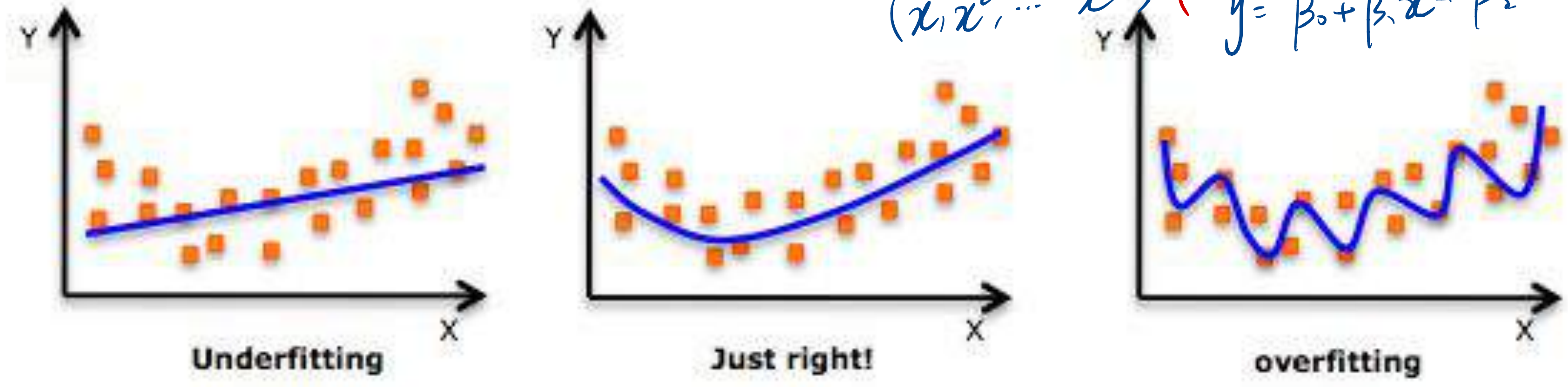
- What is overfitting and underfitting in linear regression? → *This topic will be discussed later.*
  - How to avoid overfitting?

price $y = f(x)$ area

(x, x^2)

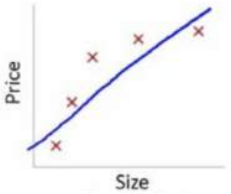
$(x, x^2, \dots x^9)$

$$\begin{cases} y = \beta_0 + \beta_1 x \Leftarrow ① \\ y = \beta_0 + \beta_1 x + \beta_2 x^2 \Leftarrow ② \\ y = \beta_0 + \beta_1 x + \beta_2 x^2 \dots + \beta_9 x^9 \quad ③ \Leftarrow \end{cases}$$



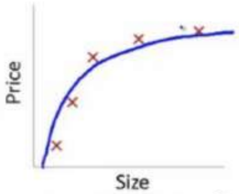
Bias vs Variance: Example

Linear Regression



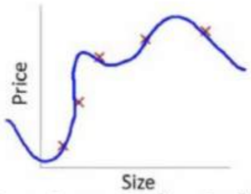
$\theta_0 + \theta_1 x$

High bias (underfit)



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

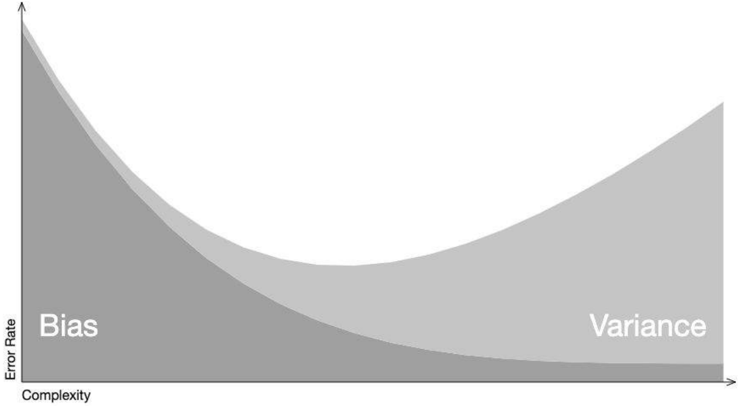
High variance (overfit)

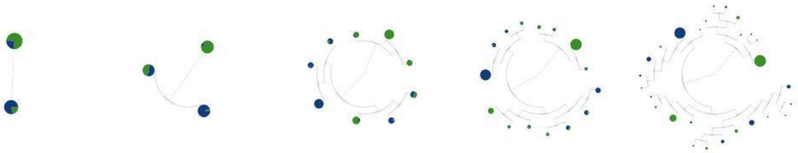
(x_1, x_2)

$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + x_1 \cdot x_2 \cdot \beta_3 + \beta_4 x_1^2 x_2$

$+ \beta_5 x_1 x_2^2 + \ldots$

Decision Tree (Week 2)



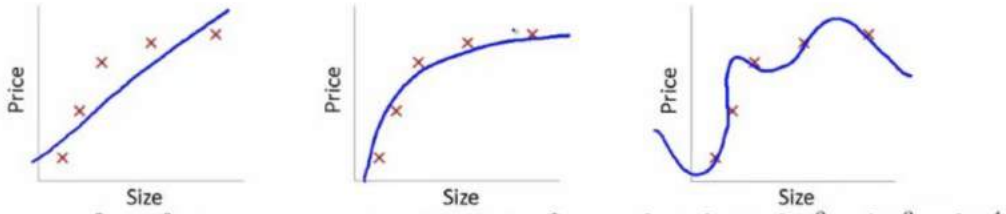


Bias vs Variance: Example

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$

$\vdots$

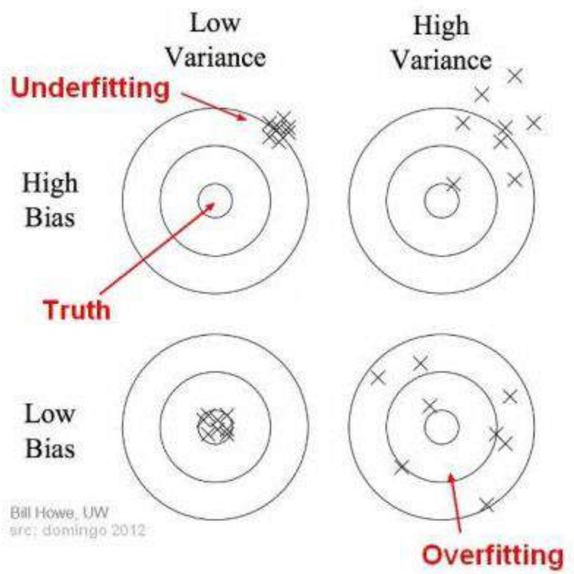
10. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{10} x^{10}$



$\theta_0 + \theta_1 x$ — High bias (underfit)

$\theta_0 + \theta_1 x + \theta_2 x^2$ — "Just right"

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ — High variance (overfit)



Closed form: LR + Regularization

- Model

$$\hat{y} = \boldsymbol{x}^T \boldsymbol{\beta} = x_1\beta_1 + x_2\beta_2 + \cdots + x_p\beta_p$$

- Original Objective

$$\min_{\boldsymbol{\beta}} \; J(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}^T\boldsymbol{\beta} - y)^2$$

- L2-Regularized Objective

$$\min_{\boldsymbol{\beta}} \; J(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}^T\boldsymbol{\beta} - y)^2 + \frac{\lambda}{2}||\boldsymbol{\beta}||^2$$

$$\frac{\partial J(\beta)}{\partial \beta} = \lambda\beta$$

$$\partial\left(\frac{\lambda}{2}\beta^T\beta\right)/\partial\beta$$

Closed form: LR + Regularization

$$\boxed{\min_{\boldsymbol{\beta}} \; J(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}^T\boldsymbol{\beta} - y)^2 + \frac{\lambda}{2}||\boldsymbol{\beta}||^2}$$

$$\frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \boldsymbol{x}(\boldsymbol{x}^T\boldsymbol{\beta} - y) + \frac{\partial \lambda \; ||\boldsymbol{\beta}||^2}{\partial \boldsymbol{\beta}}$$

$$\frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N} \boldsymbol{x}(\boldsymbol{x}^T\boldsymbol{\beta} - y) + \lambda\boldsymbol{\beta} \quad = 0$$

Linear Regression: Probabilistic Interpretation

Likelihood of one training sample (x_n, y_n)

$$p(y_n|x_n;\boldsymbol{\theta}) = \mathcal{N}(\theta_0 + \theta_1 x_n, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[y_n-(\theta_0+\theta_1 x_n)]^2}{2\sigma^2}}$$

$$\begin{aligned}
\mathcal{LL}(\boldsymbol{\theta}) &= \log P(\mathcal{D}) \\
&= \log \prod_{n=1}^{\mathrm{N}} p(y_n|x_n) = \sum_n \log p(y_n|x_n) \\
&= \sum_n \left\{ -\frac{[y_n - (\theta_0 + \theta_1 x_n)]^2}{2\sigma^2} - \log \sqrt{2\pi}\sigma \right\} \\
&= -\frac{1}{2\sigma^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 - \frac{\mathrm{N}}{2} \log \sigma^2 - \mathrm{N} \log \sqrt{2\pi} \\
&= -\frac{1}{2} \left\{ \frac{1}{\sigma^2} \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2 + \mathrm{N} \log \sigma^2 \right\} + \text{const}
\end{aligned}$$

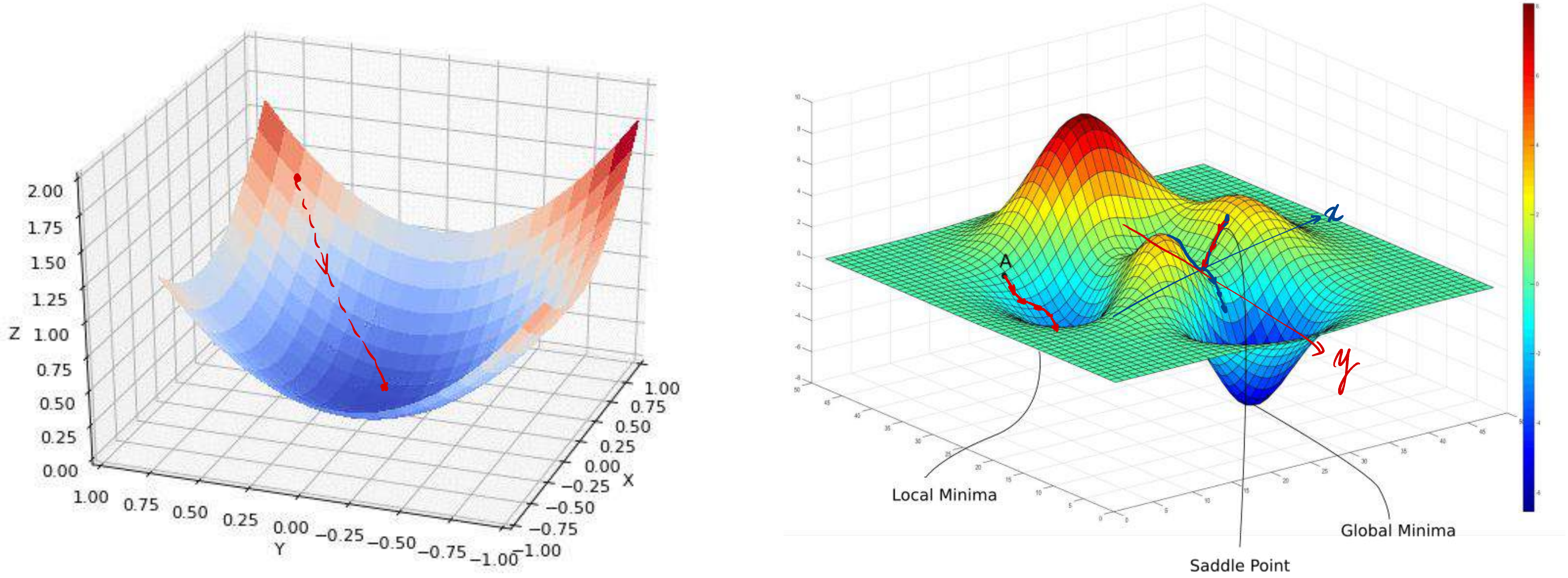
Maximize over θ_0 and θ_1

$$\max \ \log P(\mathcal{D}) \Leftrightarrow \min \sum_n [y_n - (\theta_0 + \theta_1 x_n)]^2$$

MLE = Least Square Error!

Gradient Descent

$\max(0, -y_n w x_n)$



Batch vs Stochastic Gradient Descent

Algorithm 1 Gradient descent

1: $\boldsymbol{\theta} \leftarrow \mathbf{0}$.
2: **for** $epoch = 1 \ldots T$ **do**
3: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla J(\boldsymbol{\theta})$
4: **end for**
5: **return** $\boldsymbol{\theta}$

$\frac{\partial J(\theta)}{\partial \theta}$

$J(\theta) = \sum_{i=1}^{n} (y_i - \hat{y})^2$

$\theta^T x$

Linear Regression

Algorithm 2 Stochastic Gradient descent

1: $\boldsymbol{\theta} \leftarrow \mathbf{0}$.
2: **for** $epoch = 1 \ldots T$ **do**
3: $\quad$ **for** $(\boldsymbol{x}, y) \in \mathcal{D}$ **do**
4: $\quad\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla J_{(\boldsymbol{x},y)}(\boldsymbol{\theta})$
5: $\quad$ **end for**
6: **end for**
7: **return** $\boldsymbol{\theta}$

Randomly choosing a training sample

Algorithm 2 Gradient Descent (J)

1: $t \leftarrow 0$
2: Initialize $\theta^{(0)}$
3: **repeat**
4: $\quad \nabla J(\boldsymbol{\theta}^{(t)}) = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta}^{(t)} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = \sum_n (\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta}^{(t)} - y_n)\boldsymbol{x}_n$
5: $\quad \boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \nabla J(\boldsymbol{\theta}^{(t)})$
6: $\quad t \leftarrow t + 1$
7: **until** convergence
8: Return final value of $\boldsymbol{\theta}$

$\sum_i (x_i^T \theta^{(t)} - y_i) x_i$

Algorithm 3 Stochastic Gradient Descent (J)

1: $t \leftarrow 0$
2: Initialize $\boldsymbol{\theta}^{(0)}$
3: **repeat**
4: $\quad$ Randomly choose a training a sample $\boldsymbol{x}_t$
5: $\quad$ Compute its contribution to the gradient $\boldsymbol{g}_t = (\boldsymbol{x}_t^{\mathrm{T}}\boldsymbol{\theta}^{(t)} - y_t)\boldsymbol{x}_t$
6: $\quad \boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \boldsymbol{g}_t$
7: $\quad t \leftarrow t + 1$
8: **until** convergence
9: Return final value of $\boldsymbol{\theta}$

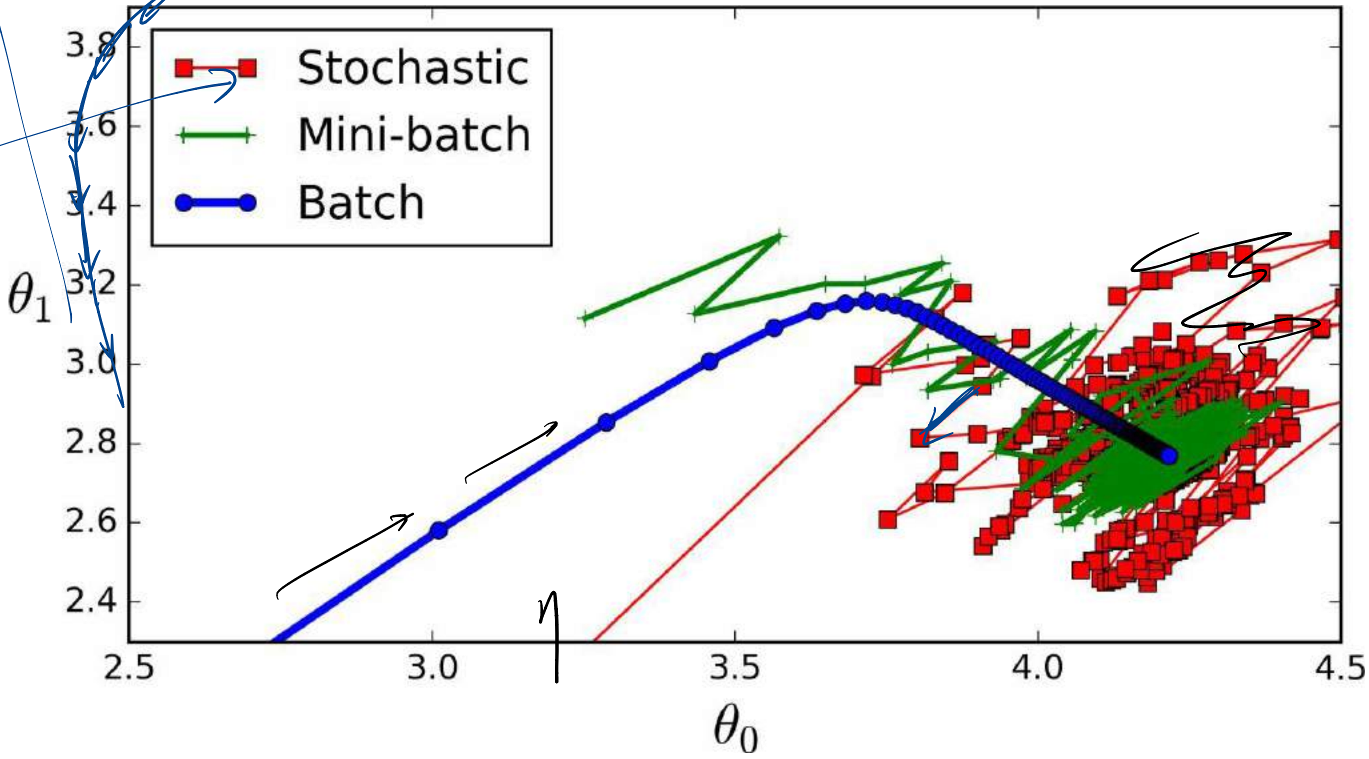
$J(\theta)$

$(x_i^T \theta^{(t)} - y_i) x_i$

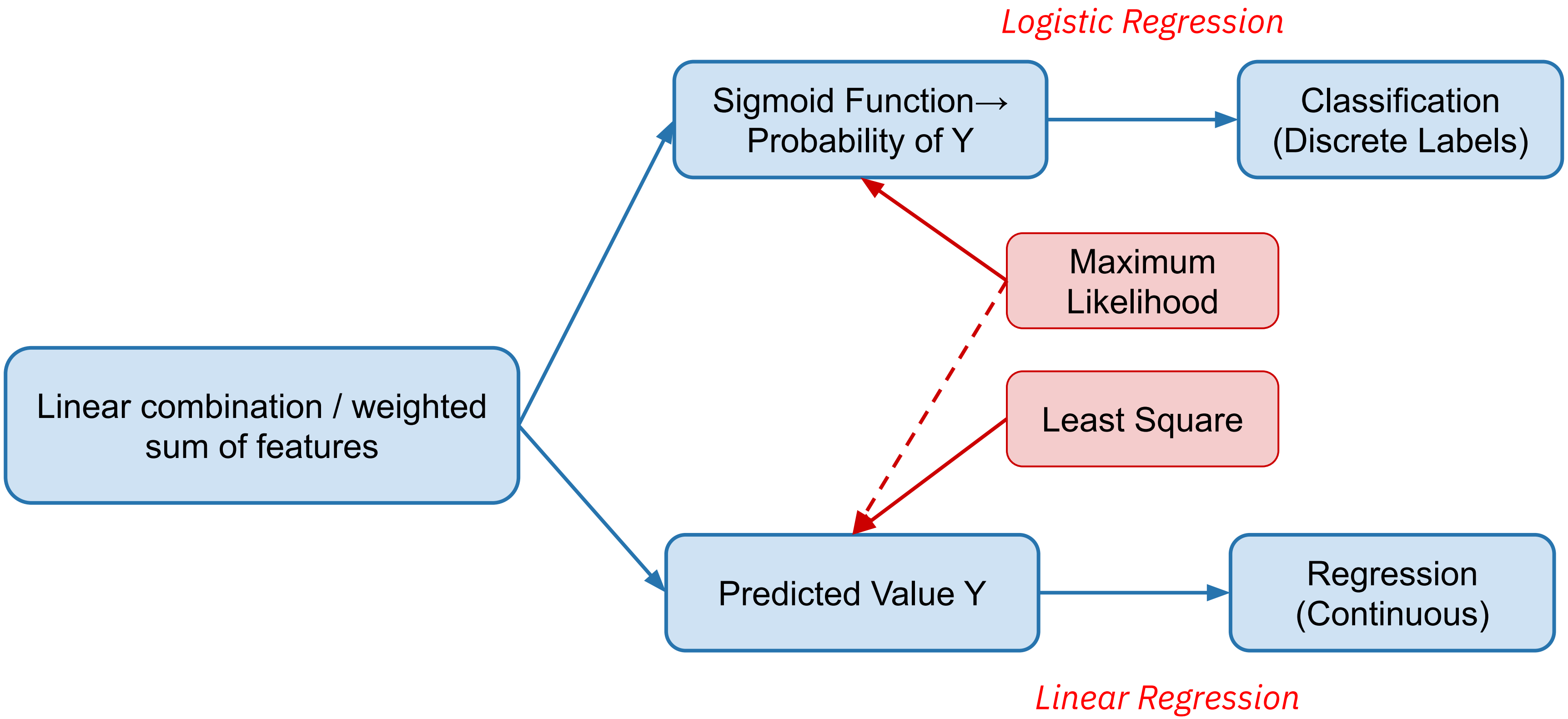
Batch / Stochastic / Mini-Batch

$y = -x^2$

$\log(x)$



Logistic Regression VS Linear Regression



* Logistic Regression: Convergence?

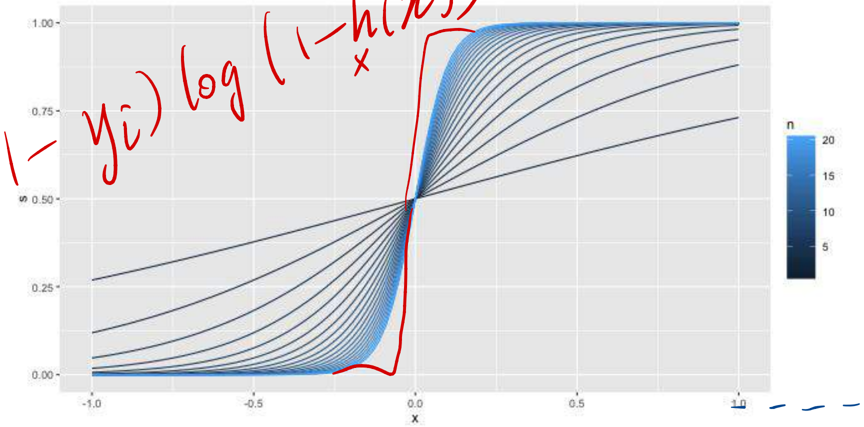
$-1 \quad -$

$+1 \quad +$

- True/False: Logistic regression cannot converge on a linearly separable dataset.

w, b

$$L(\theta) = \sum_{i=1}^{4} y_i \log(h(x)) + (1 - y_i)\log(1 - h(x))$$

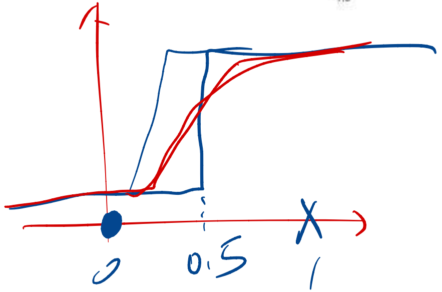


$w = 1$

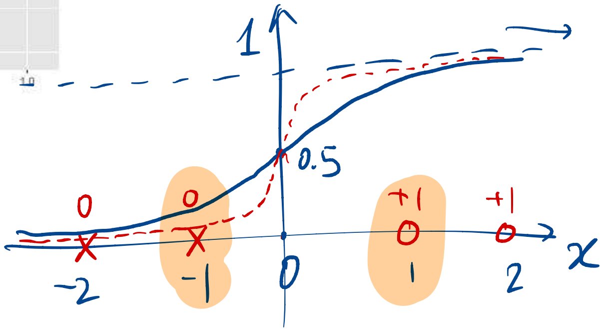
$w = 2$

$w = \infty$

$\sigma(wx)$



$y = \sigma(wx)$



UCLA Samueli Computer Science

$$J(\theta) = (\theta x_1 - \hat{y}_1)^2 + (\theta x_2 - \hat{y}_2)^2$$

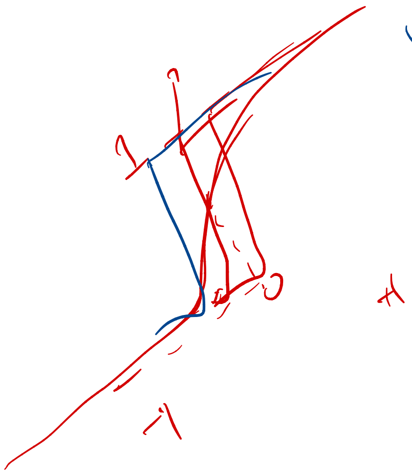
$$\frac{\partial J(\theta)}{\partial \theta} = 2(\theta x_1 - y_1)\cdot x_1 + 2(\theta x_2 - y_2)\cdot x_2$$

$$\sum_{i=1}^{2} (\theta x_i - y_i)\cdot x_i$$

$w. \to \text{inf}$

$wx + b = 0$

$b = 0$

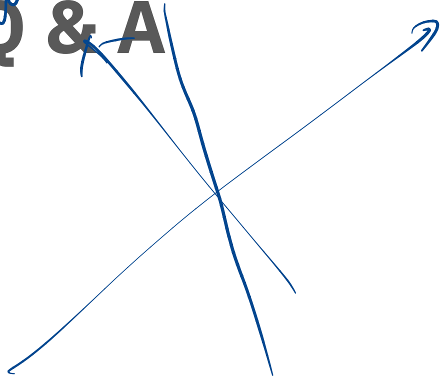


Thank you!

Q & A

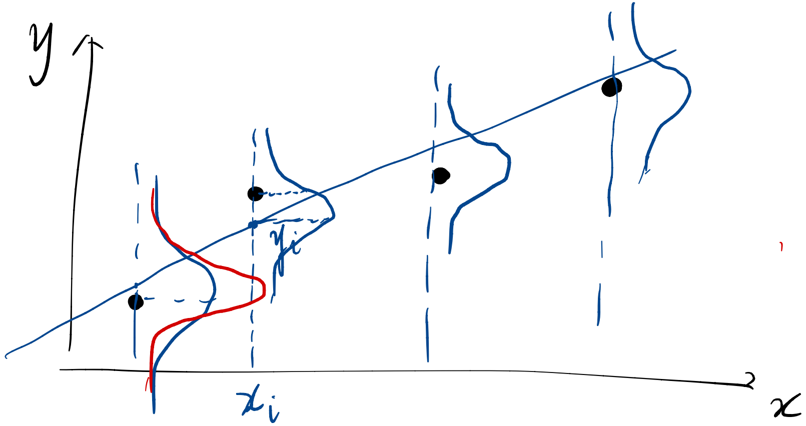
wx

w



UCLA Samueli Computer Science

$y = \theta_0 + \theta_1 x$



Thank you!

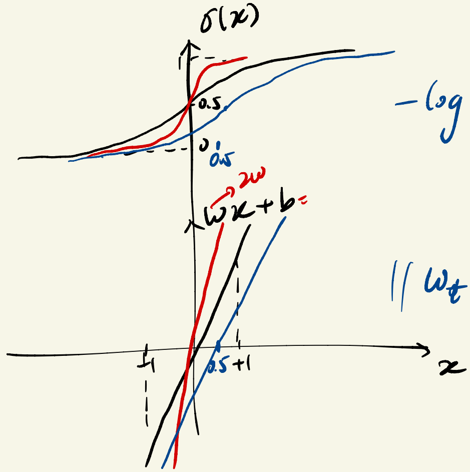
$x_i \theta = y_i$

Q & A

$\bar{\mu}_i = x_i \theta_1 + \theta_0$

$\sigma_i^2 = \sigma^2$

$\sigma^2 (x$



$-\log L$

0.05

$\|w_t - w_{t-1}\|$

UCLA Samueli Computer Science

CS M146 Discussion: Week 5

Overfitting and Regularization, Neural Nets

Junheng Hao

Friday, 02/05/2021

Roadmap

- Announcement
- Overfitting and Regularization
- Neural Nets

Announcements

- **5:00 pm PST, Feb 5 (Friday):** Weekly Quiz 5 released on Gradescope.
- **11:59 pm PST, Feb 7 (Sunday):** Weekly quiz 5 closed on Gradescope!
  - Start the quiz before **11:00 pm PST, Feb 7** to have the full 60-minute time
- **Problem set 2** released on CCLE, submission on Gradescope.
  - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
  - You do not need to submit code, only the results required by the problem set
  - Due on **11:59pm PST, Feb 12 (Friday)**

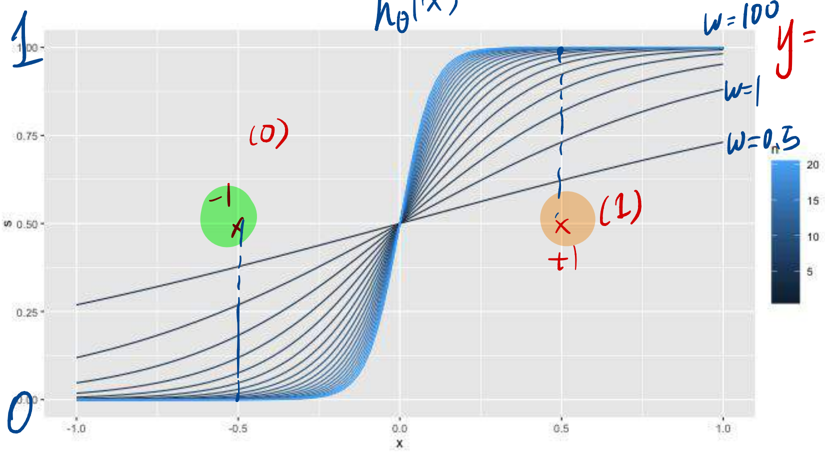
About Quiz 5

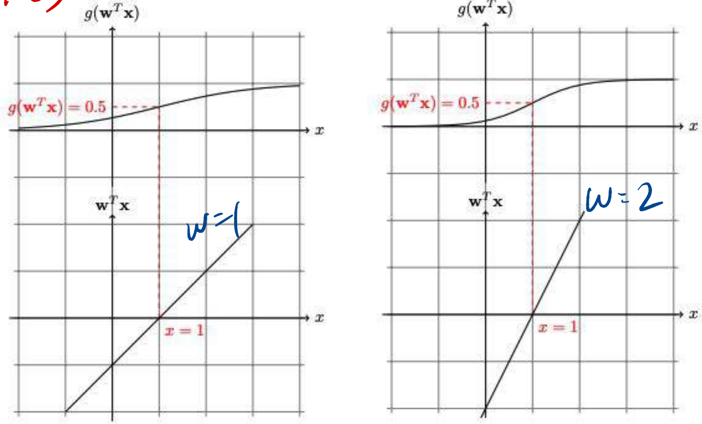
- Quiz release date and time: **Feb 5, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Feb 7, 2021 (Sunday) 11:59 PM PST**
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week 4 Quiz" on GradeScope.
- Topics: **Overfitting, Regularization, Neural Nets (without Backprop)**
- Question Types
  - True/false, multiple choices
  - Some questions may include several subquestions.
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.

Clarification: Logistic Regression Convergence

Logistic regression (without regularization) **cannot** converge on a linearly separable dataset.

$y = \sigma(wx + b)$ $\quad b = -0.02$





$100x$

Campuswire Post: https://campuswire.com/c/GB5E561C3/feed/230
Reference: https://www.cscu.cornell.edu/news/statnews/82_lgsbias.pdf

Clarification: Logistic Regression Convergence

```
X, y = np.array([[-1],[1]]), np.array([0,1]) # train data
```

In sklearn, you have solver options as `newton-cg, lbfgs, liblinear, sag, saga`.

Then train a logistic regression model **without** penalty

```
clf = LogisticRegression(random_state=0, penalty='none', solver='sag', max_iter=1000).fit(X, y) # or other solver except 'liblinear'
```

You may notice different solvers may result in different w ranging from 5 to 10 (printed by `clf.coef_`). Sometimes you might have a convergence error as follows:

```
clf=None
# solver = 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'
clf = LogisticRegression(random_state=12, penalty='none', solver='sag', max_iter=1000, verbose=10).fit(X, y)
clf.coef_, clf.intercept_, clf.n_iter_
```

```
max_iter reached after 0 seconds
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
/Users/junhenghao/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/_sag.py:330: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  "the coef_ did not converge", ConvergenceWarning)
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s finished
```

```
(array([[8.29936548]]), array([0.00109003]), array([1000], dtype=int32))
```

And again train a logistic regression model **with** L2 penalty

```
clf = LogisticRegression(random_state=0, penalty='l2', solver='lbfgs').fit(X, y) # L2 used
```

This time, you will find different solvers converges to $w = 0.675$.

Colab Link: https://colab.research.google.com/drive/1HrmthtXmg2PQ_9BHry1zrePWnSs2iQLn?usp=sharing

Overfitting

Key Questions:

- How to identify overfitting?
- How to avoid overfitting?

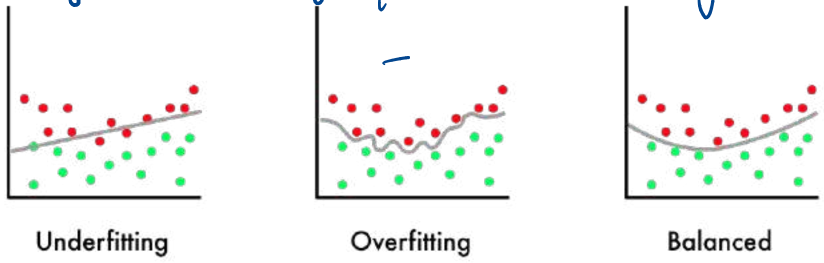
Credit:
https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

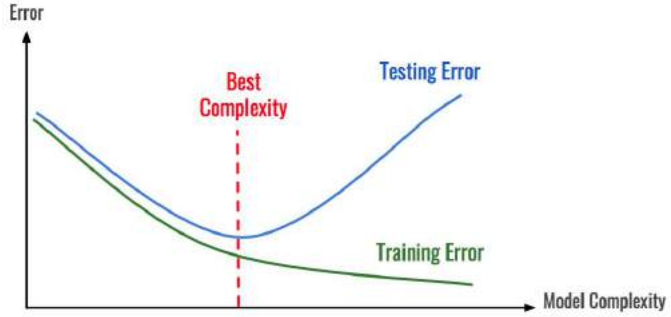
	Low *Training* Error	High *Training* Error
Low *Testing* Error	The model is learning!	Probably some error in your code. Or you've created a *psychic* AI.
High *Testing* Error	OVERFITTING	The model is not learning.

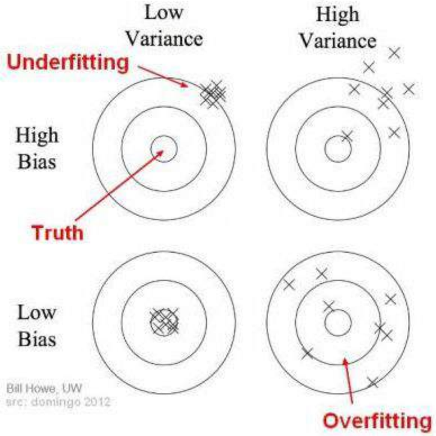
Overfitting: Polynomial Regression

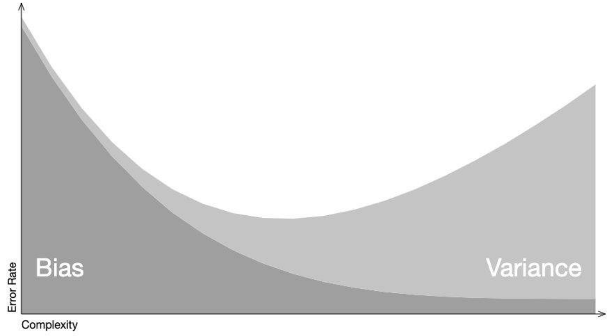
$+(w_1^2)$ $+(\sum w_i^2)$

$y = w_1 x$ $y = \sum_{i=1}^{10} w_i x^i$ $y = w_1 x + w_2 x^2$









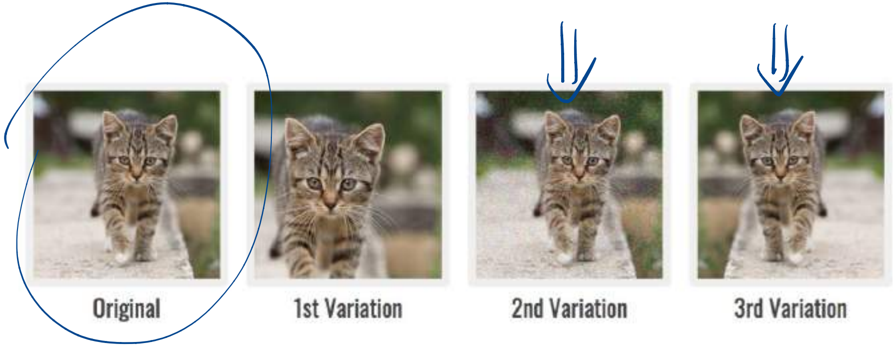
Overfitting

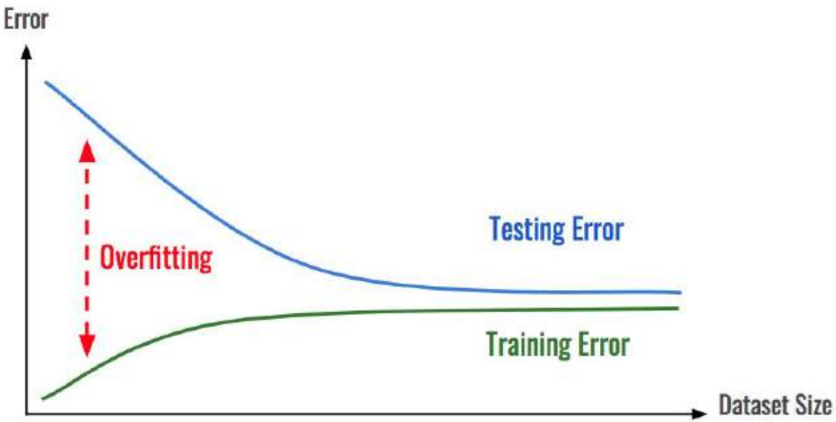
Overfitting

Data Size | Model Setting | **Regularization**

Overfitting: Data solution

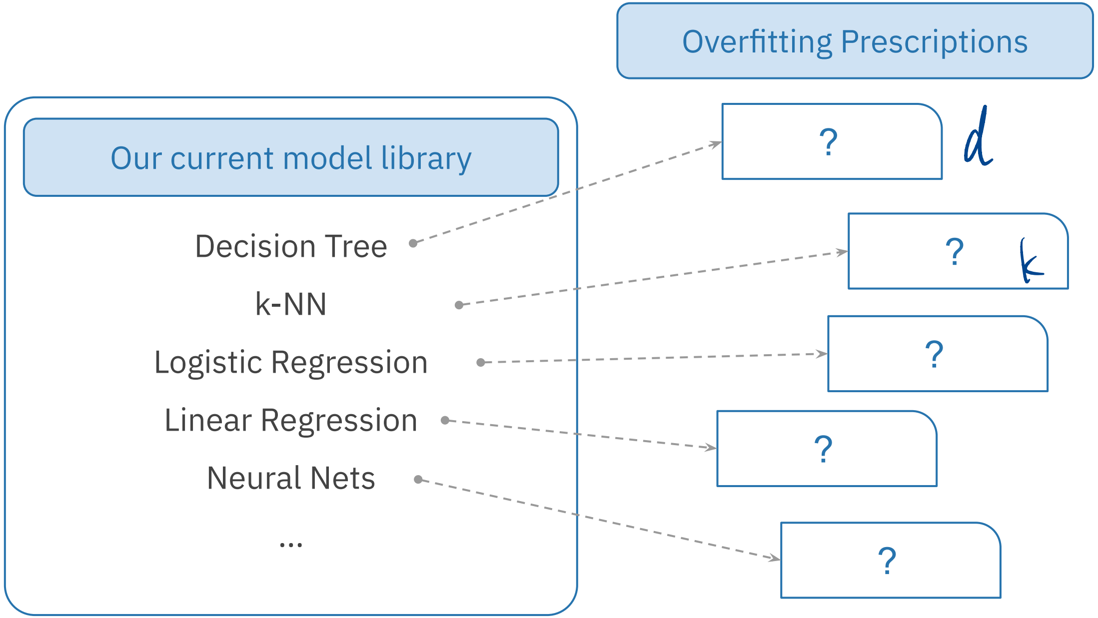
- Collecting more data
- Data augmentation

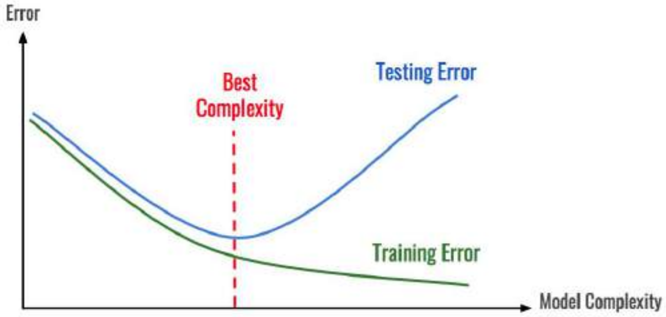




Overfitting: Model Solution

- Avoid overfitting by changing model hyperparameter selection, from the mechanism and inductive bias of the model.





Regularization

Linear Regression

- Model

$$\hat{y} = \boldsymbol{x}^T \boldsymbol{\beta} = x_1\beta_1 + x_2\beta_2 + \cdots + x_p\beta_p$$

- Original Objective

$$\min_{\boldsymbol{\beta}} \ J(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}^T\boldsymbol{\beta} - y)^2$$

- L2-Regularized Objective

m

$$\min_{\boldsymbol{\beta}} \ J(\boldsymbol{\beta}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}^T\boldsymbol{\beta} - y)^2 + \boxed{\frac{\lambda}{2}||\boldsymbol{\beta}||_2^2}$$

Logistic Regression

- Model

$$y = \sigma(X) = \frac{1}{1+e^{-X^T\beta}}$$

- Original Objective

$$J(\beta) = -\frac{1}{n}\sum_i \left(y_i x_i^T\beta - \log\left(1+\exp\{x_i^T\beta\}\right)\right)$$

- L2-Regularized Objective

$$J(\beta) = -\frac{1}{n}\sum_i \left(y_i x_i^T\beta - \log\left(1+\exp\{x_i^T\beta\}\right)\right) + \boxed{\lambda\sum_j \beta_j^2}$$

UCLA Engineer Change.

Linear Regression + Regularization

$f(\vec{\beta})$ $\frac{\partial f(\vec{\beta})}{\partial \vec{\beta}}$

$$\min_{\beta} \; J(\beta) = \frac{1}{2}\sum_{i=1}^{N}(x^T\beta - y)^2 + \frac{\lambda}{2}||\beta||^2$$

$$\frac{\partial J(\beta)}{\partial \beta} = \sum_{i=1}^{N} x(x^T\beta - y) + \frac{\partial \lambda\, ||\beta||^2}{\partial \beta}$$

$$\frac{\partial J(\beta)}{\partial \beta} = \sum_{i=1}^{N} x(x^T\beta - y) + \lambda\beta$$

$f(x) = Ax$

$\frac{\partial f(x)}{\partial x}$

$\frac{\partial f(x)}{\partial A}$

$$\boxed{\nabla_{\beta} J(\beta) = 0}$$

$\frac{\partial J(\beta)}{\partial \beta_i}$

i $[\beta_0, \beta_1, \beta_i, \beta_n]$

$$\nabla_{\beta}\left(\frac{1}{2}||X\beta - y||^2 + \frac{\lambda}{2}||\beta||^2\right) = 0$$

$$\nabla_{\beta}\left(\frac{1}{2}(X\beta - y)^T(X\beta - y) + \frac{\lambda}{2}\beta^T\beta\right) = 0$$

$$X^TX\beta - X^Ty + \lambda I\beta = 0$$

$$\boxed{\beta = (X^TX + \lambda I)^{-1}X^Ty}$$

About Norms (Vectors)

$$||\mathbf{w}||_1 = |w_1| + |w_2| + \ldots + |w_N|$$

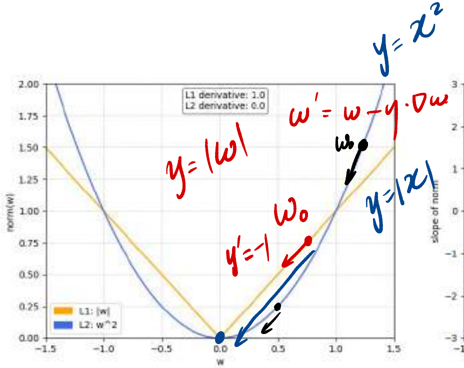
1-norm (also known as L1 norm)

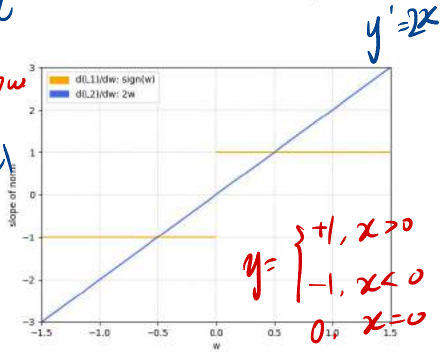
$$\|\mathbf{w}\|_2 = \left(|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2\right)^{\frac{1}{2}}$$

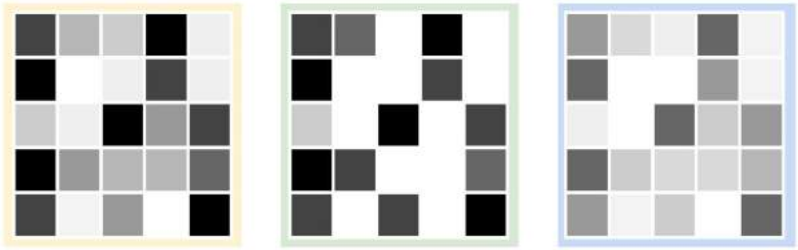
2-norm (also known as L2 norm or Euclidean norm)

$$\|\mathbf{w}\|_p = \left(|w_1|^p + |w_2|^p + \ldots + |w_N|^p\right)^{\frac{1}{p}}$$

p-norm







Baseline L1 Regularization L2 Regularization

About Norms (Vectors)

$$||\mathbf{w}||_1 = |w_1| + |w_2| + ... + |w_N|$$

1-norm (also known as L1 norm)

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

Loss function with L1 regularisation

$$||\mathbf{w}||_2 = \left(|w_1|^2 + |w_2|^2 + \ldots + |w_N|^2\right)^{\frac{1}{2}}$$

2-norm (also known as L2 norm or Euclidean norm)

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

Loss function with L2 regularisation

How does L1/L2 regularization change the gradient descent step?

About "Entrywise" Norms (Matrix)

- L(2,1) Norm and L(p,q) Norm

$$\|A\|_{2,1} = \sum_{j=1}^{n} \|a_j\|_2 = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} |a_{ij}|^2 \right)^{\frac{1}{2}} \Rightarrow \|A\|_{p,q} = \left(\sum_{j=1}^{n} \left(\sum_{i=1}^{m} |a_{ij}|^p \right)^{\frac{q}{p}} \right)^{\frac{1}{q}}$$

- Frobenius norm (Hilbert–Schmidt norm) $p=2 \quad q=2$

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2$$

$$\|W\|_F^2$$

- Max Norm

$$\|A\|_{\max} = \max_{ij} |a_{ij}|.$$

Credit: https://en.wikipedia.org/wiki/Matrix_norm

Effect on Regularization

Without L2 Regularization

$(-1, 0) \quad (-1, 1)$
$x_1 \; y_2 \quad x_2 \; y_2$

```
clf = LogisticRegression(random_state=0, penalty='none', solver='lbfgs', max_iter=100).fit(X, y) # or other solver except 'liblinear'
print(clf.coef_, clf.intercept_, clf.n_iter_)
```

```
[[9.91926856]] [0.] [13]
```

```
clf = LogisticRegression(random_state=0, penalty='none', solver='newton-cg', max_iter=1000).fit(X, y) # or other solver except 'liblinear'
print(clf.coef_, clf.intercept_, clf.n_iter_)
```

```
[[10.20283614]] [0.] [9]
```

With L2 Regularization

λ

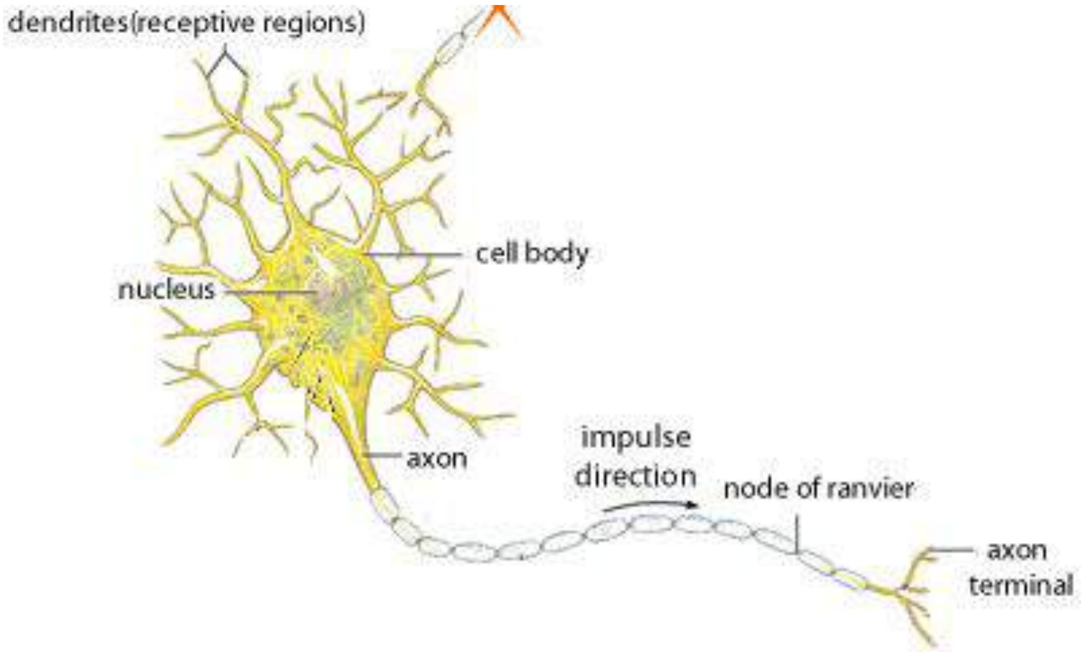
```
clf = LogisticRegression(random_state=0, penalty='l2', solver='lbfgs').fit(X, y)
print(clf.coef_, clf.intercept_, clf.n_iter_)
```

```
[[0.67483169]] [0.] [4]
```

```
clf = LogisticRegression(random_state=0, penalty='l2', solver='newton-cg').fit(X, y)
print(clf.coef_, clf.intercept_, clf.n_iter_)
```

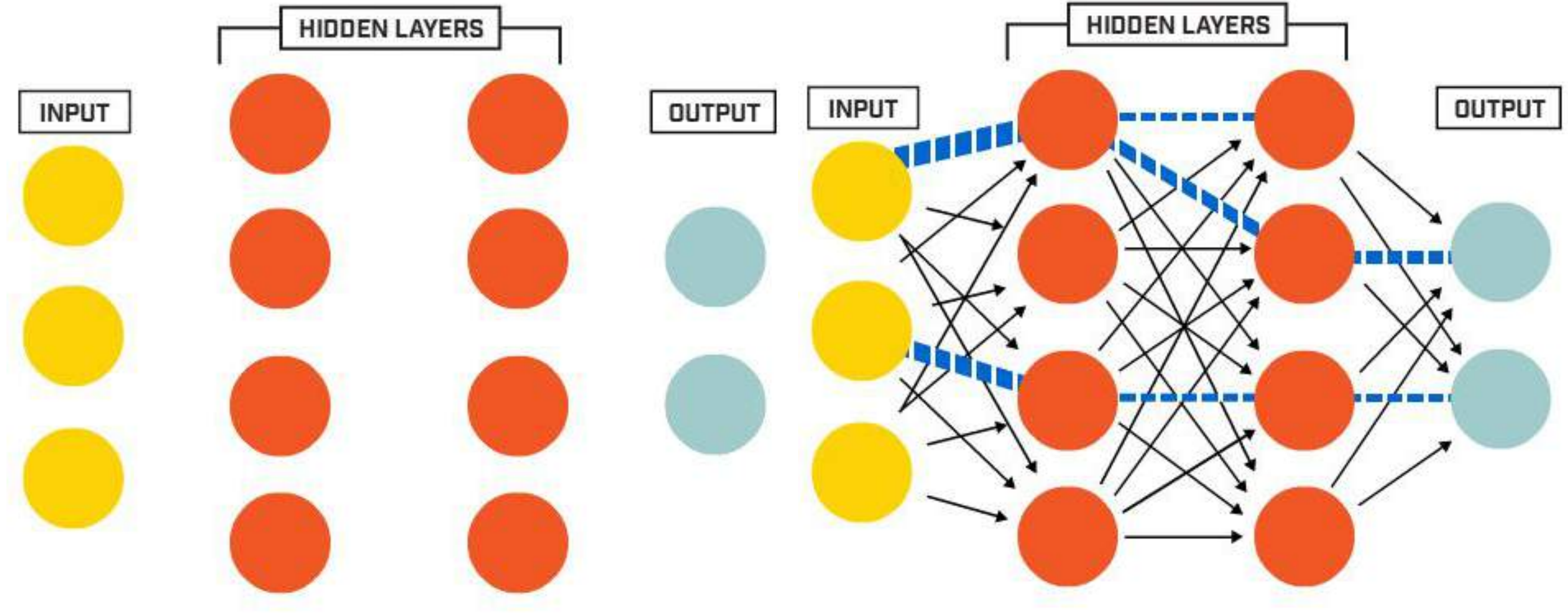
```
[[0.67482829]] [0.] [2]
```

Neural Networks: Neuron/Perceptron



https://medium.com/typeme/lets-code-a-neural-network-from-scratch-part-1-24f0a30d7d62
https://becominghuman.ai/what-is-an-artificial-neuron-8b2e421ce42e

Neural Networks: A Simple Architecture



https://www.ptgrey.com/deep-learning

NN Example: XOR

- Which NN architecture corresponds to which function?

Y	1	**0**	**1**
	0	**0**	**0**
		0	1
		X	

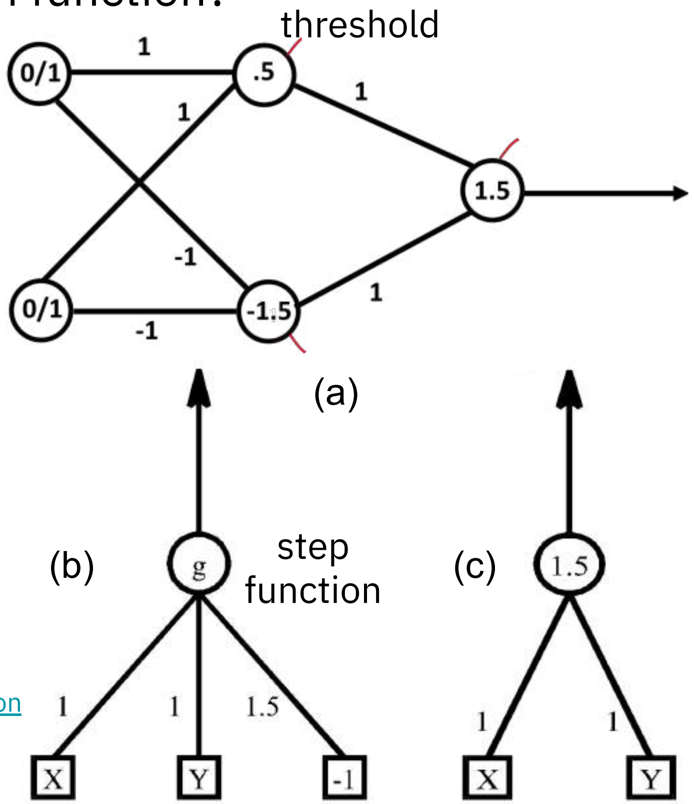
Table 1: Truth table for AND

Y	1	**1**	**1**
	0	**0**	**1**
		0	1
		X	

Table 2: Truth table for OR

Y	1	**1**	**0**
	0	**0**	**1**
		0	1
		X	

Table 3: Truth Table for XOR



https://datascience.stackexchange.com/questions/11589/creating-neural-net-for-xor-function
http://yen.cs.stir.ac.uk/~kjt/techreps/pdf/TR148.pdf
https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b

NN Example: XOR

		X=0	X=1
Y	1	0	1
	0	0	0

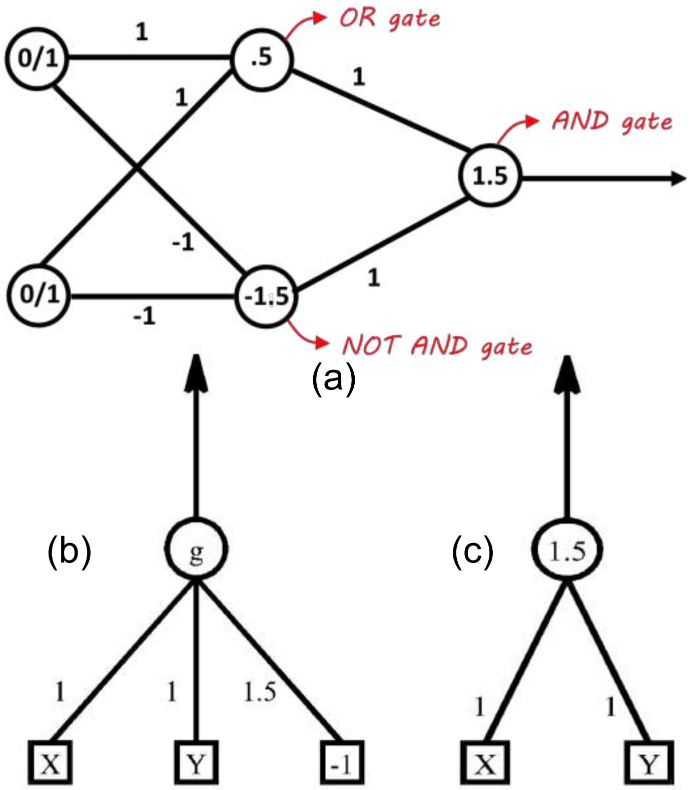
Table 1: Truth table for AND

		X=0	X=1
Y	1	1	1
	0	0	1

Table 2: Truth table for OR

		X=0	X=1
Y	1	1	0
	0	0	1

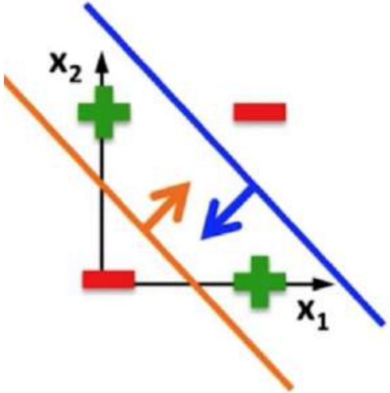
Table 3: Truth Table for XOR

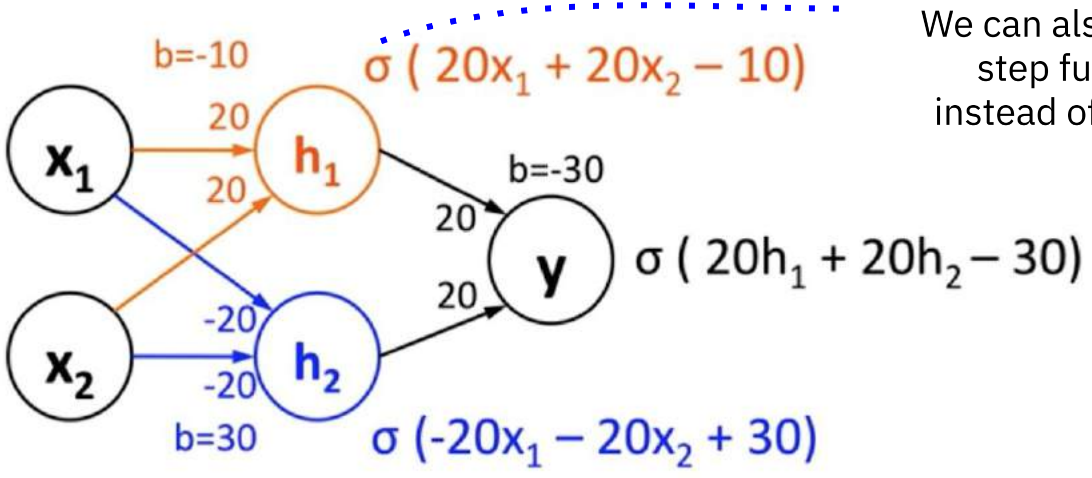


https://datascience.stackexchange.com/questions/11589/creating-neural-net-for-xor-function
http://yen.cs.stir.ac.uk/~kjt/techreps/pdf/TR148.pdf
https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b

NN Example: XOR

Linear classifiers cannot solve this





We can also use the step function instead of sigmoid

x_1 x_2	x_1 x_2	
$\sigma(20*0 + 20*0 - 10) \approx 0$	$\sigma(-20*0 - 20*0 + 30) \approx 1$	$\sigma(20*0 + 20*1 - 30) \approx 0$
$\sigma(20*1 + 20*1 - 10) \approx 1$	$\sigma(-20*1 - 20*1 + 30) \approx 0$	$\sigma(20*1 + 20*0 - 30) \approx 0$
$\sigma(20*0 + 20*1 - 10) \approx 1$	$\sigma(-20*0 - 20*1 + 30) \approx 1$	$\sigma(20*1 + 20*1 - 30) \approx 1$
$\sigma(20*1 + 20*0 - 10) \approx 1$	$\sigma(-20*1 - 20*0 + 30) \approx 1$	$\sigma(20*1 + 20*1 - 30) \approx 1$

https://www.youtube.com/watch?v=kNPGXgzxoHw

NN Example: XOR (Cont'd)

Example: XOR

Now let's consider using a two-layer neural network, with the following equation:

$$g(\mathbf{x}) = \mathbf{w}^T \max(0, \mathbf{W}^T\mathbf{x} + \mathbf{c}) + b$$

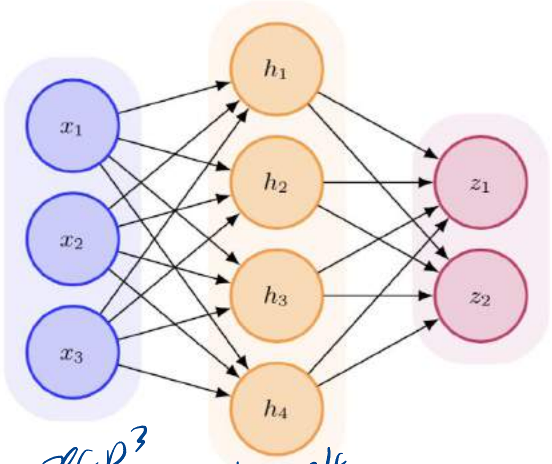
We haven't yet discussed how to optimize these parameters, but the point here is to show that by introducing a simple nonlinearity like $f(x) = \max(0, x)$, we can now solve the $\mathrm{xor}(\cdot)$ problem. Consider the solution:

$$\begin{aligned} \mathbf{W} &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \\ \mathbf{c} &= [0, -1]^T \\ \mathbf{w} &= [1, -2]^T \end{aligned}$$

2-Layer NN Example

Neural network architecture

An example 2-layer network is shown below.



$x \in R^3$ $\quad$ $h \in R^4$

Here, the three dimensional inputs ($\mathbf{x} \in \mathbb{R}^3$) are processed into a four dimensional intermediate representation ($\mathbf{h} \in \mathbb{R}^4$), which are then transormed into the two dimensional outputs ($\mathbf{z} \in \mathbb{R}^2$).

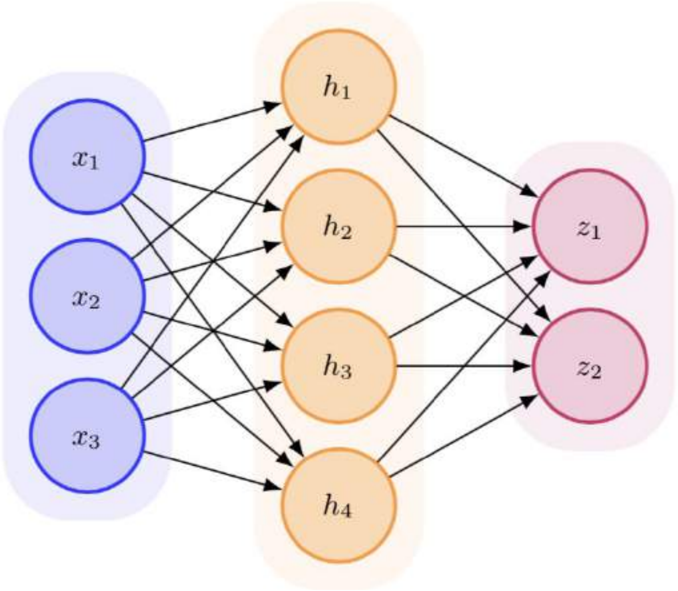
$$\underset{R^4}{h} = f(W_1^T \underset{R^3}{x} + b_1)$$

$$\underset{R^2}{z} = f(W_2^T \underset{R^4}{h} + b_2)$$

$W_1 \in R^{3\times4}$ $\quad$ $b_1 \in R^4$

$W_2 \in R^{4\times2}$ $\quad$ $b_2 \in R^2$

2-Layer NN Example



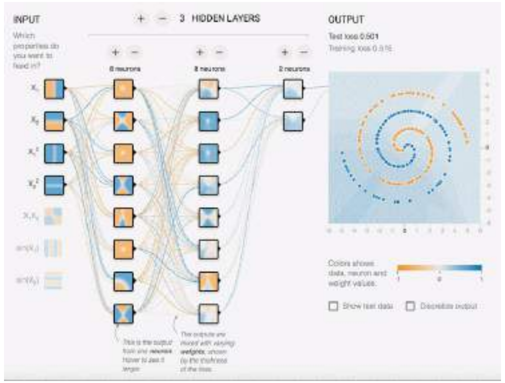
- Layer 1: $\mathbf{h}_1 = f(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$
- Layer 2: $\mathbf{h}_2 = f(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$
- $\vdots$
- Layer N: $\mathbf{z} = \mathbf{W}_N\mathbf{h}_{N-1} + \mathbf{b}_N$

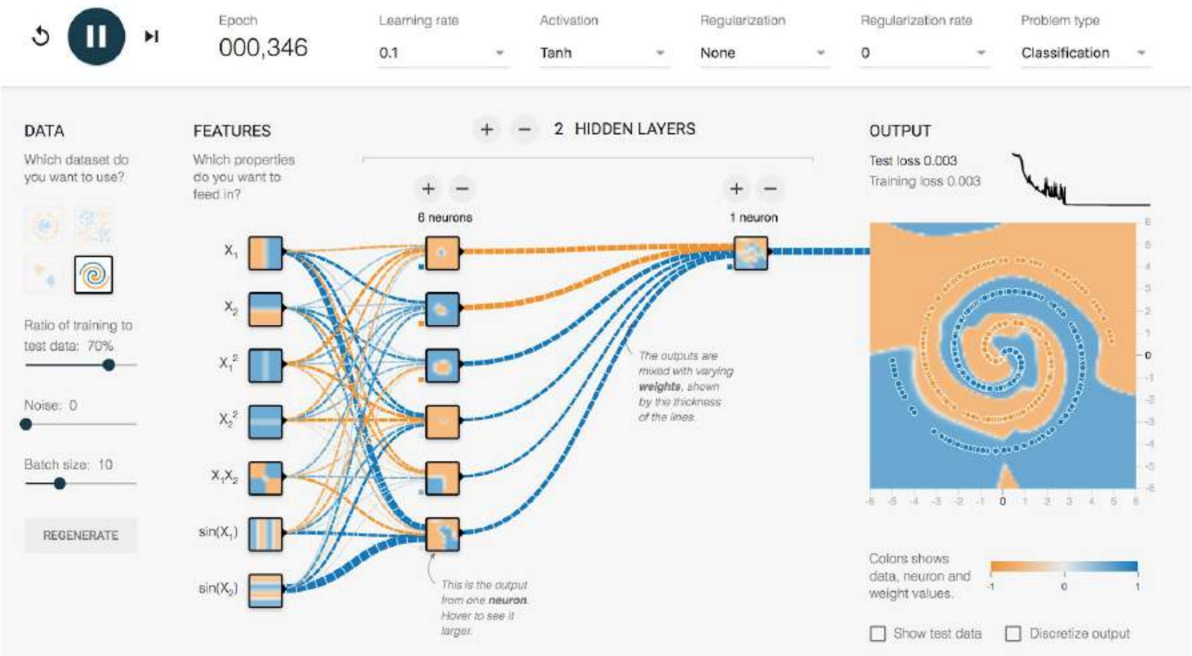
Questions:

1. Neural network model (in equations)
2. Number of neurons?
3. Number of weight parameters / bias parameters / total learnable parameters?

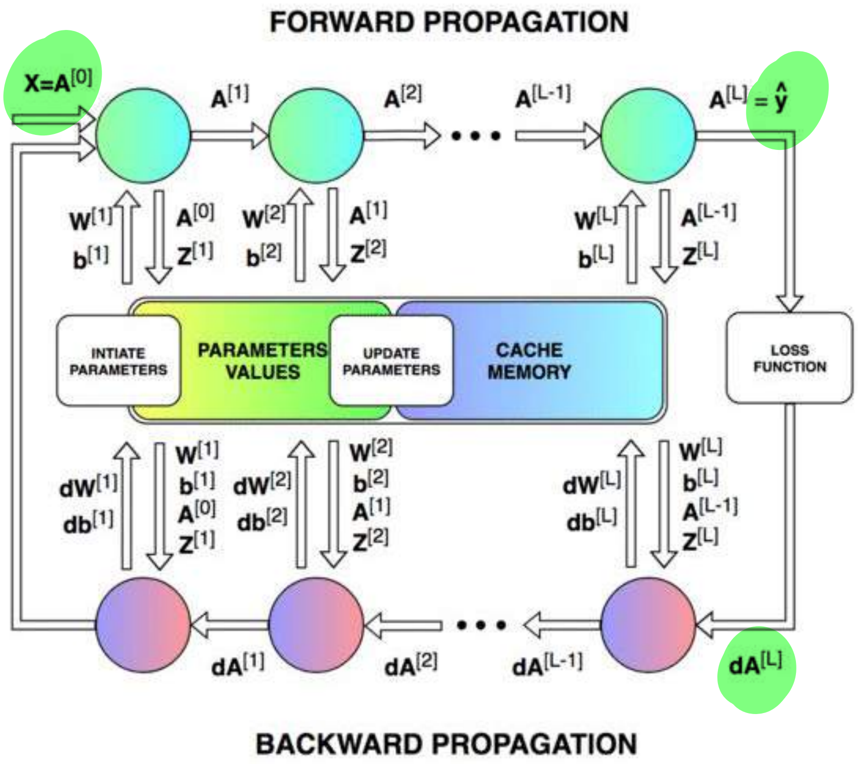
Neural Networks: Demo

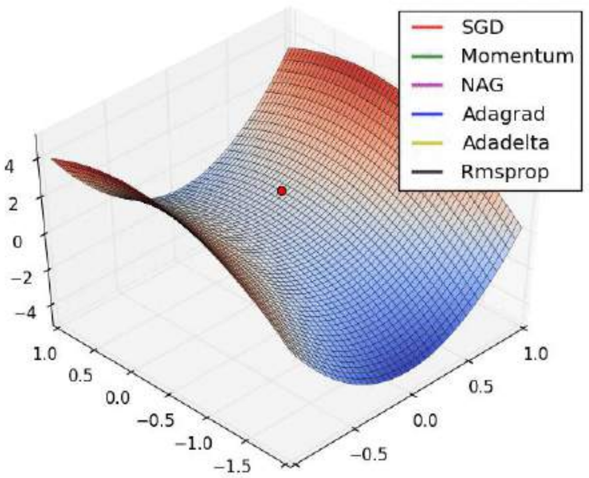
- Let’s play with it: https://playground.tensorflow.org/





Neural Networks: Backpropagation





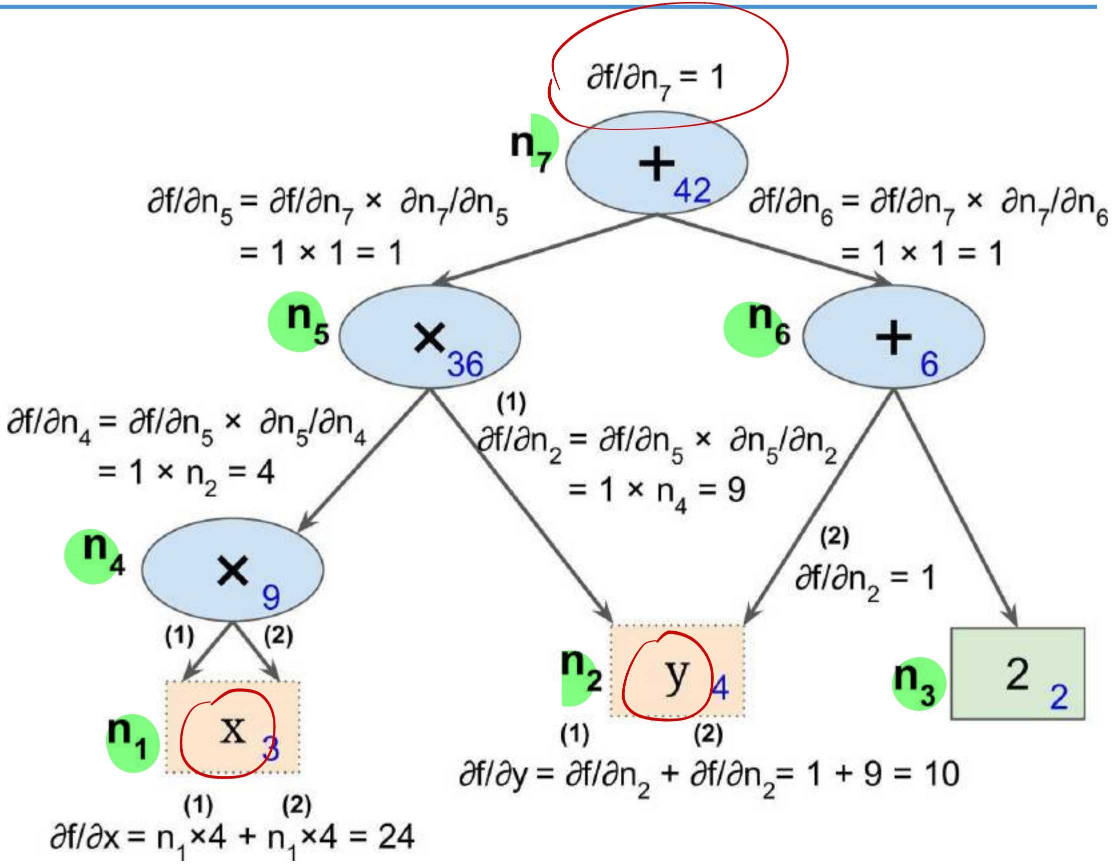
https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

Neural Networks: Backpropagation

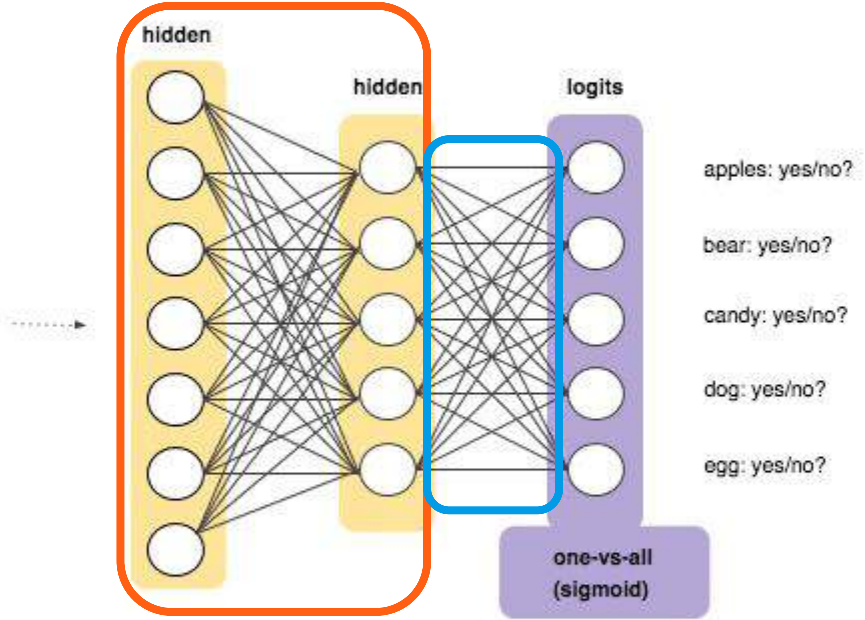
- A simple example to understand the intuition
- **f(x,y) = $x^2y + y + 2$**
- Forward pass:
    - x=3, y=4 → f(3,4)=42
- Backward pass:
    - Chain rule:

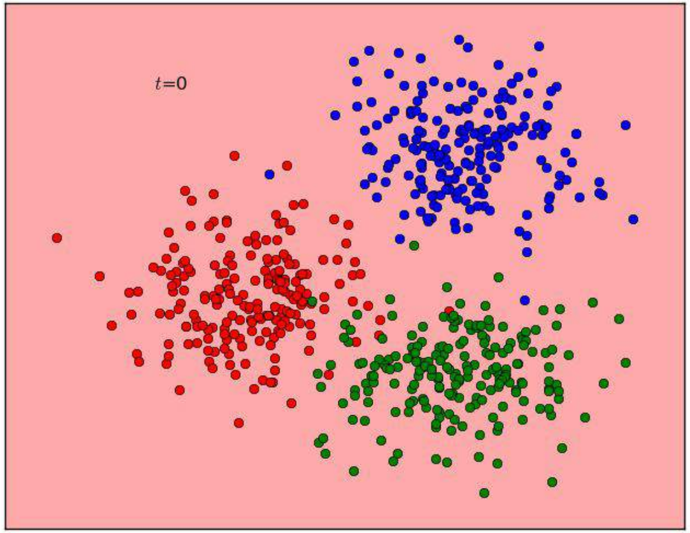
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_i} \times \frac{\partial n_i}{\partial x}$$

Another better demo: http://colah.github.io/posts/2015-08-Backprop/



Multiclass Classification





5 separate **binary classifiers**

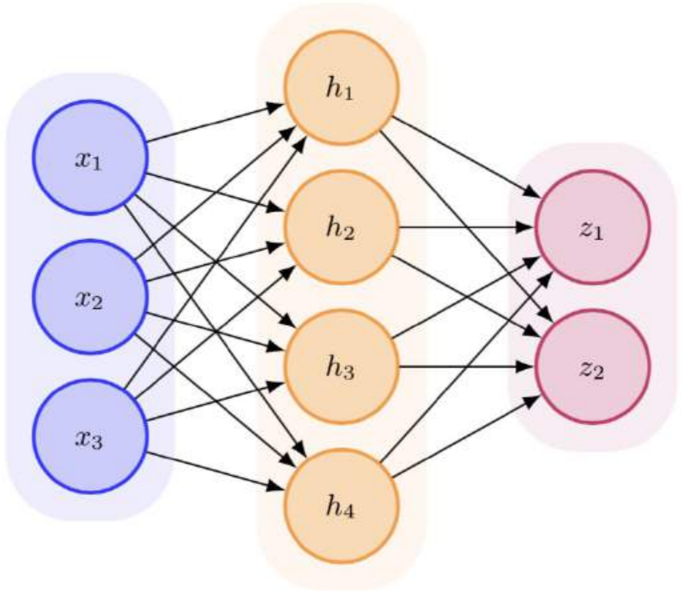
Key: **sharing the same hidden layers** with **different weights at the end**

Question: Pros and cons?

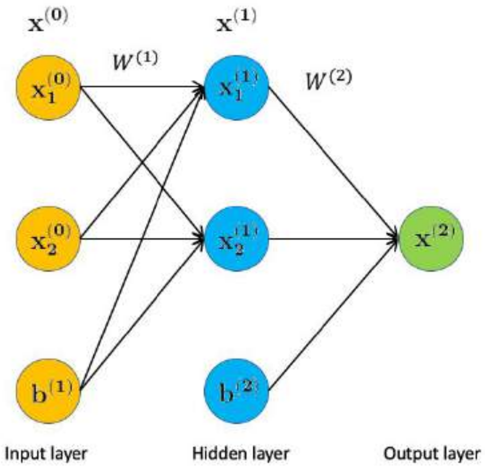
https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/one-vs-all
http://www.briandolhansky.com/blog/2013/9/23/artificial-neural-nets-linear-multiclass-part-3

2-Layer NN Example

Demo in class : Back propagation for a 2-layer network



Backprop: Exercise (next week)



In this question, let's consider a simple two-layer neural network and manually do the forward and backward pass. For simplicity, we assume our input data is two dimension. Then the model architecture looks like the following. Notice that in the example we saw in class, the bias term `b` was not explicit listed in the architecture diagram. Here we include the term `b` explicitly for each layer in the diagram. Recall the formula for computing $\mathbf{x}^{(l)}$ in the l-th layer from $\mathbf{x}^{(l-1)}$ in the $(l-1)$-th layer is $\mathbf{x}^{(l)} = \mathbf{f}^{(l)}(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)})$. The activation function $\mathbf{f}^{(l)}$ we choose is the `sigmoid` function for all layers, i.e. $\mathbf{f}^{(l)}(z) = \frac{1}{1+\exp(-z)}$. The final loss function is $\frac{1}{2}$ of the `mean squared error` loss, i.e. $l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2}||\mathbf{y} - \hat{\mathbf{y}}||^2$.

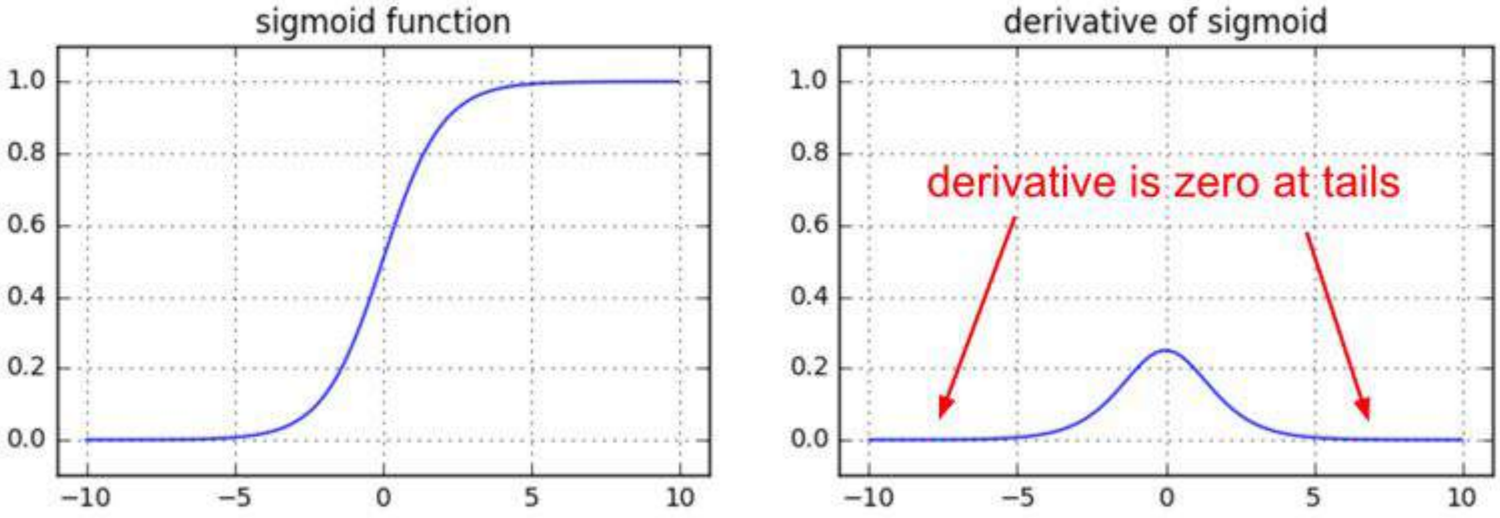
We initialize our weights as

$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.15 & 0.2 \\ 0.25 & 0.3 \end{bmatrix}, \quad \mathbf{W}^{(2)} = [0.4, 0.45], \quad \mathbf{b}^{(1)} = [0.35, 0.35], \quad \mathbf{b}^{(2)} = 0.6$$

1. When the input $\mathbf{x}^{(0)} = [0.05, 0.1]$, what will be the value of $\mathbf{x}^{(1)}$ in the hidden layer? (Show your work).
2. Based on the value $\mathbf{x}^{(1)}$ you computed, what will be the value of $\mathbf{x}^{(2)}$ in the output layer? (Show your work).
3. When the target value of this input is $y = 0.01$, based on the value $\mathbf{x}^{(2)}$ you computed, what will be the loss? (Show your work).

Why understanding backpropagation?

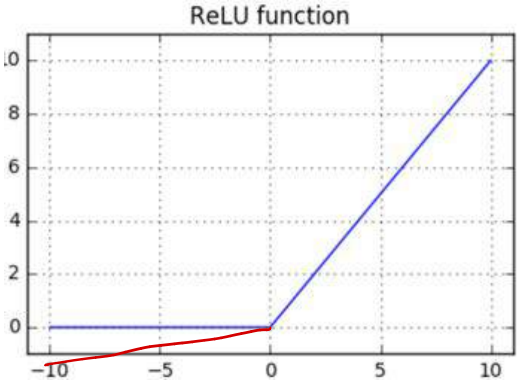
- “Why do we have to write the backward pass when frameworks in the real world, such as TensorFlow/PyTorch, compute them for you automatically?”
- Vanishing gradients on Sigmoids

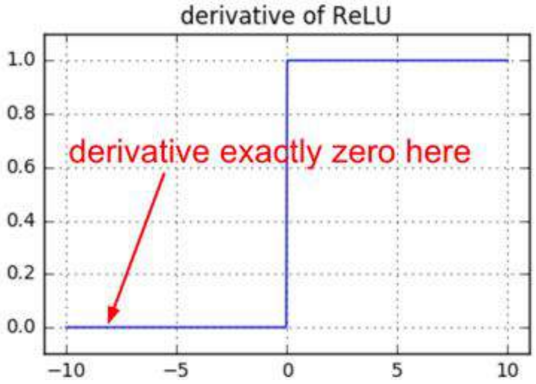


https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

Why understanding backpropagation?

- ReLUs



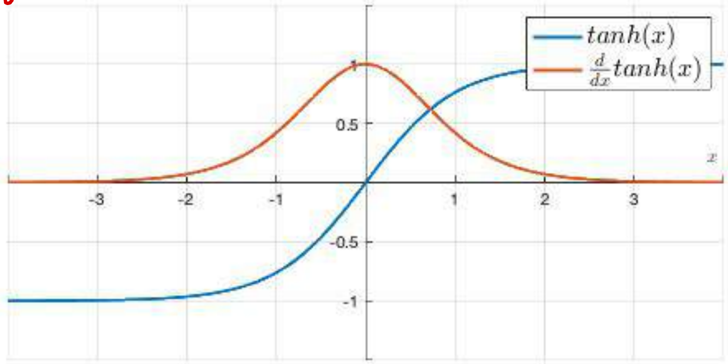


leaky ReLU

$$y = \begin{cases} x, & x \geq 0 \\ kx, & x < 0 \end{cases} \quad (k = 0.01)$$

$x<0$

- tanh



Why understanding backpropagation?

- Examples of activation function: Sigmoid, ReLU, leaky ReLU, **tanh**, etc
- Properties we focus:
  - Differentiable
  - Range: Whether saturated or not? (
  - Whether zero-centered or not?
- Activation function family
  - Wiki: https://en.wikipedia.org/wiki/Activation_function

NN: Backpropagation Reading List

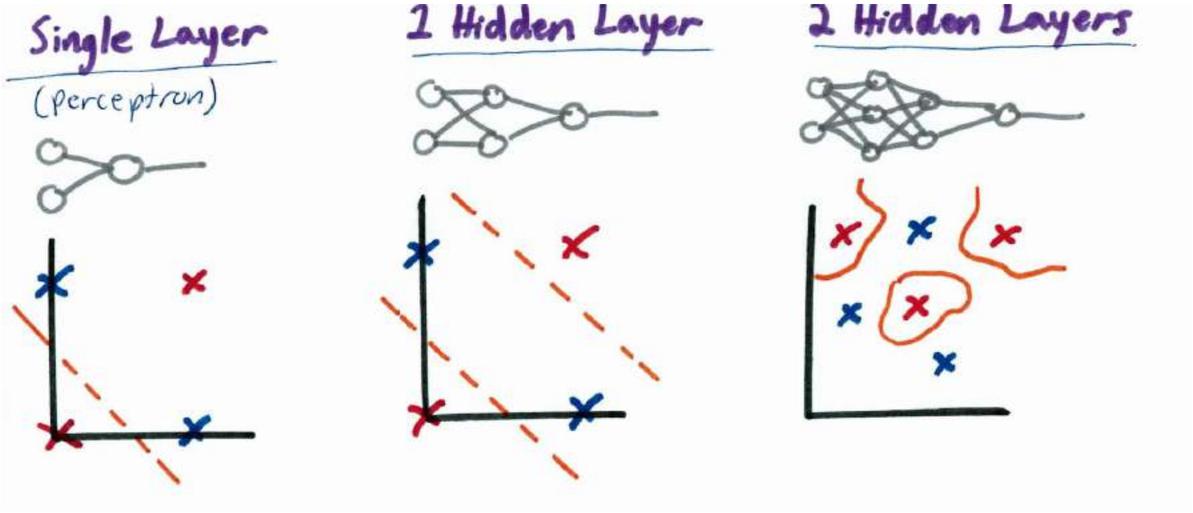
- Backpropagation (CS 231N at Stanford)
  - https://cs231n.github.io/optimization-2/
  - https://www.youtube.com/watch?v=i94OvYb6noo
- (Optional) Matrix-Level Operation:
  - https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c

NN: Number of iterations to converge

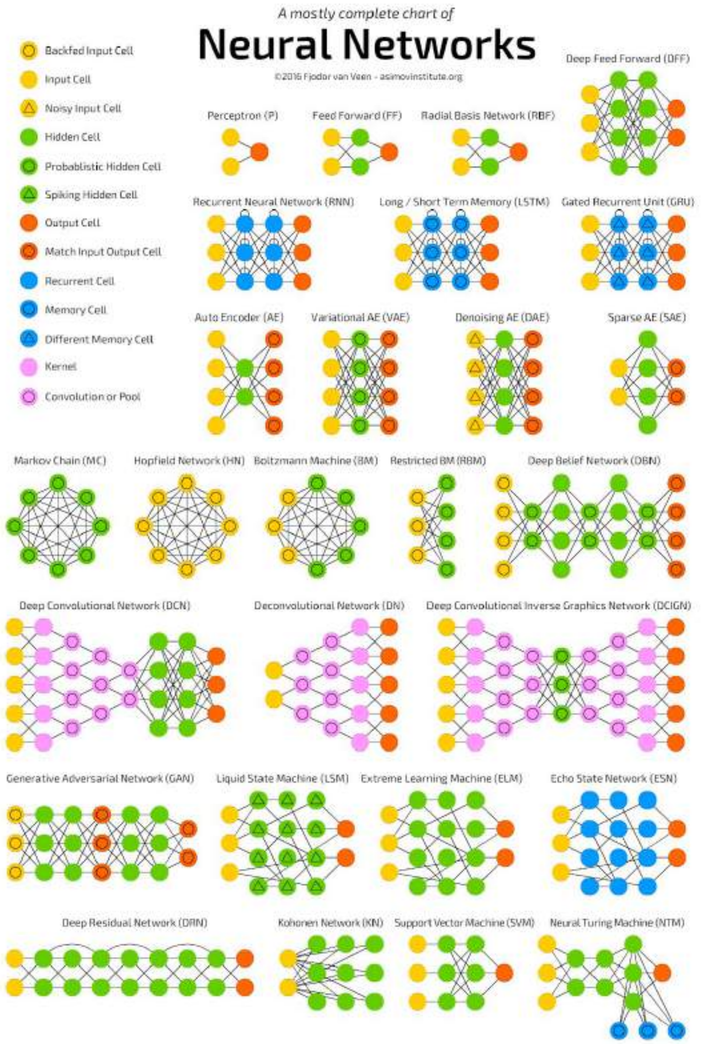
- Architecture/Meta-parameters of the network, e.g. # layers, activation
- Quality of training data (input-output correlation, normalization, noise cleansing, class distribution/imbalance)
- Random initialization of the parameters/weights
- Optimization algorithm, e.g. SGD, Adam, etc.
- Learning rate
- Batch size
- (In practice) Implementation quality (Bug-free? Optimized?)

https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e
https://www.quora.com/Machine-Learning-What-are-some-tips-and-tricks-for-training-deep-neural-networks

NN Summary

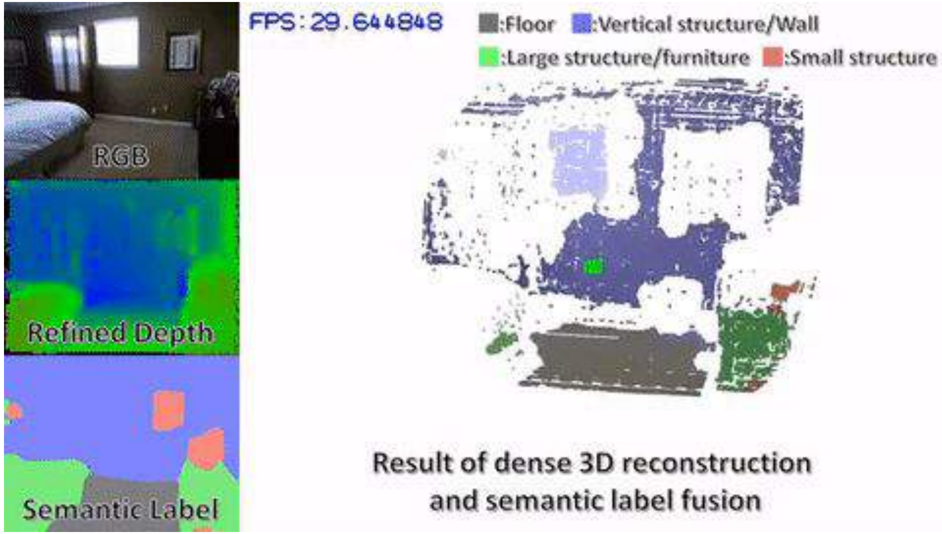


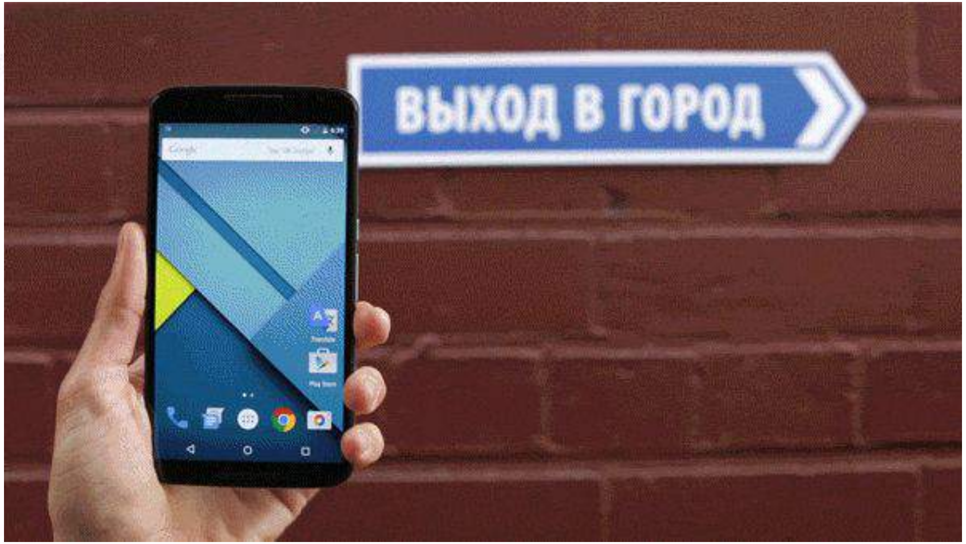
Flexibility



https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec27/pros-and-cons-of-neural-networks
http://kseow.com/nn/
https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b

NN Summary: Pros and Cons

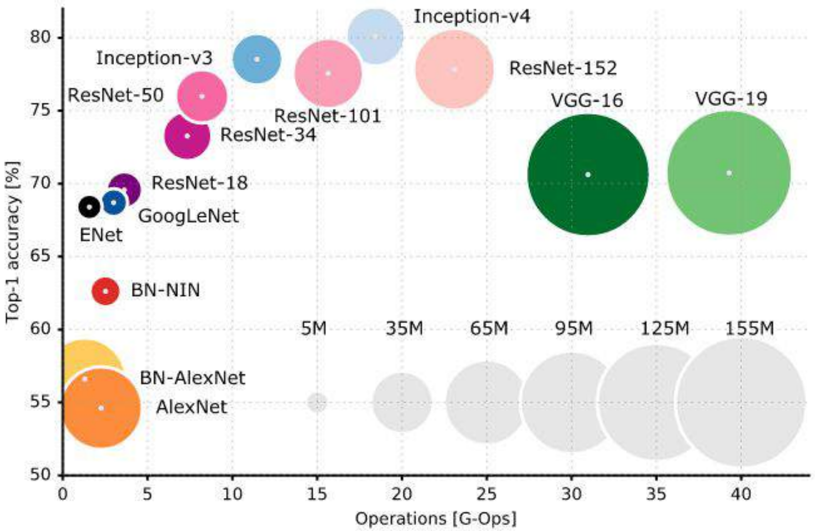




Efficiency (In many cases, prediction/inference/testing is fast)

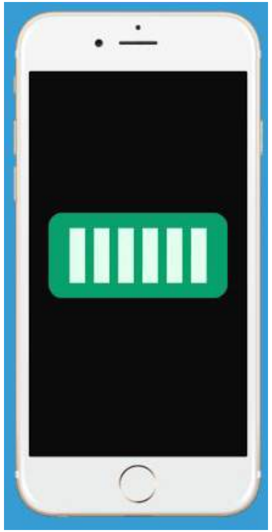
https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec27/pros-and-cons-of-neural-networks
http://www.luigifreda.com/2017/04/08/cnn-slam-real-time-dense-monocular-slam-learned-depth-prediction/
http://www.missqt.com/google-translate-app-now-supports-instant-voice-and-visual-translations/

NN Summary: Pros and Cons



We trained both our baseline models for about 600,000 iterations (33 epochs) – this is similar to the 35 epochs required by Nallapati et al.'s (2016) best model. Training took 4 days and 14 hours for the 50k vocabulary model, and 8 days 21 hours for the 150k vocabulary model. We found the pointer-generator model quicker to train, requiring less than 230,000 training iterations (12.8 epochs); a total of 3 days and 4 hours. In particular, the pointer-generator model makes much quicker progress in the early phases of training.

ments. This work was begun while the first author was an intern at Google Brain and continued at Stanford. Stanford University gratefully acknowl-



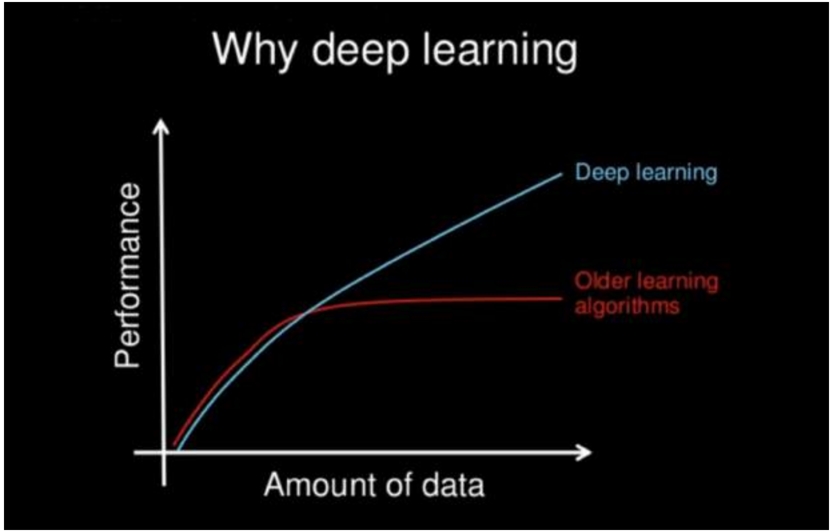
Efficiency (Big model → slow training, huge energy consumption (e.g. for cell phone))

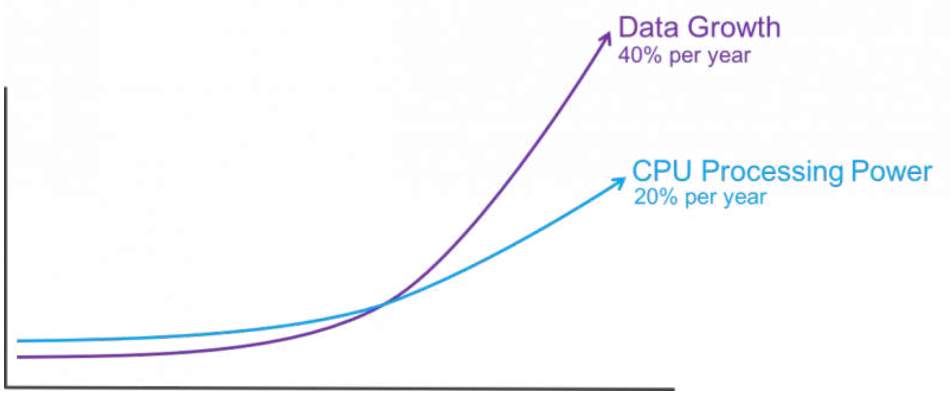
https://www.kdnuggets.com/2017/08/first-steps-learning-deep-learning-image-classification-keras.html/2

See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." *arXiv preprint arXiv:1704.04368* (2017).

https://www.lifewire.com/my-iphone-wont-charge-what-do-i-do-2000147

NN Summary: Pros and Cons

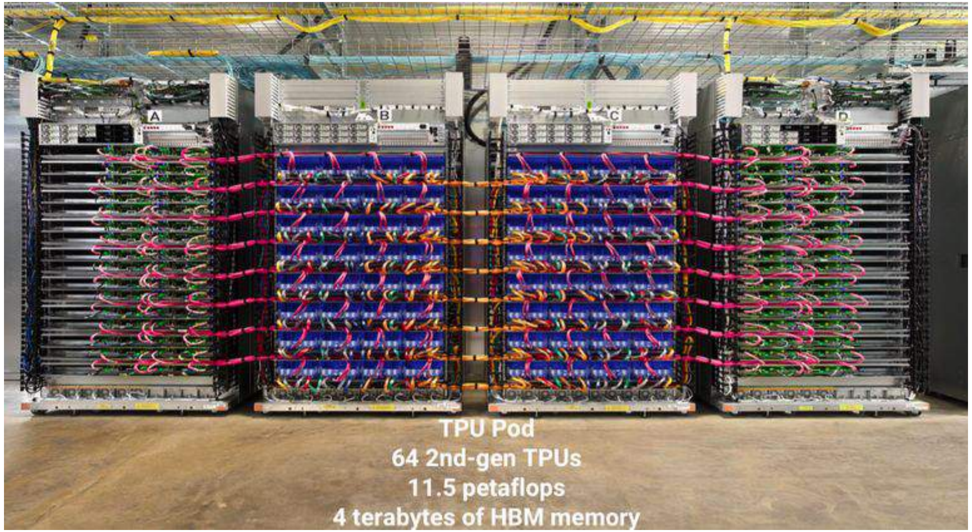




Data (Both a pro and a con)

https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b

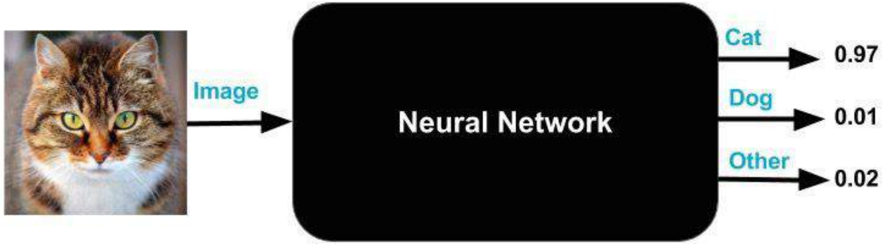
NN Summary: Pros and Cons



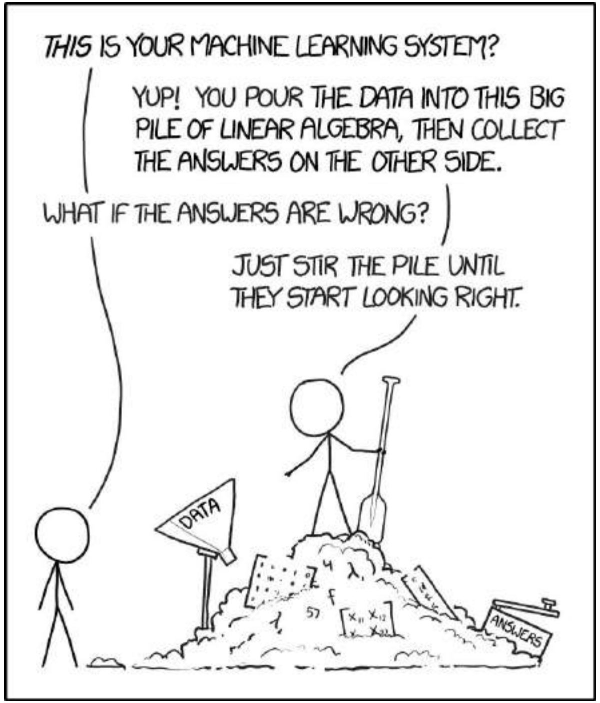
Computational Power (Both a pro and a con)

https://www.anandtech.com/show/10864/discrete-desktop-gpu-market-trends-q3-2016
https://www.zdnet.com/article/gpu-killer-google-reveals-just-how-powerful-its-tpu2-chip-really-is/

NN Summary: Pros and Cons



Black Box
Interpretability



https://towardsdatascience.com/hype-disadvantages-of-neural-networks-6af04904ba5b
https://xkcd.com/1838/

What's next?

- In next week's discussion, we will continue to discuss:
  - Backpropagation in neural nets
- Programming Guide
  - PyTorch (for PS3)

Thank you!

Q & A

CS M146 Discussion: Week 6

Neural Networks, Learning Theory, Kernels, PyTorch

Junheng Hao

Friday, 02/12/2021

Roadmap

- Announcement
- Neural Nets: Back Propagation
- Learning Theory
- Programming Guide: PyTorch

Happy Holidays!

No lecture next Monday (Feb 15)!

Announcements

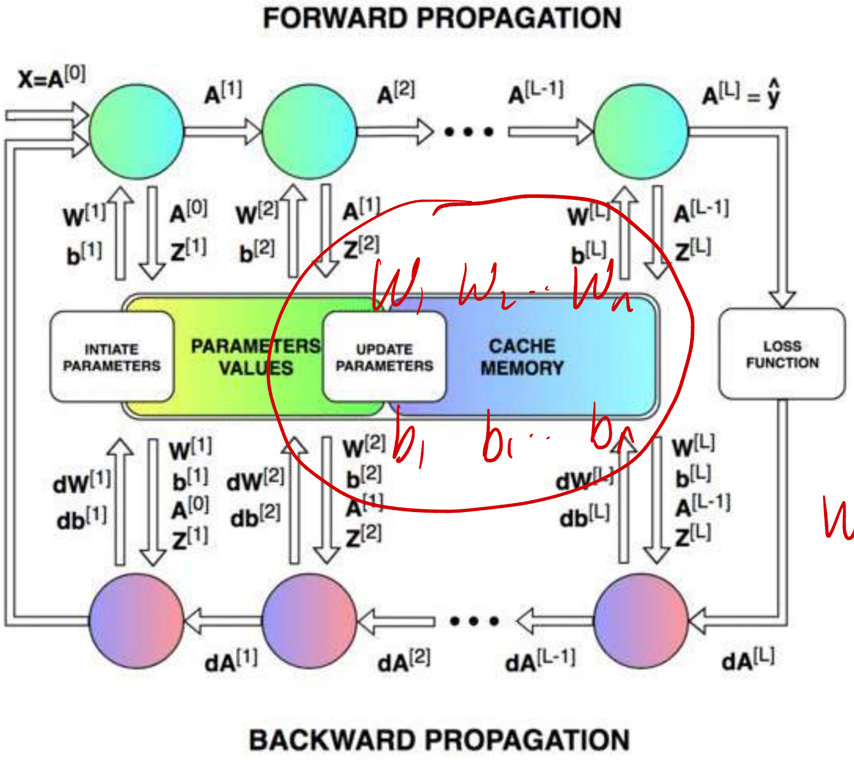
- **5:00 pm PST, Feb 12 (Friday):** Weekly Quiz 6 released on Gradescope.
- **11:59 pm PST, Feb 14 (Sunday):** Weekly quiz 6 closed on Gradescope!
  - Start the quiz before **11:00 pm Feb 14, Feb 14** to have the full 60-minute time
- **Problem set 1: Regrade request due today**
- **Problem set 3: Problem set 1:** Will be released later today, due **Feb 26 11:59PM PST**
- **Problem set 2** submission on Gradescope.
  - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
  - Due on **TODAY 11:59pm PST, Feb 12 (Friday)**

Late Submission of PS will NOT be accepted!

About Quiz 6

- Quiz release date and time: **Feb 12, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Feb 14, 2021 (Sunday) 11:59 PM PST**
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week ~~4~~ 6 Quiz" on GradeScope.
- Topics: **Neural Nets, Learning Theory**
- Question Types
  - True/false, multiple choices
  - Some questions may include several subquestions.
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.

Neural Networks: Backpropagation

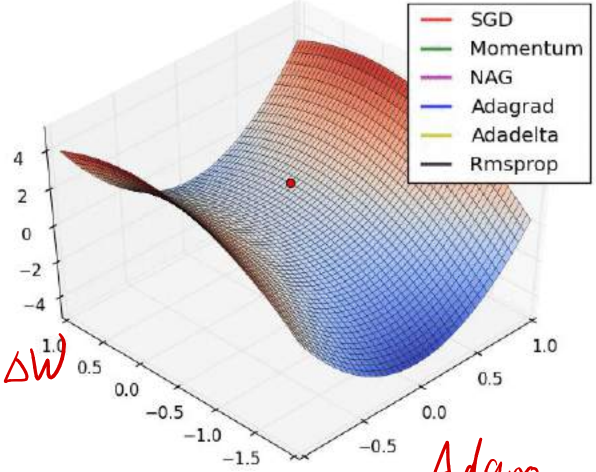


$W_1 \; W_2 \cdots W_n$

$b_1 \; b_2 \cdots b_n$

∂l

$$W' = W - \eta \cdot \Delta W$$



Adam

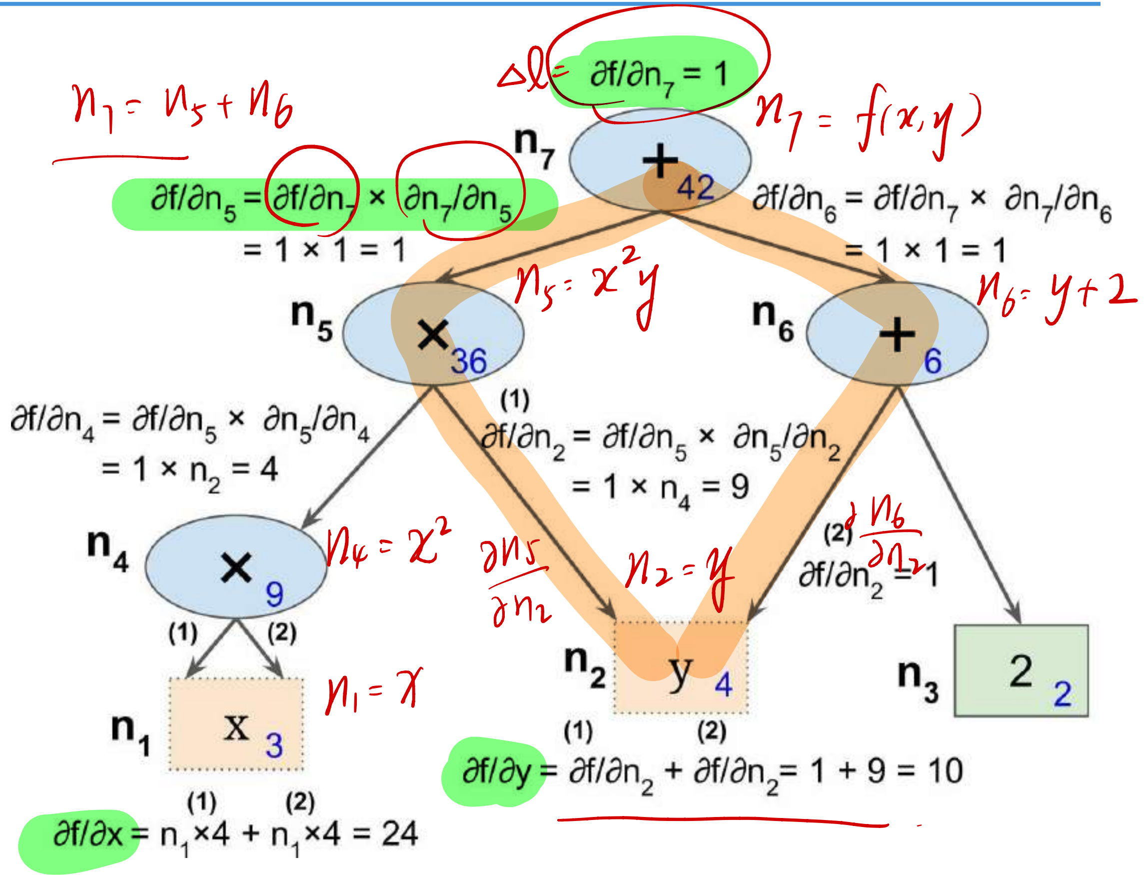
https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e

Neural Networks: Backpropagation

- A simple example to understand the intuition
- **f(x,y) = x^2y + y + 2**
- Forward pass:
  - x=3, y=4 → f(3,4)=42
- Backward pass:
  - Chain rule:

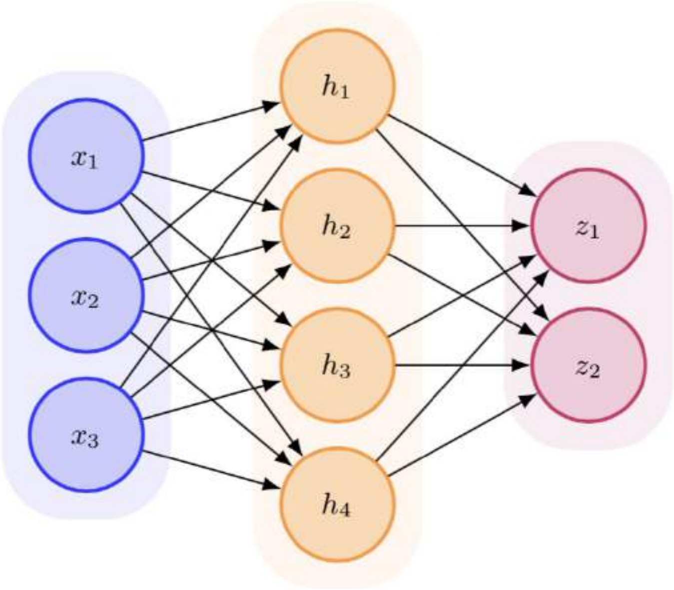
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_i} \times \frac{\partial n_i}{\partial x}$$

Another better demo:
http://colah.github.io/posts/2015-08-Backprop/

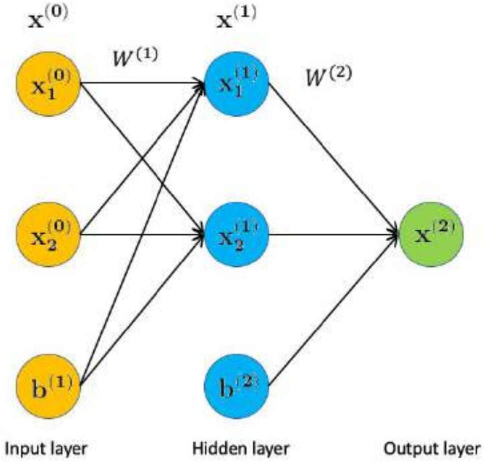


2-Layer NN Example

Demo in class : Back propagation for a 2-layer network



Backprop: Exercise

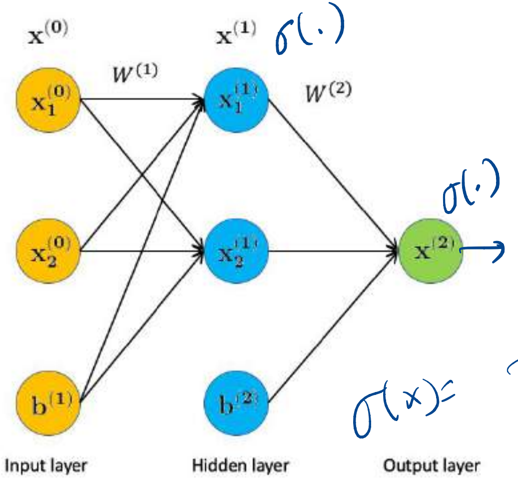


In this question, let's consider a simple two-layer neural network and manually do the forward and backward pass. For simplicity, we assume our input data is two dimension. Then the model architecture looks like the following. Notice that in the example we saw in class, the bias term `b` was not explicit listed in the architecture diagram. Here we include the term `b` explicitly for each layer in the diagram. Recall the formula for computing $\mathbf{x}^{(l)}$ in the l-th layer from $\mathbf{x}^{(l-1)}$ in the $(l-1)$-th layer is $\mathbf{x}^{(l)} = \mathbf{f}^{(l)}(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)})$. The activation function $\mathbf{f}^{(l)}$ we choose is the `sigmoid` function for all layers, i.e. $\mathbf{f}^{(l)}(z) = \frac{1}{1+\exp(-z)}$. The final loss function is $\frac{1}{2}$ of the `mean squared error` loss, i.e. $l(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2}||\mathbf{y} - \hat{\mathbf{y}}||^2$.

We initialize our weights as

$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.15 & 0.2 \\ 0.25 & 0.3 \end{bmatrix}, \quad \mathbf{W}^{(2)} = [0.4, 0.45], \quad \mathbf{b}^{(1)} = [0.35, 0.35], \quad \mathbf{b}^{(2)} = 0.6$$

Backprop: Exercise



$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.15 & 0.2 \\ 0.25 & 0.3 \end{bmatrix}, \quad \mathbf{W}^{(2)} = [0.4, 0.45],$$

$$\mathbf{b}^{(1)} = [0.35, 0.35], \quad \mathbf{b}^{(2)} = 0.6$$

$$\text{input } \mathbf{x}^{(0)} = [0.05, 0.1]^T$$

Forward Pass

1. When the input $\mathbf{x}^{(0)} = [0.05, 0.1]$, what will be the value of $\mathbf{x}^{(1)}$ in the hidden layer? (Show your work).
2. Based on the value $\mathbf{x}^{(1)}$ you computed, what will be the value of $\mathbf{x}^{(2)}$ in the output layer? (Show your work).
3. When the target value of this input is $y = 0.01$, based on the value $\mathbf{x}^{(2)}$ you computed, what will be the loss? (Show your work).

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$z^{(1)} = W^{(1)} X^{(0)} + b^{(1)} = \begin{bmatrix} 0.3775 \\ 0.3925 \end{bmatrix}$$

$$x^{(1)} = \sigma(z^{(1)}) = \begin{bmatrix} 0.5933 \\ 0.5969 \end{bmatrix}$$

σ

$$z^{(2)} = W^{(2)} X^{(1)} + b^{(2)} =$$

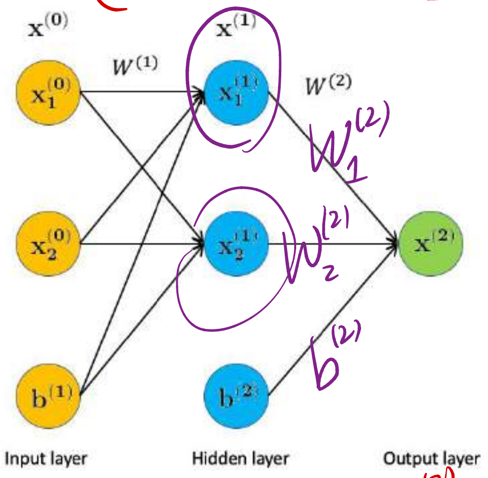
output

$$x^{(2)} = \sigma(z^{(2)}) =$$

$$l = \frac{1}{2}(x^{(2)} - \hat{y})^2 =$$

Backprop: Exercise

Back Propagation $Z^{(2)} = W_1^{(2)} x_1^{(1)} + W_2^{(2)} \cdot x_2^{(1)} + b^{(2)}$



1. Consider the loss l of the same input $\mathbf{x}^{(0)} = [0.05, 0.1]$, what will be the update of $\mathbf{W}^{(2)}$ and $\mathbf{b}^{(2)}$ when we backprop, i.e. $\frac{\partial l}{\partial \mathbf{W}^{(2)}}, \frac{\partial l}{\partial \mathbf{b}^{(2)}}$
2. Based on the result you computed in part 1, when we keep backproping, what will be the update of $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$, i.e. $\frac{\partial l}{\partial \mathbf{W}^{(1)}}, \frac{\partial l}{\partial \mathbf{b}^{(1)}}$

$$l = \frac{1}{2}(x^{(2)} - y)^2$$

$$\Delta l = \frac{\partial l}{\partial x^{(2)}} = (x^{(2)} - y)$$

$$\sigma' = \sigma(1-\sigma)$$

$$\frac{\partial l}{\partial W^{(2)}} = \left[\frac{\partial l}{\partial W_1^{(2)}}, \frac{\partial l}{\partial W_2^{(2)}}\right]$$

$$\frac{\partial l}{\partial W_1^{(2)}} = \frac{\partial l}{\partial x^{(2)}} \cdot \frac{\partial x^{(2)}}{\partial Z^{(2)}} \cdot \frac{\partial Z^{(2)}}{\partial W_1^{(2)}}$$

$\sigma(Z^{(2)})(1-\sigma(Z^{(2)}))$ $\qquad x_1^{(1)}$ $\qquad x_2^{(1)}$

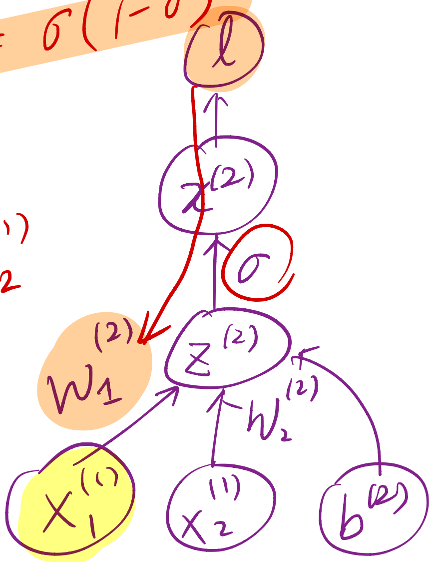
$W_1^{(2)}$

$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.15 & 0.2 \\ 0.25 & 0.3 \end{bmatrix}, \quad \mathbf{W}^{(2)} = [0.4, 0.45],$$

$$\mathbf{b}^{(1)} = [0.35, 0.35], \quad \mathbf{b}^{(2)} = 0.6$$

$$\text{input } \mathbf{x}^{(0)} = [0.05, 0.1]$$

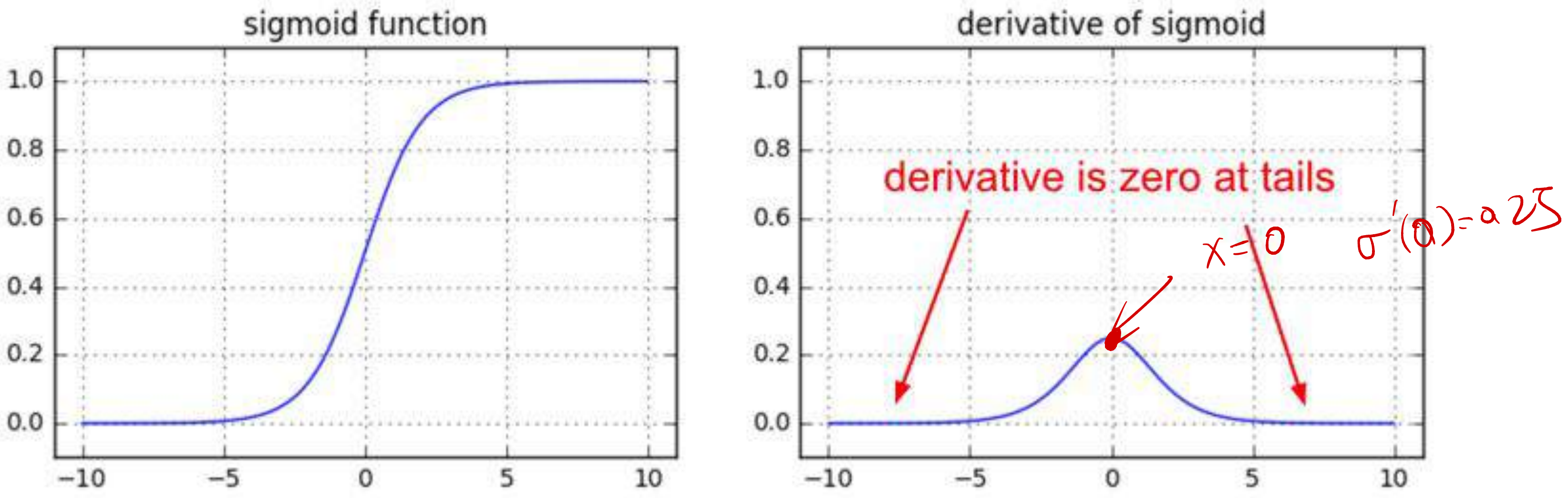
target value of this input is $y = 0.01$



Why understanding activation function?

- "Why do we have to write the backward pass when frameworks in the real world, such as TensorFlow/PyTorch, compute them for you automatically?"
- Vanishing gradients on Sigmoids

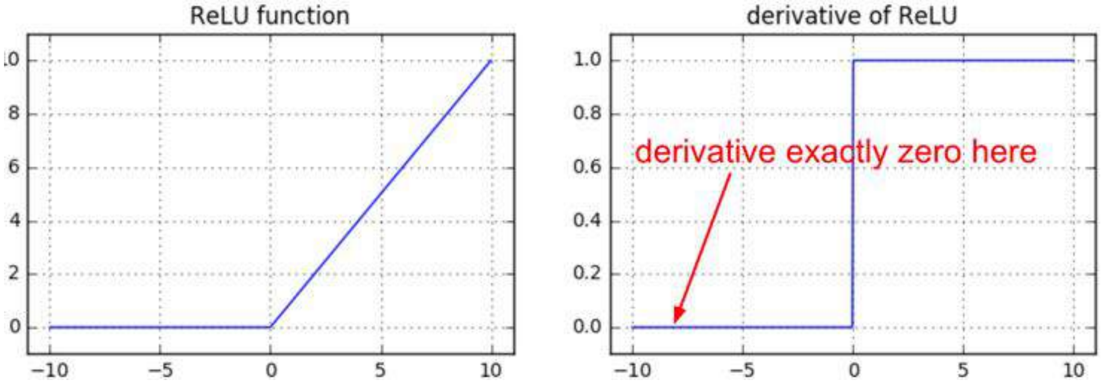
$\sigma(1-\sigma)$



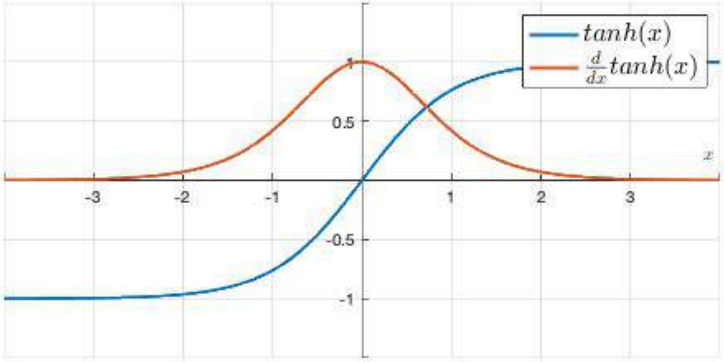
https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b

Why understanding backpropagation?

- ReLUs



- tanh

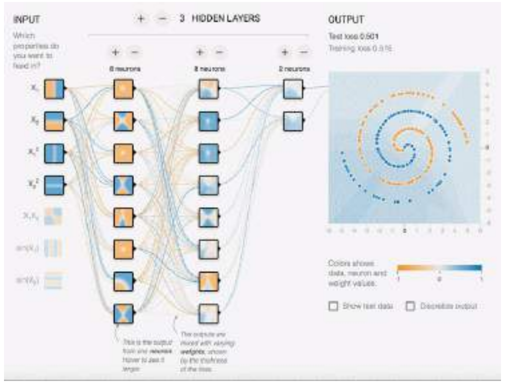


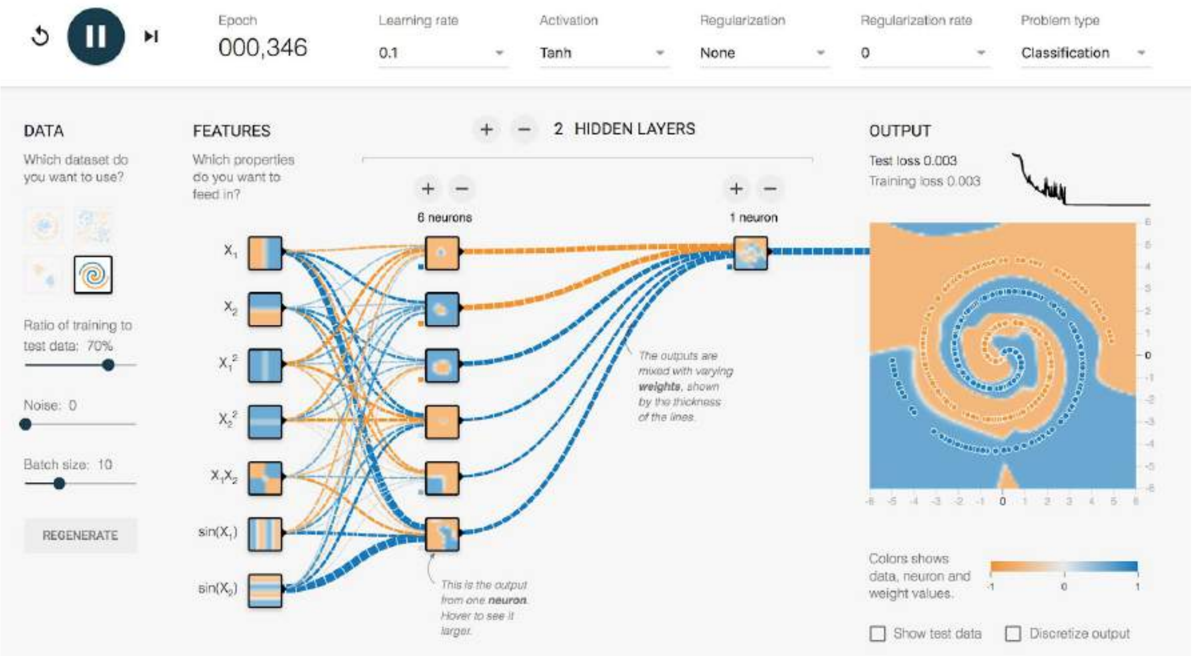
Why understanding backpropagation?

- Examples of non-linear activation functions: `Sigmoid, ReLU, leaky ReLU,` **`tanh,`** `etc`
- Properties we focus on:
  - Differentiable
  - Range: Whether saturated or not? (
  - Whether zero-centered or not?
- Activation function family
  - Wiki: https://en.wikipedia.org/wiki/Activation_function

Neural Networks: Online Demo

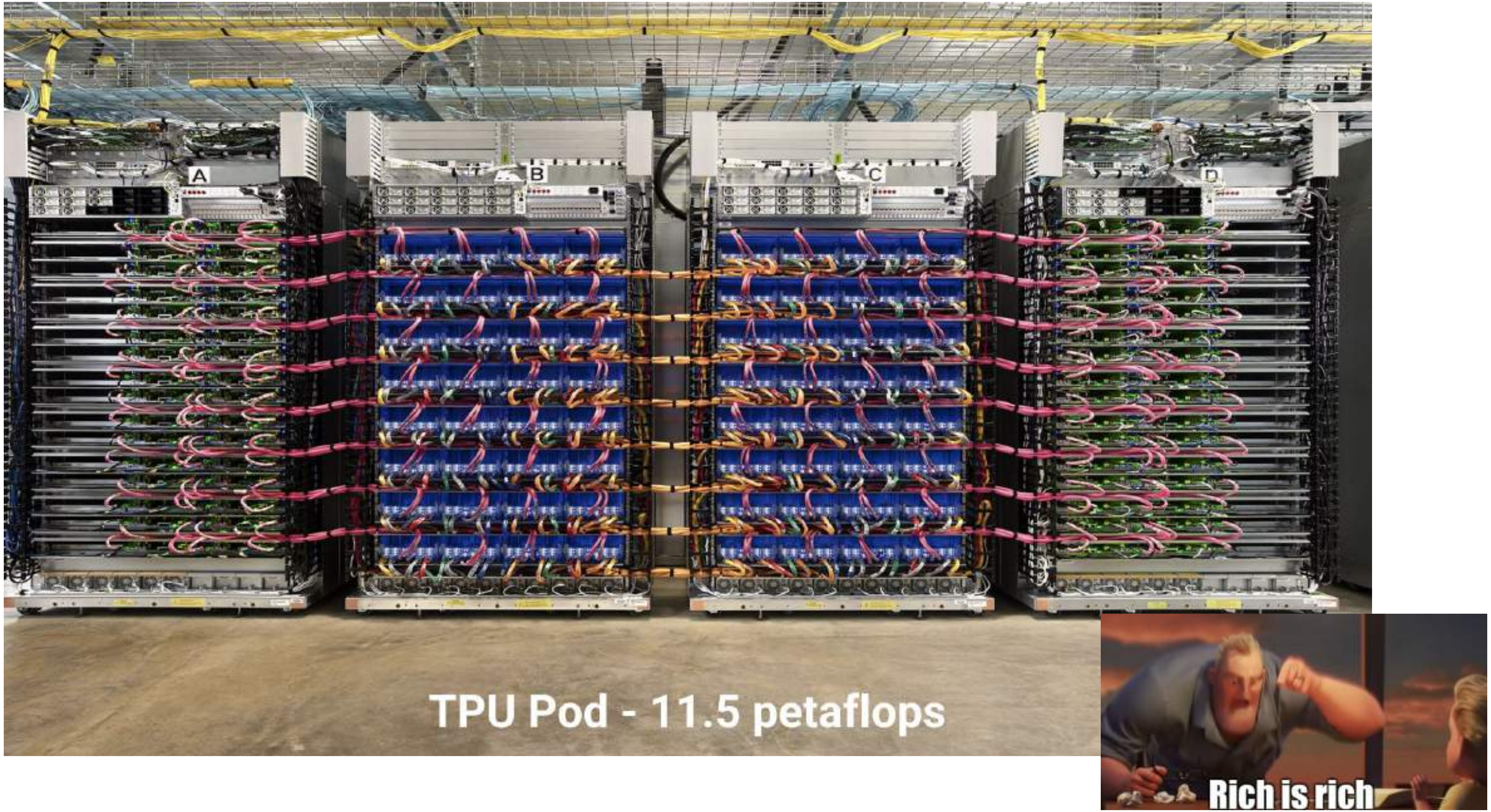
- Let’s play with it: https://playground.tensorflow.org/



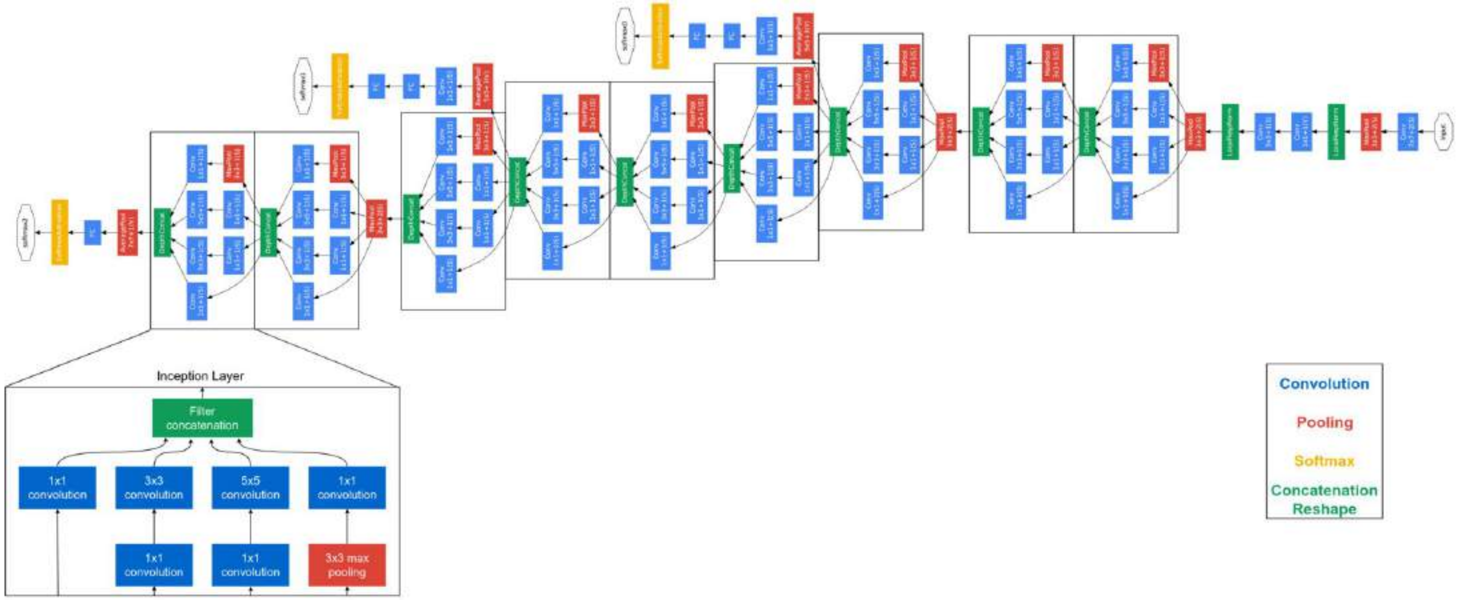


UCLA Engineer Change.

Story of Computing

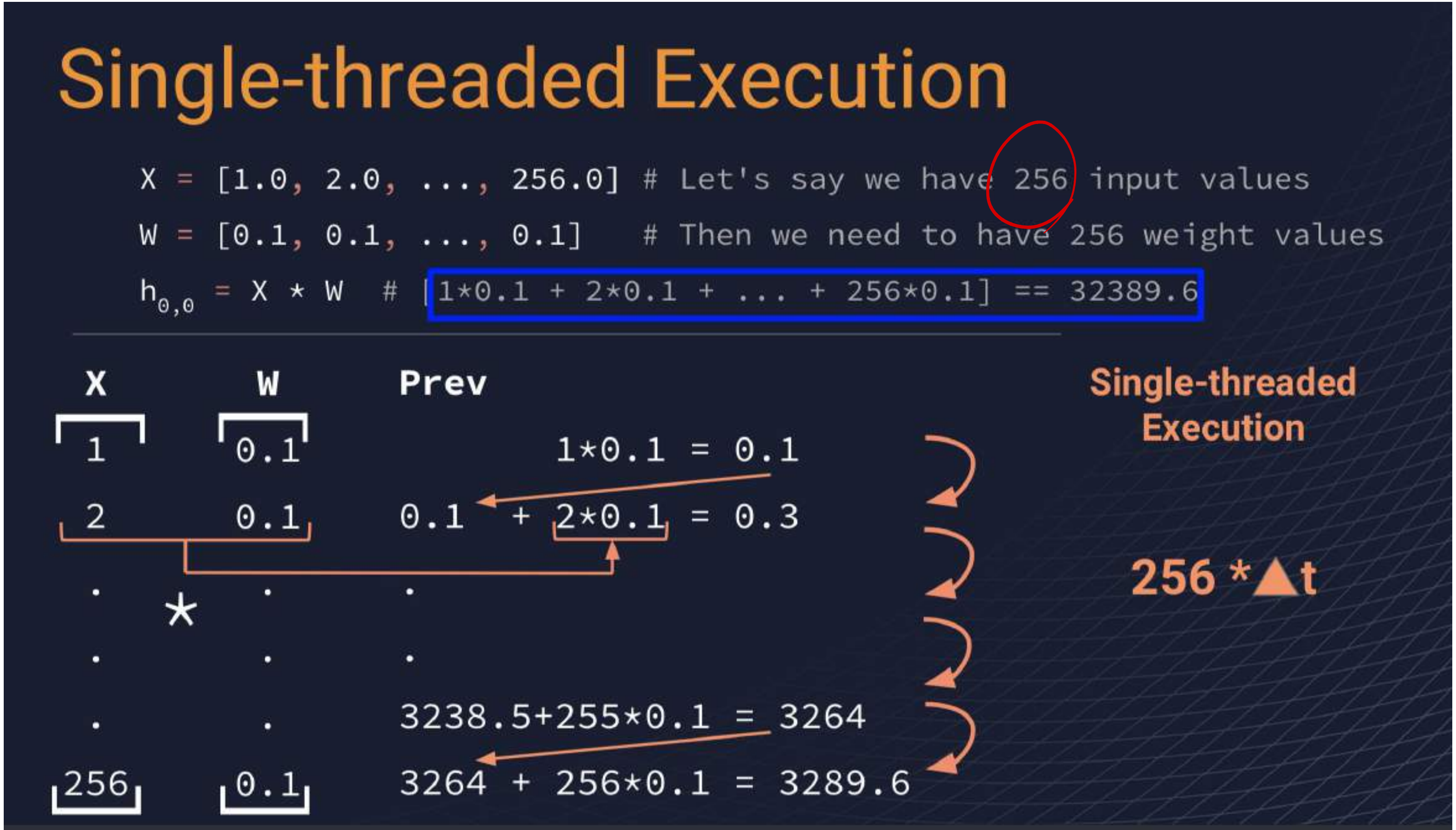


Story of Computing

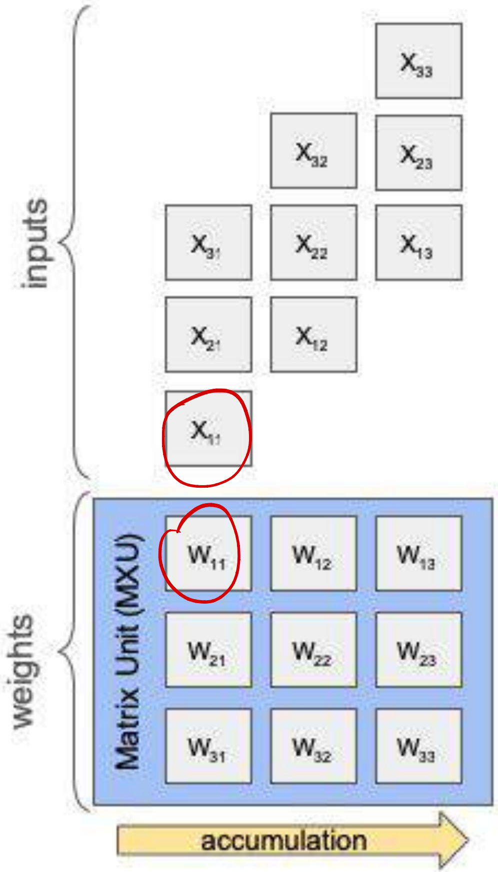


Matrix Multiplication is Eating (the computing resource of) the World!

Single-thread Computing of *X*W*



Neural Networks: Computation Example

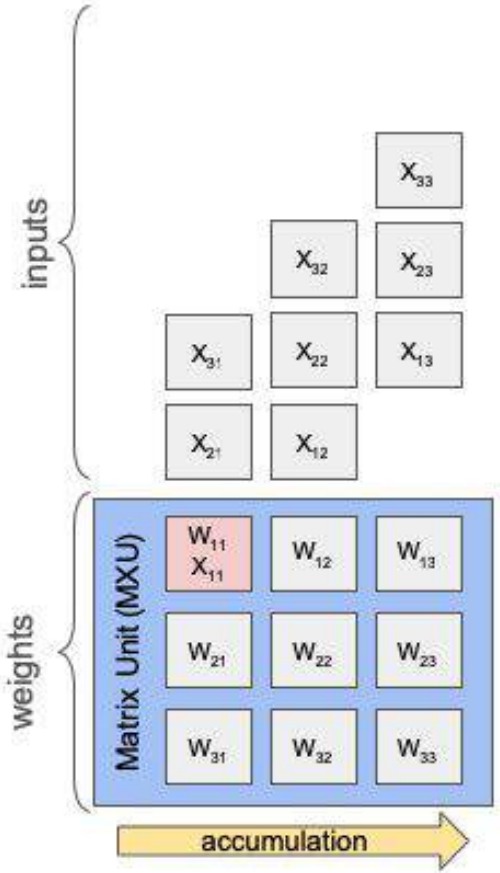


Matrix Unit Systolic Array

Computing y = Wx

3x3 systolic array
W = 3x3 matrix
Batch-size(x) = 3

Neural Networks: Computation Example

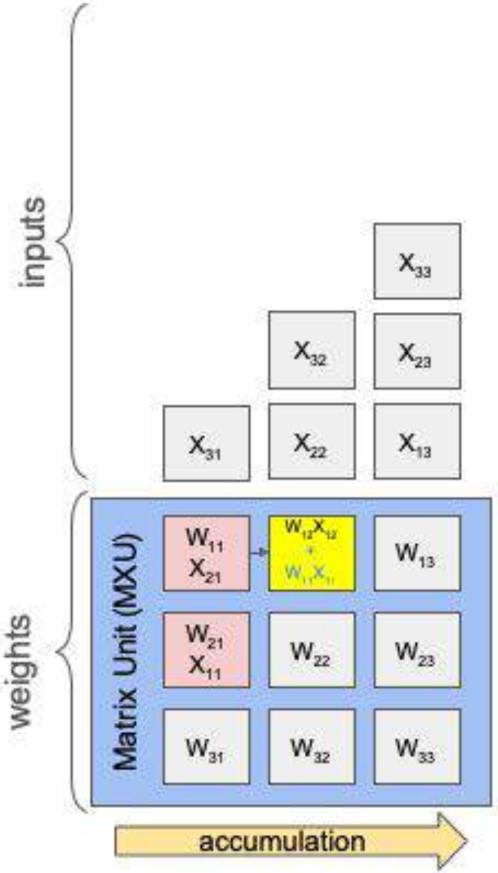


Matrix Unit Systolic Array

Computing y = Wx

with W = 3x3, batch-size(x) = 3

Neural Networks: Computation Example

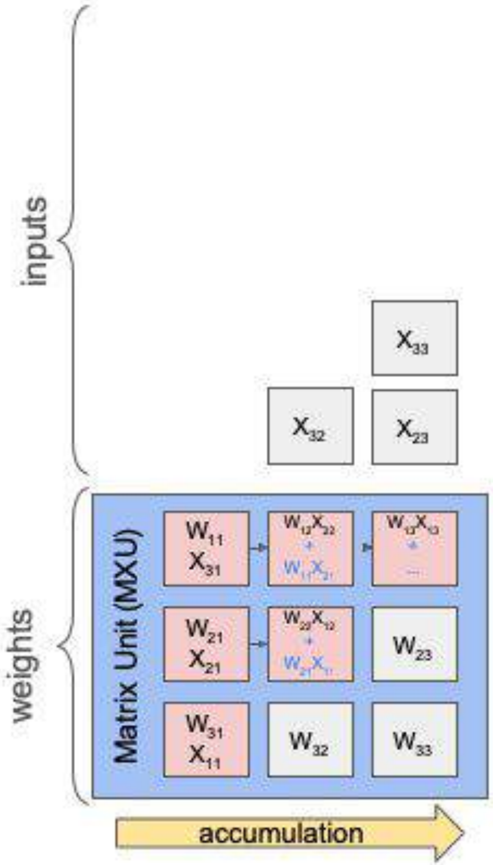


Matrix Unit Systolic Array

Computing $y = Wx$

with $W = 3x3$, batch-size$(x) = 3$

Neural Networks: Computation Example

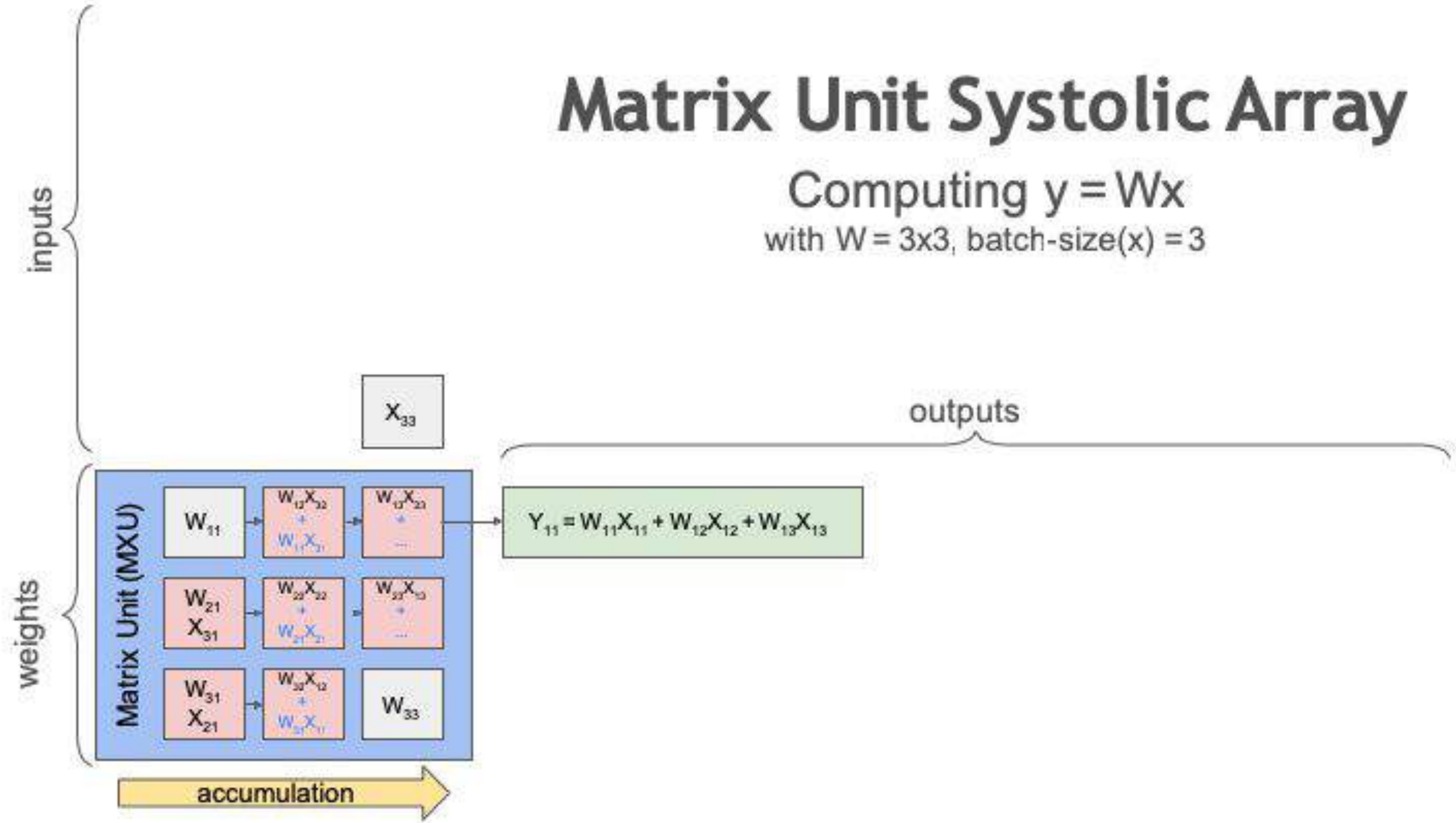


Matrix Unit Systolic Array

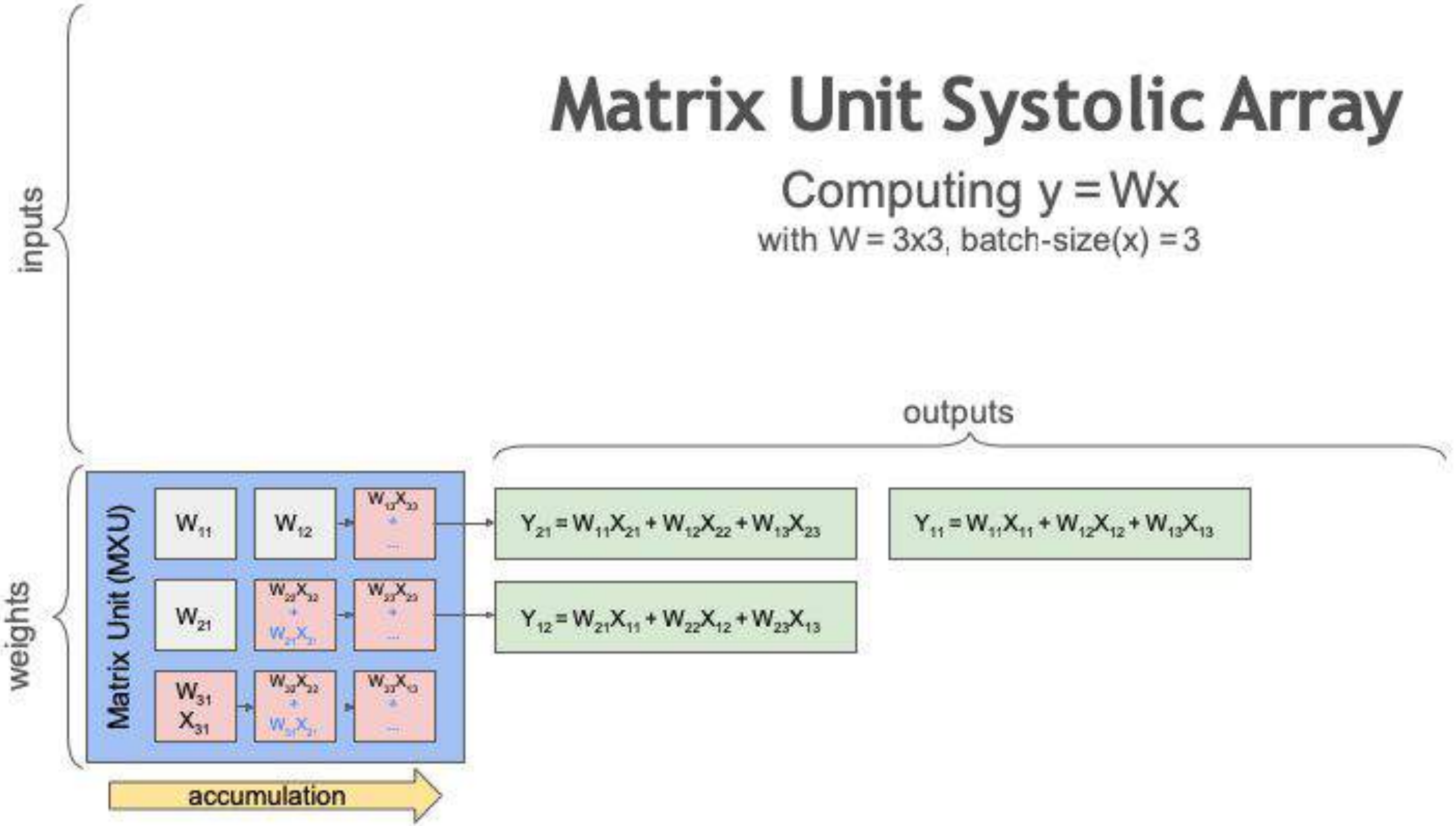
Computing y = Wx

with W = 3x3, batch-size(x) = 3

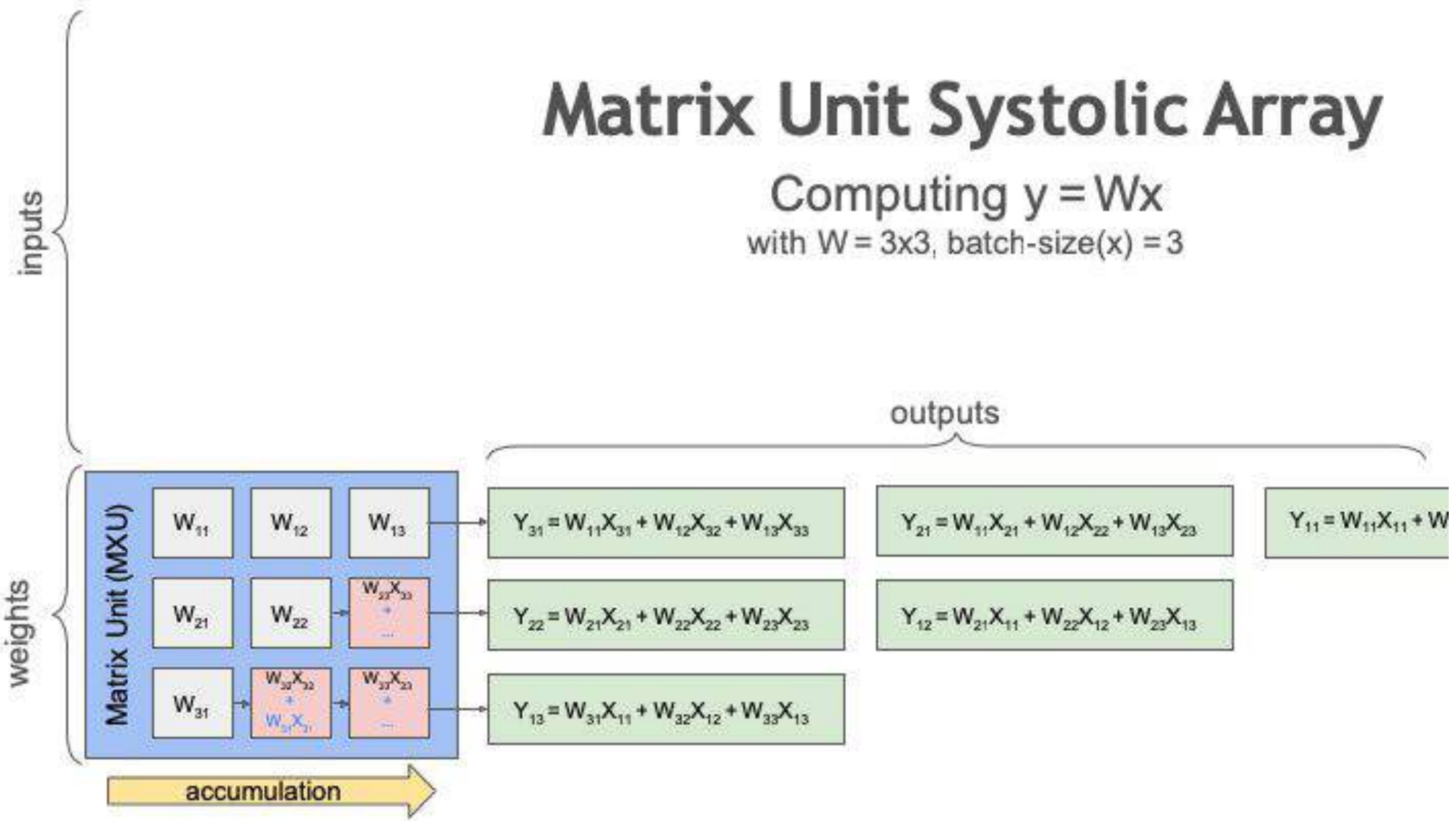
Neural Networks: Computation Example



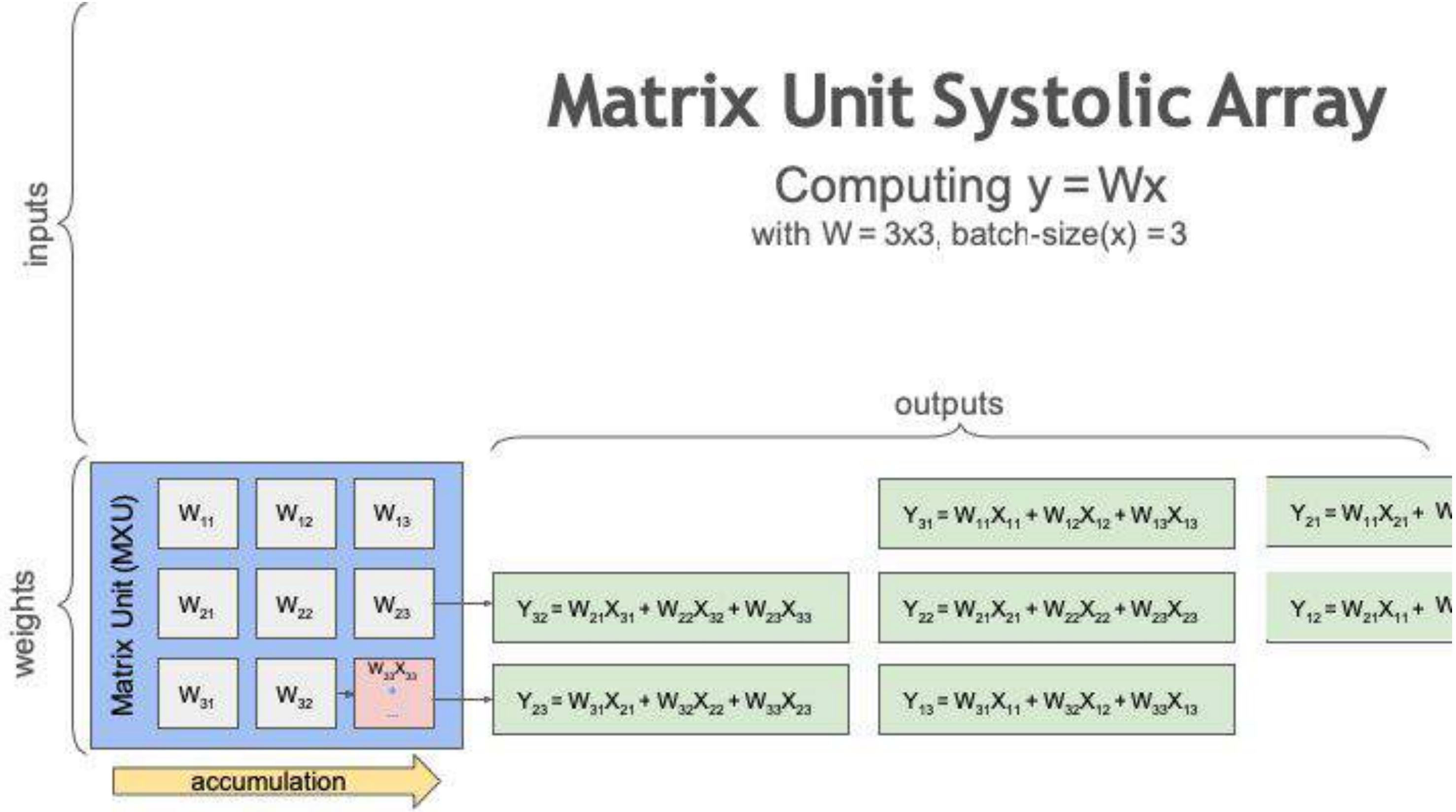
Neural Networks: Computation Example



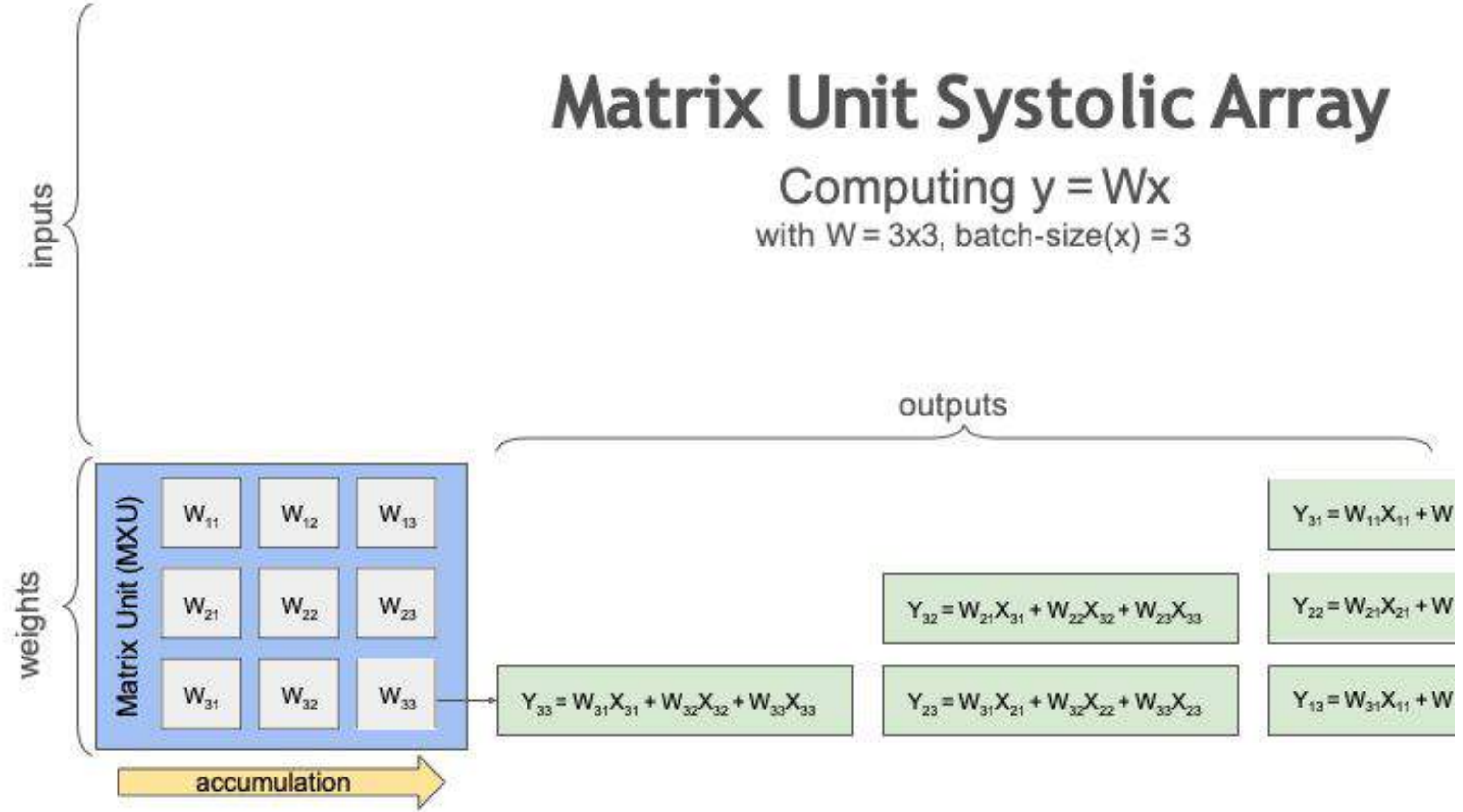
Neural Networks: Computation Example



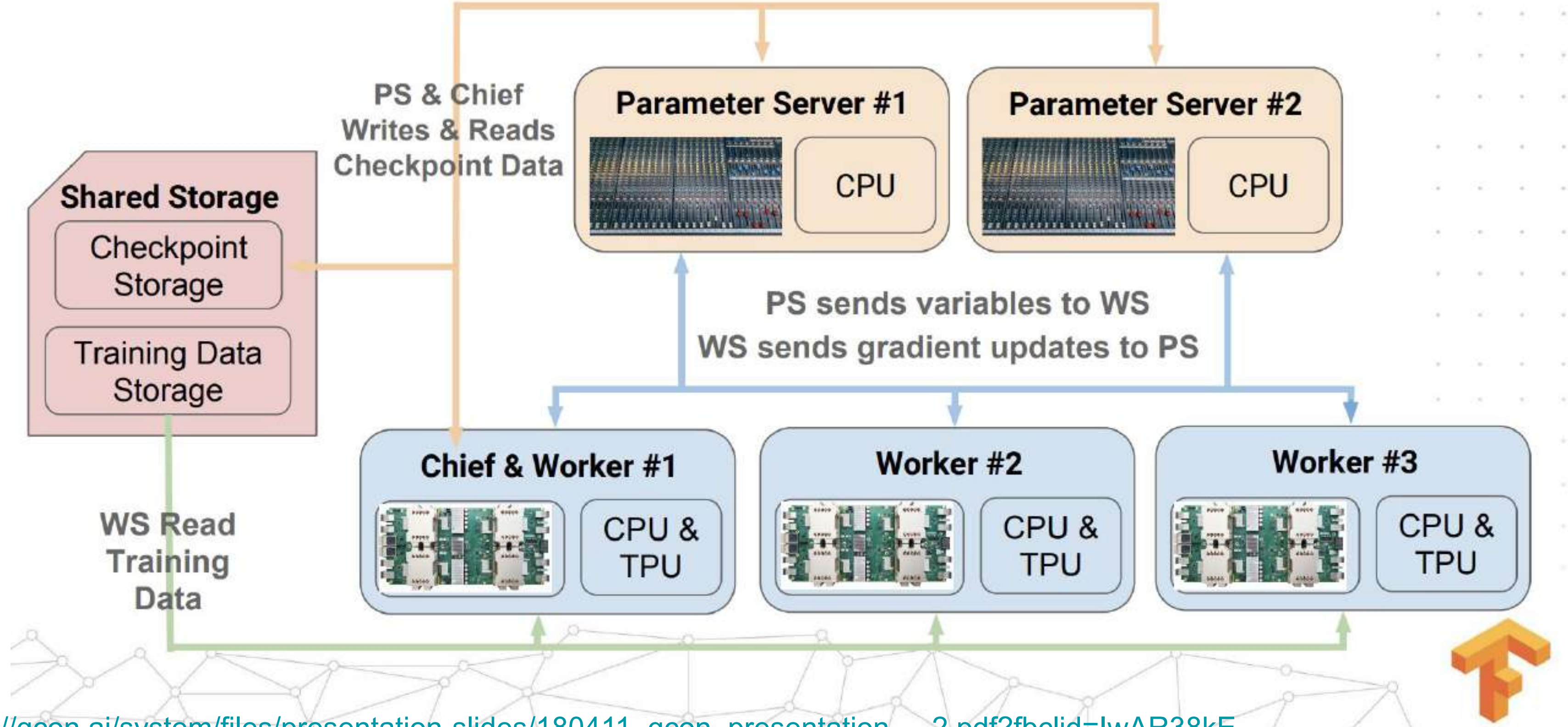
Neural Networks: Computation Example



Neural Networks: Computation Example



Neural Networks: Computation Example



https://qcon.ai/system/files/presentation-slides/180411_qcon_presentation_-_2.pdf?fbclid=IwAR38kESwm8e2NAhkj5JFqgz0F0VtnCpFyBp1HH5itsoSQlIYvkyYEwsc9uY

Learning Theory

- Let H be any finite hypothesis space. With probability 1 -δ a hypothesis $h \rightarrow H$ that is consistent with a training set of size m will have an error < ϵ on future examples if

$$m > \boxed{\frac{1}{\epsilon}}\left(\boxed{\ln(|H|)} + \boxed{\ln\frac{1}{\delta}}\right)$$

1. Expecting lower error increases sample complexity (i.e more examples needed for the guarantee)

2. If we have a larger hypothesis space, then we will make learning harder (i.e higher sample complexity)

3. If we want a higher confidence in the classifier we will produce, sample complexity will be higher.

Credit: http://cs229.stanford.edu/summer2020/cs229-notes4.pdf

VC Dimension

- Given a **hypothesis class** H over instance space X, we then define its Vapnik Chervonenkis dimension, written as $VC(H)$, to be the size of the largest finite subset of X that is shattered by H.
- In general, the VC dimension of an **n-dimensional** linear function is $n+1$

This term will decrease

This term may decrease

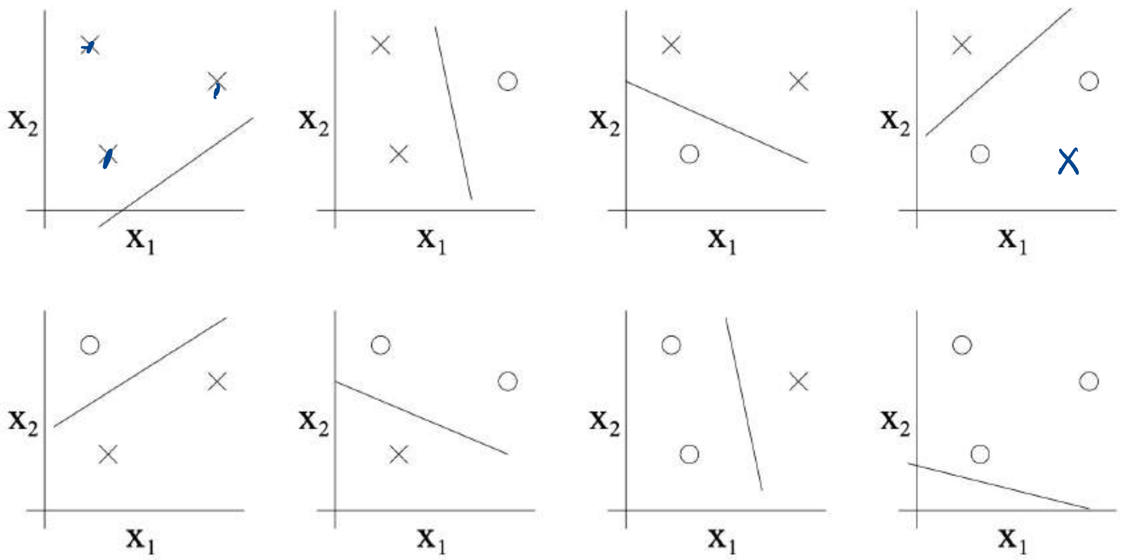
$$err_D(h) \le err_S(h) + \sqrt{\frac{VC(H)\left(\ln \frac{2m}{VC(H)} + 1\right) + \ln \frac{4}{\delta}}{m}}$$

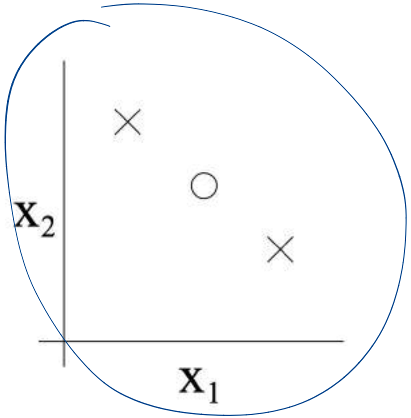
- Sample size for infinite H

$$m \ge \frac{1}{\varepsilon}\left(4\log_2\left(\frac{2}{\delta}\right) + 8 \cdot VC(H) \log_2\left(\frac{13}{\varepsilon}\right)\right)$$

VC Dimension of Half Space

- How to determine the set H of linear classifiers in two dimension has a VC(H)=3?

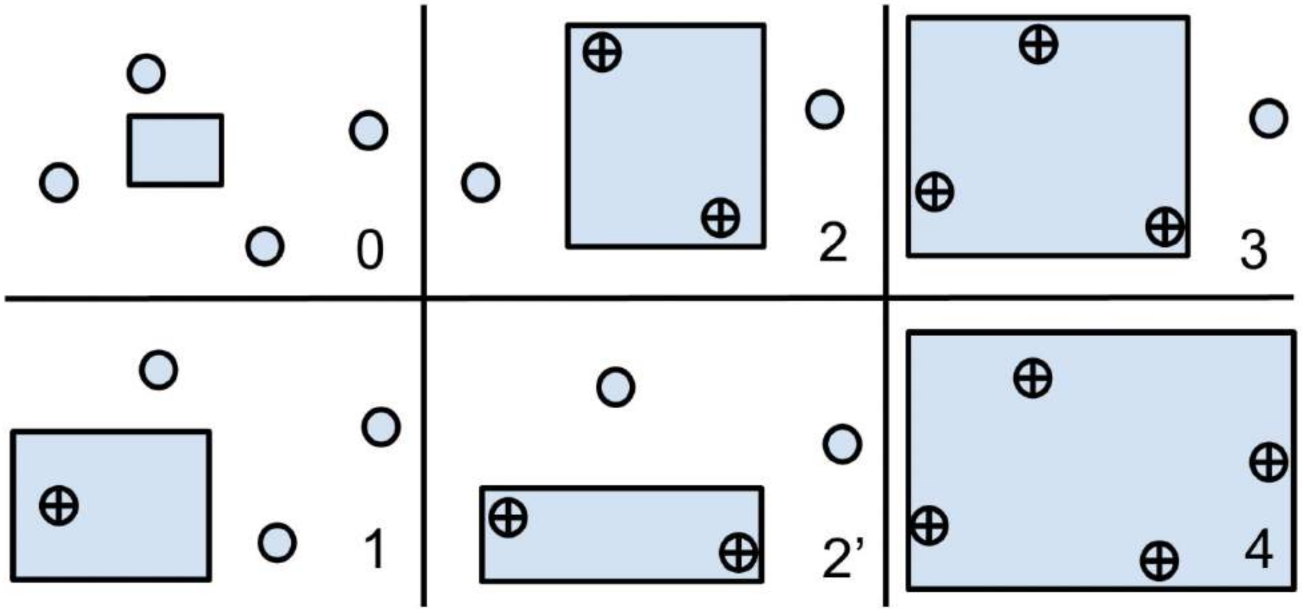




VC dimension of H here is 3 even though there may be sets of size 3 that it cannot shatter.

VC Dimension of Rectangles

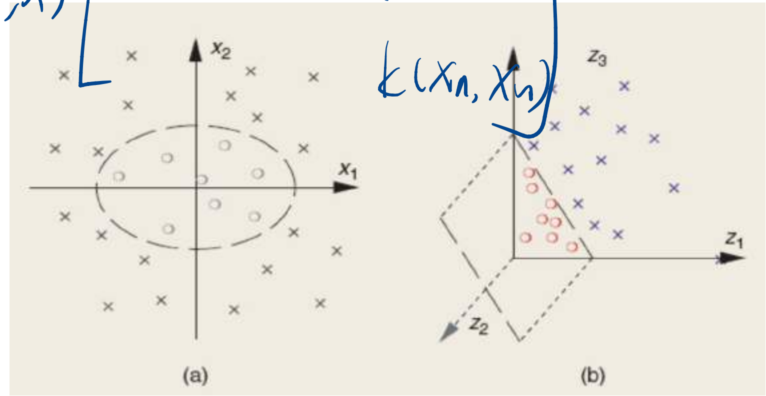
- What is the VC Dimension of Axis-aligned rectangles?



Credit: https://www.cs.princeton.edu/courses/archive/spring14/cos511/scribe_notes/0220.pdf

Kernels

$$K(X,X) = \begin{bmatrix} k(x_1,x_1) & & \\ & \ddots & \\ & & k(x_n,x_n) \end{bmatrix}$$



- Motivation: Transformed feature space

$\Phi(x)$

- Basic idea: Define K, called kernel, such that:

$$K: \underline{X \times X} \to \underline{\mathbb{R}} \qquad \Phi(x) \cdot \Phi(y) = \underline{K(x, y)}$$

which is often as a similarity measure.

- Benefit:
  - Efficiency: is often more efficient to compute than and the dot product.
  - Flexibility: can be chosen arbitrarily so long as the existence of is guaranteed (Mercer's condition).

Credit: https://cs.nyu.edu/~mohri/icml2011-tutorial/tutorial-icml2011-1.pdf

Polynomial Kernels

- **Definition:**

$d \geqslant 2$

$$\forall x, y \in \mathbb{R}^N, \;\; K(x, y) = (x \cdot y + c)^d, \quad c > 0.$$

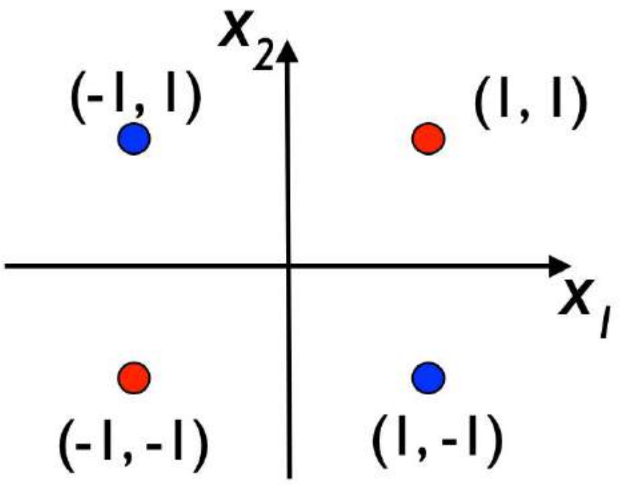
- **Example:** for $N = 2$ and $d = 2$,

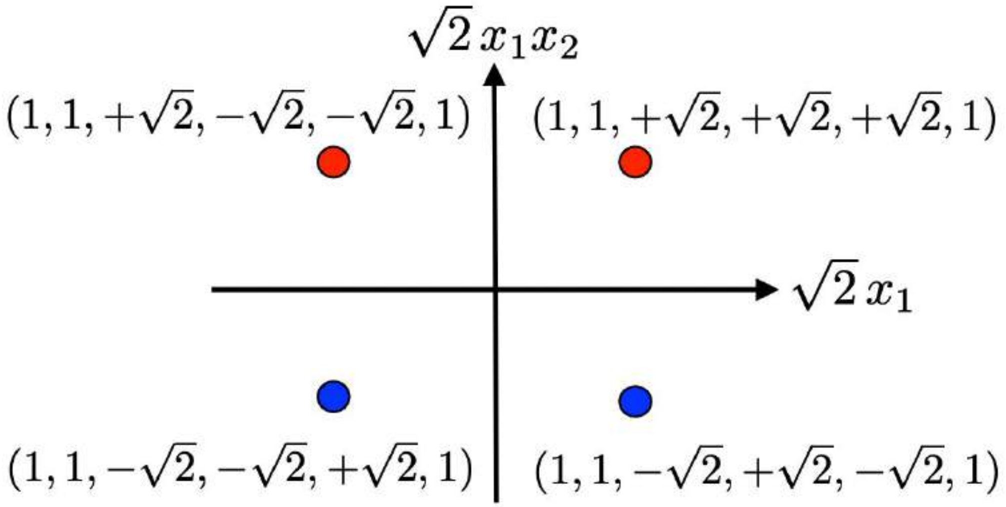
$$\boxed{K(x, y) = (x_1 y_1 + x_2 y_2 + c)^2}$$

$$= \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \\ \sqrt{2c}\, x_1 \\ \sqrt{2c}\, x_2 \\ c \end{bmatrix} \cdot \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}\, y_1 y_2 \\ \sqrt{2c}\, y_1 \\ \sqrt{2c}\, y_2 \\ c \end{bmatrix}.$$

$\underline{\phi(x)}$ $\underline{\phi(y)}$

Kernels: XOR Example





Linearly non-separable

Linearly separable by $x_1 x_2 = 0$.

Other Kernel Options

Gaussian kernels:

$$K(x,y) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right), \quad \sigma \neq 0.$$

Also known as "Radial Basis Function Kernel"

Sigmoid Kernels:

$$K(x,y) = \tanh(a(x \cdot y) + b), \quad a, b \geq 0.$$

Note: The RBF/Gaussian kernel as a projection into infinite dimensions, commonly used in kernel SVM.

$$K(x,x') = \exp\left(-(x-x')^2\right)$$

$$= \exp\left(-x^2\right) \exp\left(-x'^2\right) \underbrace{\sum_{k=0}^{\infty} \frac{2^k (x)^k (x')^k}{k!}}_{\exp(2xx')}$$

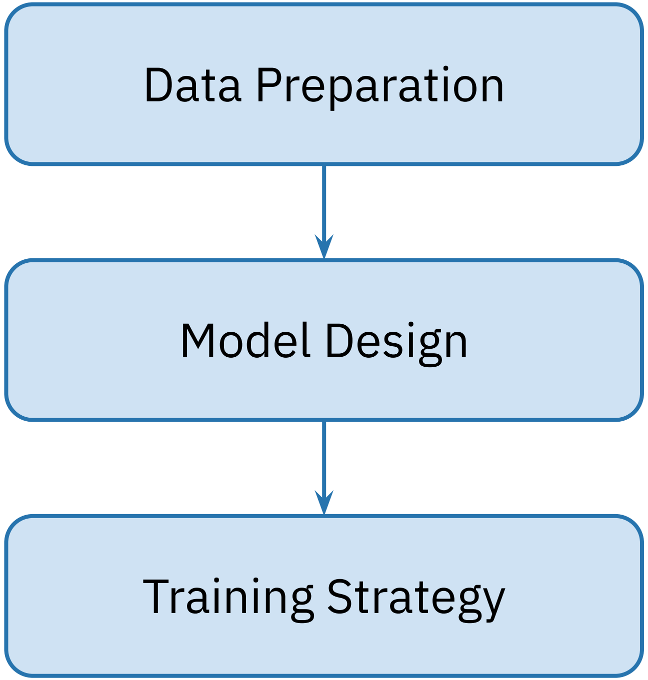
Taylor Expansion

Credit: http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf

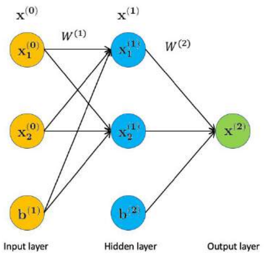
Programming Guide for PS3: PyTorch

- Important Concept Checklist
  - Tensors, Variable, Module
  - Autograd
  - Creating neural nets with provided modules: `torch.nn`
  - Training pipeline (loss, optimizer, etc): `torch.optim`
  - Util tools: Dataset
  - *(most important)* Search on official document or google
- A Not-so-short Tutorial: https://web.cs.ucdavis.edu/~yjlee/teaching/ecs289g-winter2018/Pytorch_Tutorial.pdf → Details and demo code in another slides
- Youtube: https://www.youtube.com/playlist?list=PLlMkM4tgfjnJ3I-dbhO9JTw7gNty6o_2m

PyTorch Project Pipeline



Use PyTorch to check your gradient calculation




```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.l1 = nn.Linear(2, 2, bias=True)
        self.l2 = nn.Linear(2, 1, bias=True)

        self.l1.weight.data = torch.Tensor([[0.15, 0.2], [0.25, 0.3]])
        self.l2.weight.data = torch.Tensor([[0.4, 0.45]])
        self.l1.bias.data = torch.Tensor([0.35, 0.35])
        self.l2.bias.data = torch.Tensor([0.6])

    def forward(self, x0):
        z1 = self.l1(x0)
        x1 = torch.sigmoid(z1)
        z2 = self.l2(x1)
        x2 = torch.sigmoid(z2)
        print("z1:", z1)
        print("x1:", x1)
        print("z2:", z2)
        print("x2:", x2)
        return x2

    def loss(self, x2, y):
        l = nn.MSELoss()
        return 0.5 * l(x2, y)
```

```python
x = torch.Tensor([0.05, 0.1])
y = torch.Tensor([0.01])
net = Net()
y_hat = net(x)
loss = net.loss(y_hat, y)
print(loss)
loss.backward()
```

```
z1: tensor([0.3775, 0.3925], grad_fn=<AddBackward0>)
x1: tensor([0.5933, 0.5969], grad_fn=<SigmoidBackward>)
z2: tensor([1.1059], grad_fn=<AddBackward0>)
x2: tensor([0.7514], grad_fn=<SigmoidBackward>)
tensor(0.2748, grad_fn=<MulBackward0>)
```

```python
print("d[W1]", list(net.l1.parameters())[0].grad)
print("d[b1]", list(net.l1.parameters())[1].grad)
print("d[W2]", list(net.l2.parameters())[0].grad)
print("d[b2]", list(net.l2.parameters())[1].grad)
```

```
d[W1] tensor([[0.0007, 0.0013],
        [0.0007, 0.0015]])
d[b1] tensor([0.0134, 0.0150])
d[W2] tensor([[0.0822, 0.0827]])
d[b2] tensor([0.1385])
```

Colab Link: https://colab.research.google.com/drive/1FHo_mkFaTatKgpBw5VRRVUSBJzoMt_U8?usp=sharing

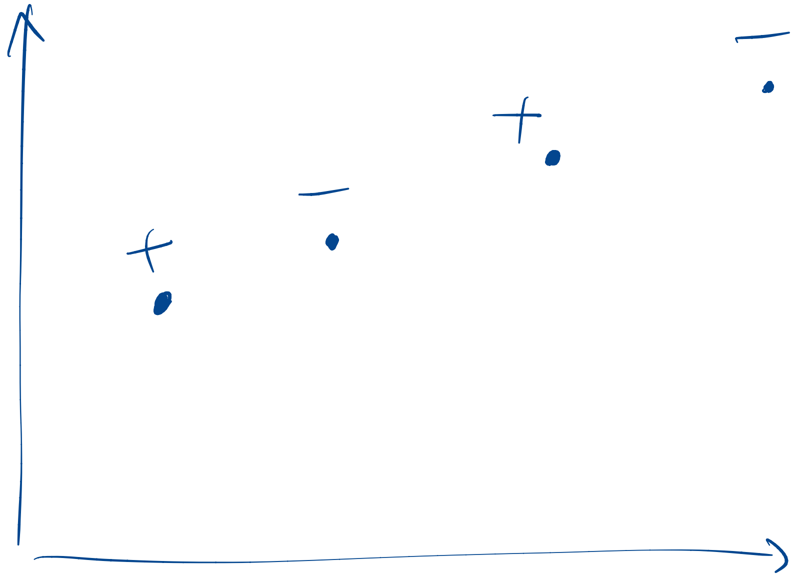
Thank you!

Q & A

Whiteboard

- Content

16



Whiteboard

- Content

UCLA Samueli Computer Science

CS M146 Discussion: Week 8

Ensemble Method, Multi-Class Classification, ML Evaluation

Junheng Hao

Friday, 02/26/2021

Roadmap

- Announcement
- Ensemble Method
- Multi-Class Classification
- ML Evaluation

Announcements

- **5:00 pm PST, Feb 26 (Friday):** Weekly Quiz 8 released on Gradescope.
- **11:59 pm PST, Feb 28 (Sunday):** Weekly Quiz 8 closed on Gradescope!
  - Start the quiz before **11:00 pm PST, Feb 28** to have the full 60-minute time
- **Problem set 3** released on CCLE, submission on Gradescope.
  - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
  - You need to submit code and the results required by the problem set
  - Due on **Today 11:59pm PST, Feb 26 (Friday)!**

Late Submission of PS will NOT be accepted!

About Quiz 8

- Quiz release date and time: **Feb 26, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Feb 28, 2021 (Sunday) 11:59 PM PST**
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week 8 Quiz" on GradeScope.
- Topics: **Ensemble Method, Multi-Class Classification, ML Evaluations**
- Question Types
    - True/false, multiple choices
    - Some questions may include several subquestions.
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.

Quiz 7 Review: Kernel SVM

Q6 Kernel SVM

2 Points

We can introduce non-linearity to SVM using the kernel trick. Instead of searching for a hyperplane $\mathbf{w}^T\mathbf{x} + b$ that maximizes the margin, we are looking for $\mathbf{w}^T\phi(\mathbf{x}) + b$ where ϕ is the non-linear basis function.

Which **one** of the following statements is **wrong** about kernel SVM?

- (selected) We can learn the optimal value of the weights $\mathbf{w}$ using only the kernel function. ✗
- We can predict the label of a new sample using only the kernel function.
- If we apply kernel functions, non-separable data may be separable.
- A valid kernel function should have a positive-semidefinite kernel matrix.

	Linear SVM $\boldsymbol{w}^\top \boldsymbol{x} + b$	Kernel SVM $\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_n) + b$
Weight parameter	$\boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{x}_n$	$\boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$
Predicting new data	$\mathrm{SIGN}(\sum_n y_n \alpha_n (\boldsymbol{x}_n^T \boldsymbol{x}) + b)$	$\mathrm{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$

UCLA Engineer Change.

Quiz 7 Review: Kernel SVM

Q4 SVM on non-separable data

2 Points

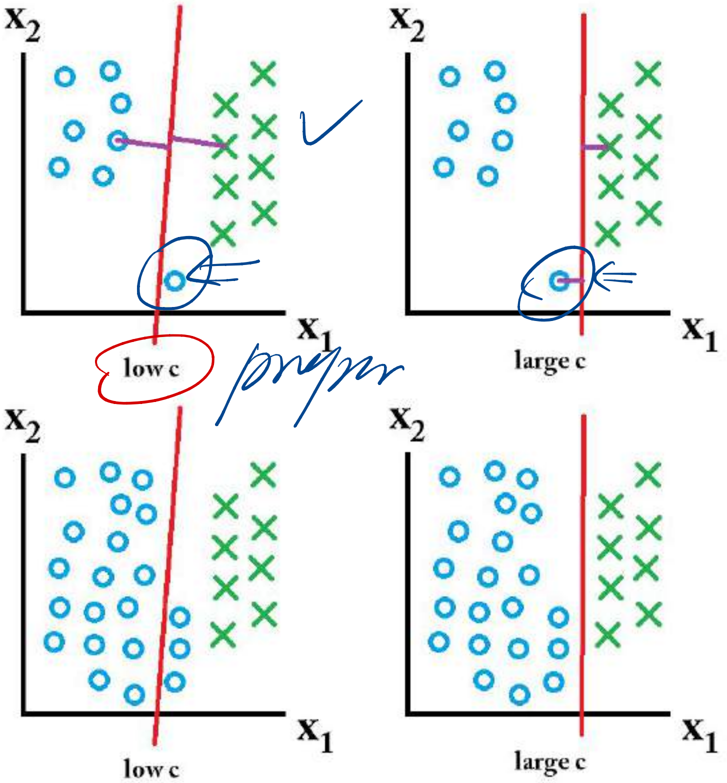
Which **one** of the following statements is **wrong** about SVM applied to non-separable data (soft-margin SVM)?

- ○ The use of slack variables permits samples to be misclassified.
- ○ As we increase the value of the hyperparameter C, the margin of the learned hyperplane decreases. → overfit ✓
- ○ As we decrease the value of the hyperparameter C, the training accuracy of the learned SVM may decrease. → underfit ✓
- ◉ During training, we do not only minimize $\frac{1}{2}||\omega||_2^2$, but also minimize $\sum(1-\xi_n)$ so that the value of slack variable can be controlled.

$\Sigma \xi_n$

SVM: Understanding C

- The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.
- For large values of C, the optimization will choose **a smaller-margin hyperplane** if that hyperplane does a better job of getting all the training points classified correctly.
- Conversely, a very small value of C will cause the optimizer to look for **a larger-margin separating hyperplane**, even if that hyperplane misclassified more points.



Reference: https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

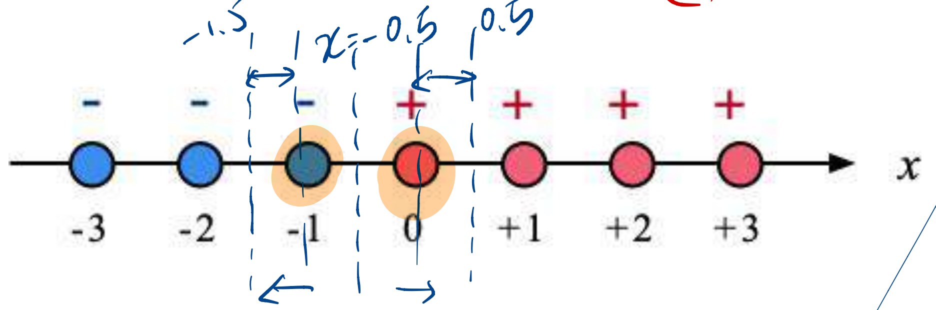
Additional: SVM on 1-dim data

0

inf

$$\min_{w,b} \quad \frac{1}{2}w^T w + \sum_{i=1}^{m} \epsilon_i \cdot C$$

subject to: $y_i\left(w^T x_i + b\right) \geq 1 - \epsilon_i, \epsilon_i \geq 0, \forall i$



- What if C is super large, i.e. C→ +inf? #SV = 2 (-1, 0) γ = 1
- What is C is super small, i.e. C→ 0 (a very small positive number, such as 0.0001)?

#SV = 7 γ = → ∞

Additional: SVM on 1-dim data (code)

Colab link:
https://colab.research.google.com/drive/1Ru_gN8UikD_fGY3DfarHTfXQQDg4b-D4?usp=sharing

```
[1] from sklearn import svm
    import numpy as np
```

```
[2] X = [[-3], [-2], [-1], [0], [1], [2], [3]]
    y = [0, 0, 0, 1, 1, 1, 1]
    X, y = np.array(X), np.array(y)
```

```
[3] def print_svm_decision(X, y, C):
      clf = svm.LinearSVC(C=C)
      clf.fit(X,y)
      decision_function = clf.decision_function(X)
      support_vector_indices = np.where( (2*y-1)*decision_function <= 1)[0]
      print("Support Vectors are: ", support_vector_indices)
      print("Predicting y=wx+b:", decision_function)
```

```
print_svm_decision(X, y, C=1)

Support Vectors are:  [2 3]
Predicting y=wx+b: [-2.36363017 -1.45454339 -0.54545661  0.36363017  1.27271694  2.18180372
  3.0908905 ]
```

```
[5] print_svm_decision(X, y, C=0.5)

    Support Vectors are:  [2 3 4]
    Predicting y=wx+b: [-1.75000807 -1.08333882 -0.41666956  0.24999969  0.91666894  1.5833382
      2.25000745]
```

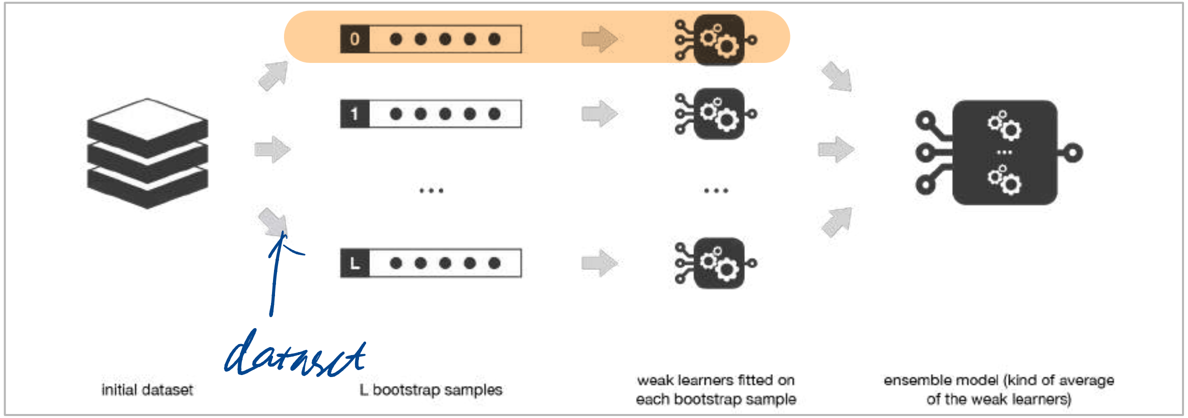
```
[6] print_svm_decision(X, y, C=0.1)

    Support Vectors are:  [1 2 3 4 5]
    Predicting y=wx+b: [-1.09999783 -0.69999856 -0.29999929  0.09999998  0.49999925  0.89999852
      1.29999779]
```

Case: C is suffciently small and every vector is within the margin, i.e. every data point is a support vector.

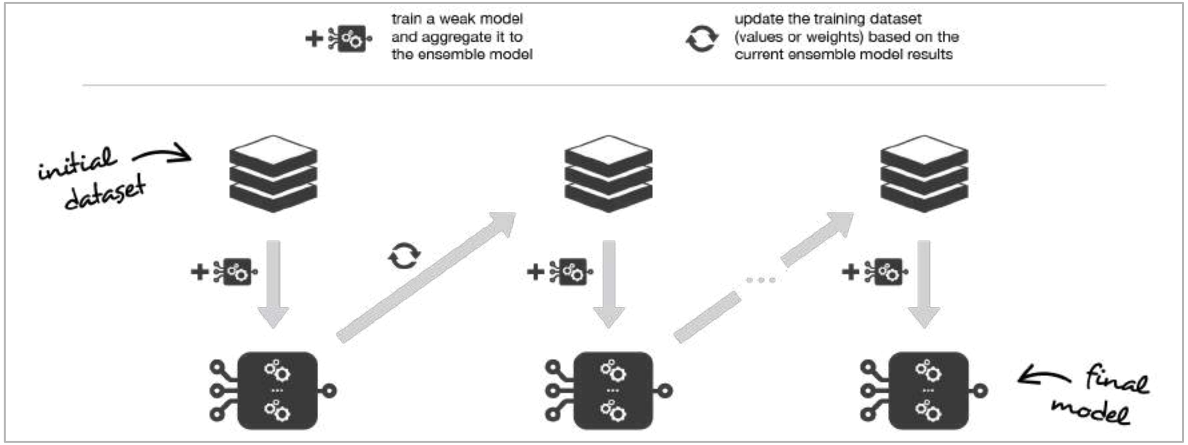
```
[7] print_svm_decision(X, y, C=0.01)

    Support Vectors are:  [0 1 2 3 4 5 6]
    Predicting y=wx+b: [-0.44399465 -0.29014848 -0.13630231  0.01754386  0.17139003  0.3252362
      0.47908237]
```

Ensemble: Bagging and Boosting

Bagging
"Parallel Learner"



Boosting
"Sequential Learner"

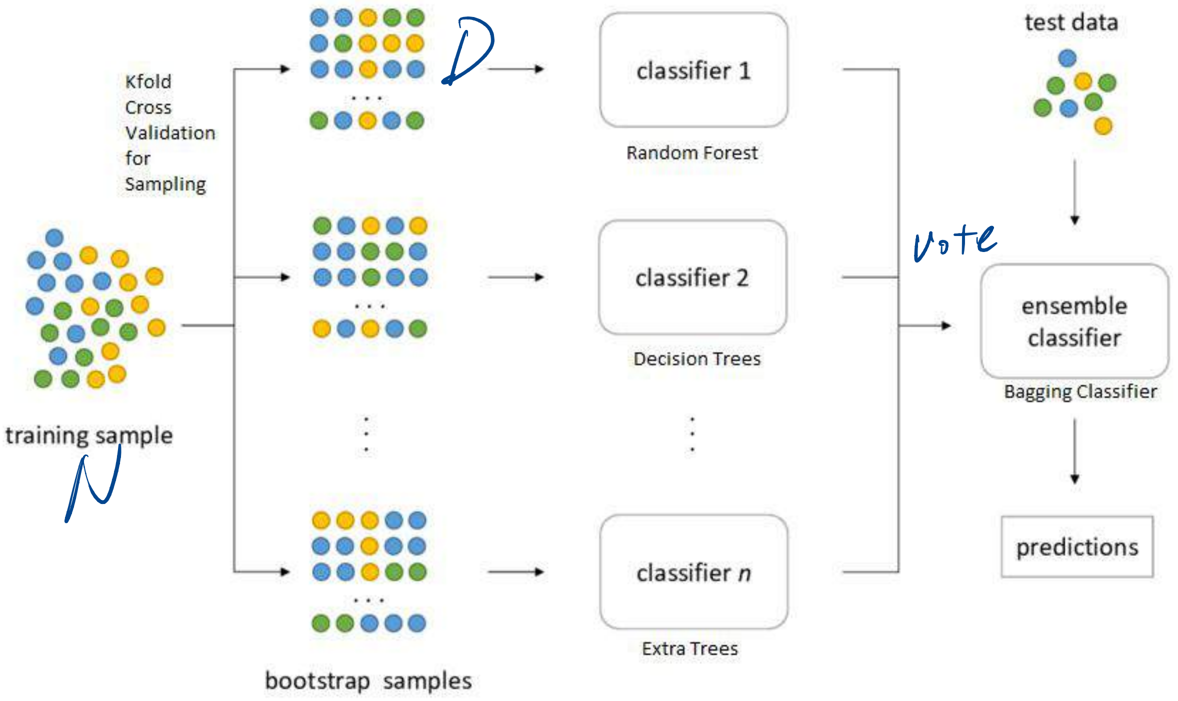


Ensemble: Bagging (Bootstrap Aggregation)

- **“Multiple”** dataset and **multiple** classifier

m

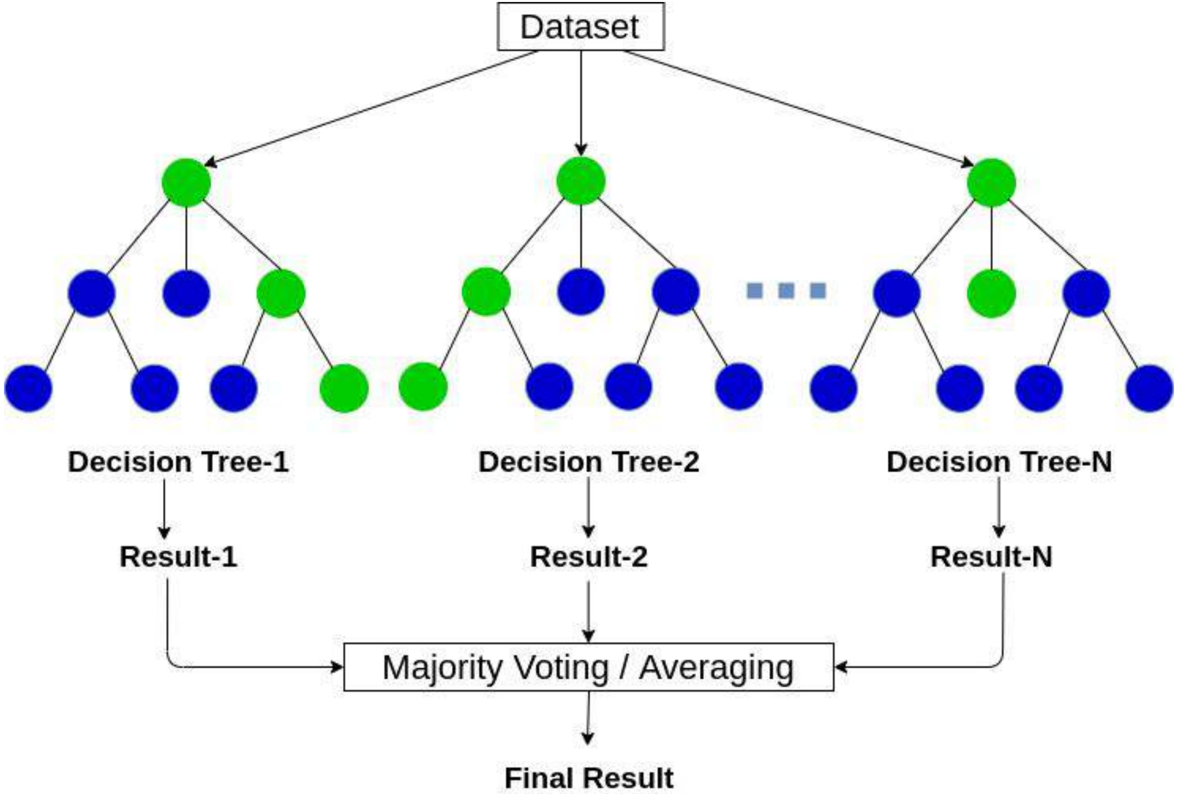
m



Bagging Classifier Process Flow

Ensemble: Bagging (Random Forest)

- **Single:** Decision Tree → **Bagging:** Random Forest



Ensemble: Boosting (Adaboost)

- Given: N samples $\{\boldsymbol{x}_n, y_n\}$, where $y_n \in \{+1, -1\}$, and some way of constructing weak (or base) classifiers
- Initialize weights $w_1(n) = \frac{1}{N}$ for every training sample
- For $t = 1$ to T
  1. Train a weak classifier $h_t(\boldsymbol{x})$ using current weights $w_t(n)$, by minimizing the weighted classification error

$$\epsilon_t = \sum_n w_t(n) \mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)]$$

  2. Compute contribution for this classifier: $\beta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  3. Update weights on training points

$$w_{t+1}(n) \propto w_t(n) e^{-\beta_t y_n h_t(\boldsymbol{x}_n)}$$

  and normalize them such that $\sum_n w_{t+1}(n) = 1$
- Output the final classifier

$$h[\boldsymbol{x}] = \text{sign}\left[\sum_{t=1}^{T} \beta_t h_t(\boldsymbol{x})\right]$$

1. Calculating error

$\epsilon_t \uparrow$

2. Calculating classifier weights of t

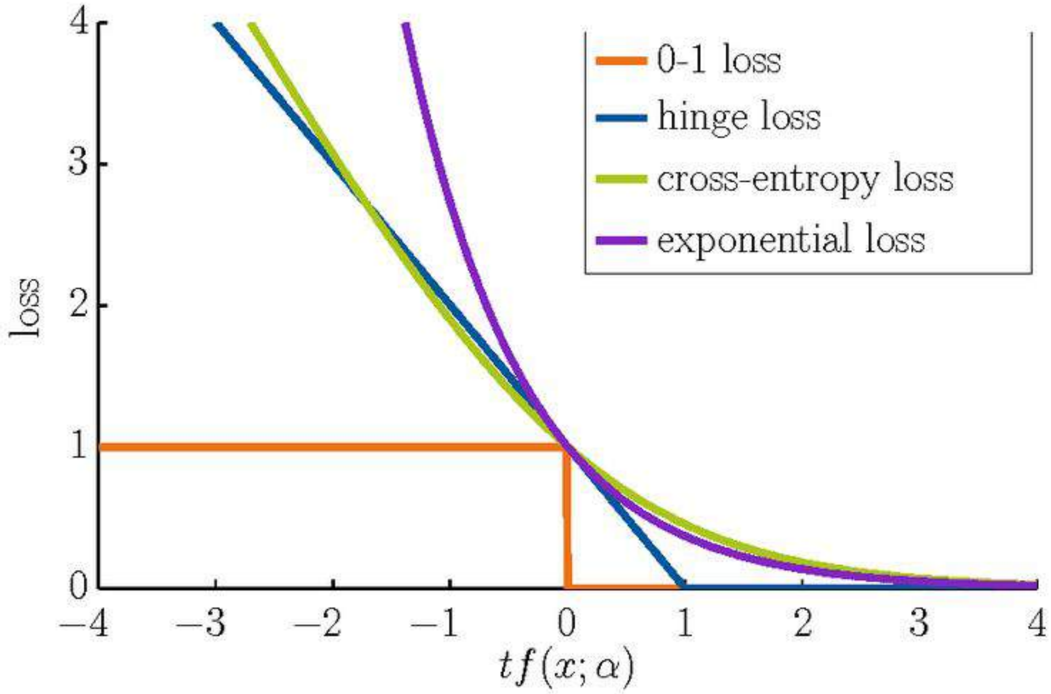
3. Reweighting training points

ensemble

~~Bagging~~ Classifier

Ensemble: Adaboost Loss Function

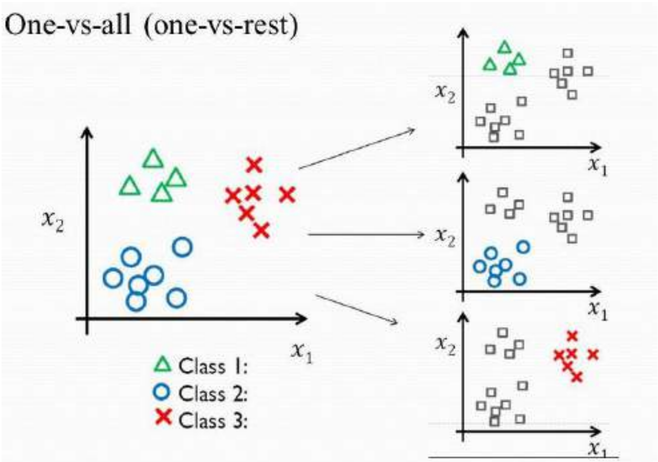
- **Exponential loss**, instead of 0/1 loss



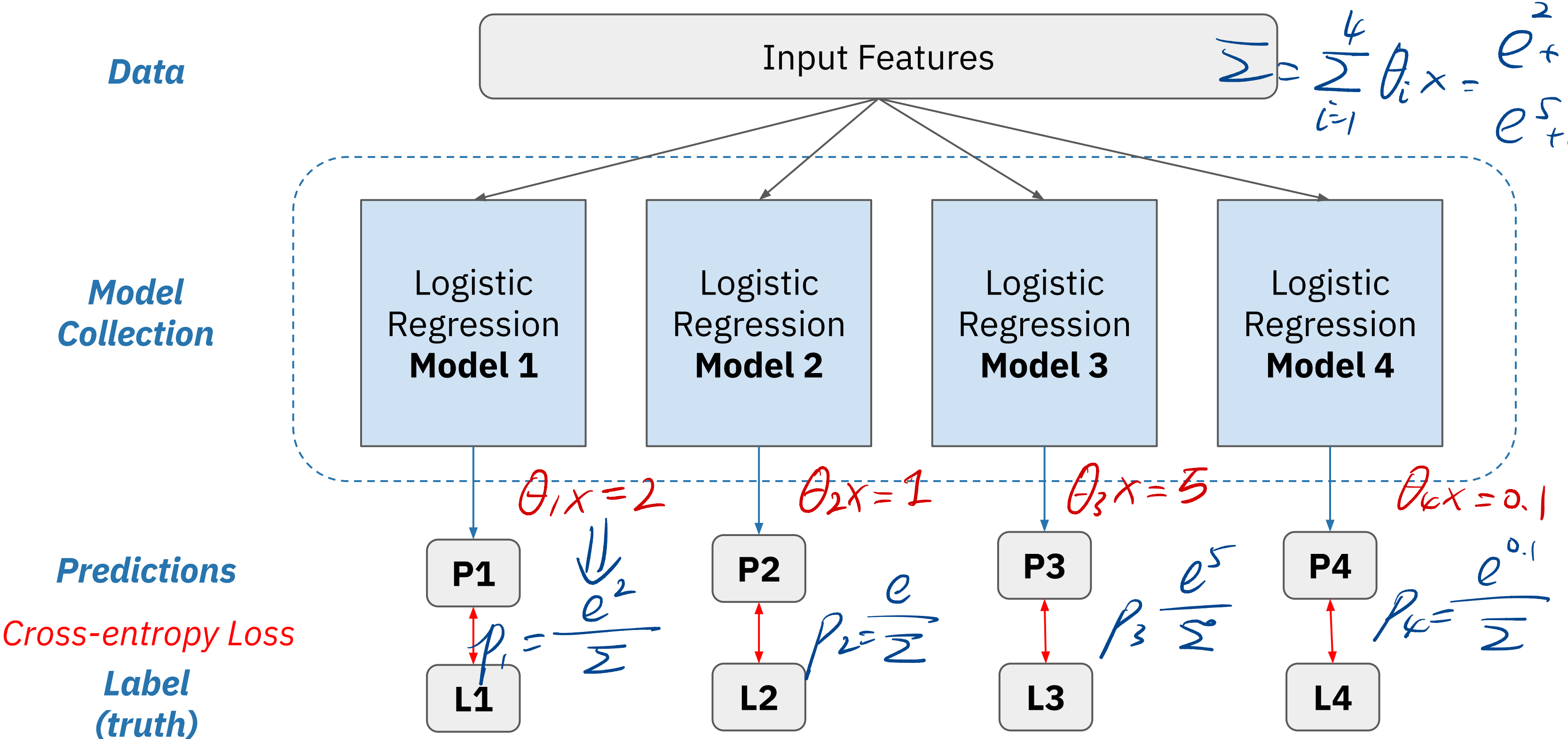
Multi-Class Classification: Two modes

Given **C** classes and **N** data points each class:

Comparison	**One-vs-one**	**One-vs-rest (all)**
# Binary Classifiers	$\frac{1}{2}C(C-1)$	C
# Training Data	$2N$ $O(N)$	$C \cdot N$
Pros		
Cons		



Logistic Regression: Multiclass Case



$$\Sigma = \sum_{i=1}^{4} \theta_i x = e^2 + e + e^5 + e^{0.1}$$

$\theta_1 x = 2$ $\Downarrow$ $\quad \theta_2 x = 1 \quad \theta_3 x = 5 \quad \theta_4 x = 0.1$

$p_1 = \frac{e^2}{\Sigma} \quad p_2 = \frac{e}{\Sigma} \quad p_3 = \frac{e^5}{\Sigma} \quad p_4 = \frac{e^{0.1}}{\Sigma}$

$$p_1 + p_2 + p_3 + p_4 = 1$$

Multinomial Logistic Regression

0 $\begin{matrix}3\\1\\2\end{matrix}$

#batch size

Model

For each class C_k, we have a parameter vector $\boldsymbol{\theta}_k$ and model the probability of class C_k as

Model: $\sum_{k=1}^{C} p(y=C_k) = 1$

$$p(y = C_k|\boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \frac{e^{\boldsymbol{\theta}_k^{\mathrm{T}}\boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{\theta}_{k'}^{\mathrm{T}}\boldsymbol{x}}}$$

Ypred = (# × 3)

batch_y

← This is called *softmax*

Decision boundary: assign $\boldsymbol{x}$ with the label that is the maximum of

k

$$\arg\max_k P(y = C_k|\boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) \rightarrow \arg\max_k \boldsymbol{\theta}_k^{\mathrm{T}}\boldsymbol{x}$$

y_{nk} n $\begin{bmatrix} 0 & 1 & 0 & 0 \\ & & & \\ & & & \end{bmatrix}$

Likelihood

$$\sum_n \log P(y_n|\boldsymbol{x}_n; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \sum_n \log \prod_{k=1}^{K} P(y = C_k|\boldsymbol{x}_n; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)^{y_{nk}}$$

$$= \sum_n \sum_k y_{nk} \log P(y = C_k|\boldsymbol{x}_n; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$$

MLR: Cost Function and Optimization

Likelihood

$$\sum_n \log P(y_n|\boldsymbol{x}_n;\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_K) = \sum_n \log \prod_{k=1}^{K} P(y=C_k|\boldsymbol{x}_n;\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_K)^{y_{nk}}$$
$$= \sum_n \sum_k y_{nk} \log P(y=C_k|\boldsymbol{x}_n;\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_K)$$

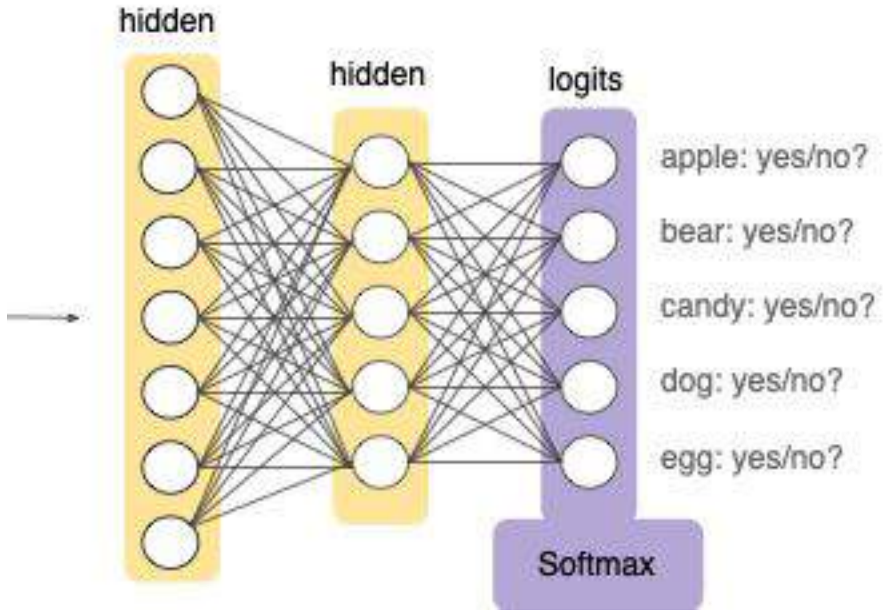
Cost Function

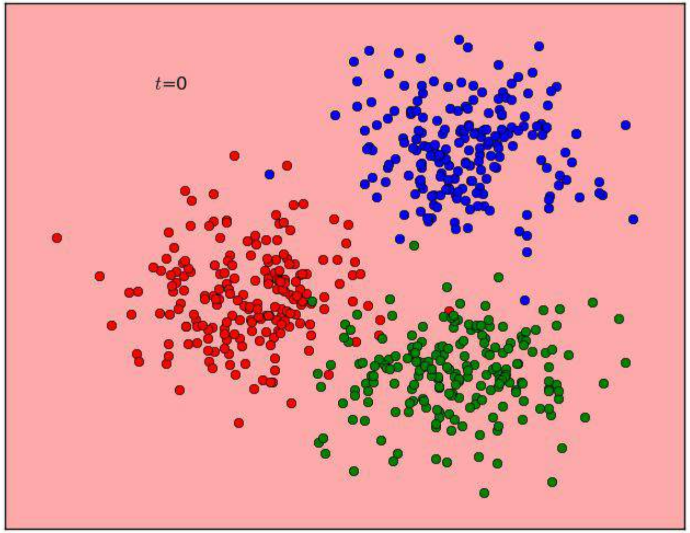
$$\boxed{\begin{aligned} J(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\ldots,\boldsymbol{\theta}_K) &= -\sum_n \sum_k y_{nk} \log P(y=C_k|\boldsymbol{x}_n;\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\ldots,\boldsymbol{\theta}_K) \\ &= -\sum_n \sum_k y_{nk} \log\left(\frac{e^{\boldsymbol{\theta}_k^{\mathrm{T}}\boldsymbol{x}}}{\sum_{k'} e^{\boldsymbol{\theta}_{k'}^{\mathrm{T}}\boldsymbol{x}}}\right) \\ &= -\sum_n \sum_k y_{nk}\boldsymbol{\theta}_k^{\mathrm{T}}\boldsymbol{x} - y_{nk}\log\left(\sum_{k'} e^{\boldsymbol{\theta}_{k'}^{\mathrm{T}}\boldsymbol{x}}\right) \end{aligned}}$$

Optimization

Convex → SGD

Multiclass Classification in Neural Nets





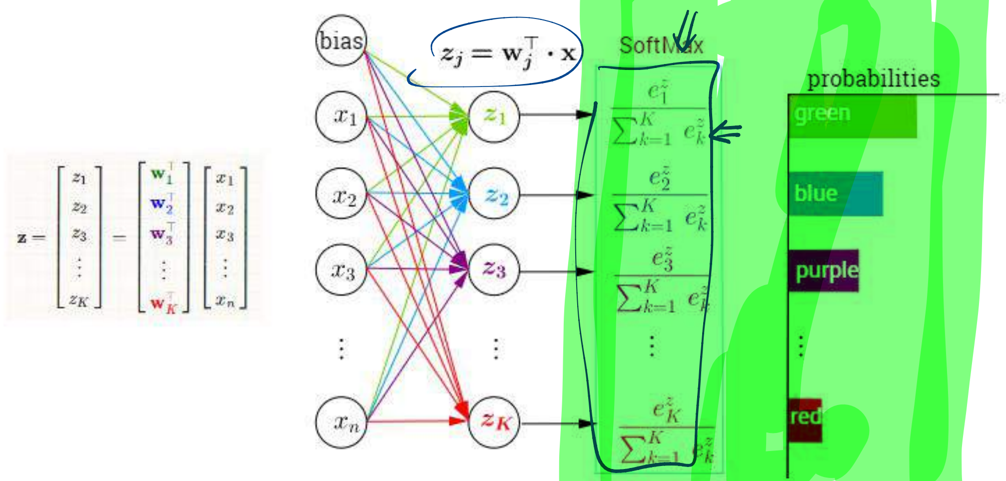
5 separate **binary classifiers**

Key: **sharing the same hidden layers** with **different weights at the end**

https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/one-vs-all
http://www.briandolhansky.com/blog/2013/9/23/artificial-neural-nets-linear-multiclass-part-3

Softmax Layer in Neural Nets

Multi-Class Classification with NN and SoftMax Function



$\Sigma = 1$

$$p(y = j|\mathbf{x}) = \frac{e^{(\mathbf{w}_j^T \mathbf{x} + b_j)}}{\sum_{k \in K} e^{(\mathbf{w}_k^T \mathbf{x} + b_k)}}$$

→ Implementation in PyTorch?

torch.nn.CrossEntropyLoss( )

$Y_{pred} \in R^{\#n \times 3}$

$0 \to (C-1)$

$Y_{targ} \in R^{\#n}$

$[\quad]$

Credit:
https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

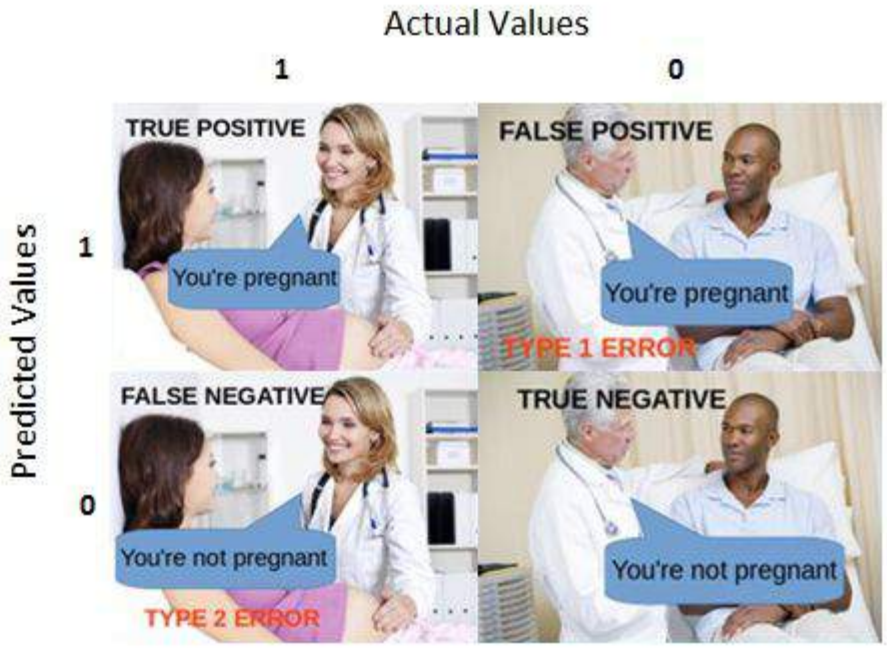
Evaluation: Binary Classifier

- Diagnostic testing table

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{TPR}{FPR}$	Diagnostic odds ratio (DOR) = $\frac{LR+}{LR-}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{FNR}{TNR}$		

Credit: https://en.wikipedia.org/wiki/F-score

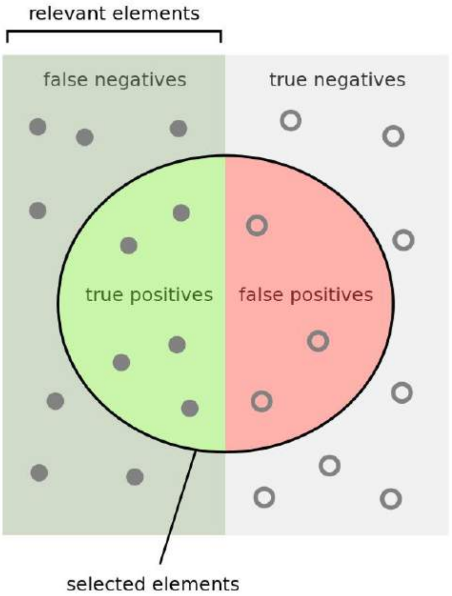
Evaluation: TP, FP, FN, TN & Two Types of Errors

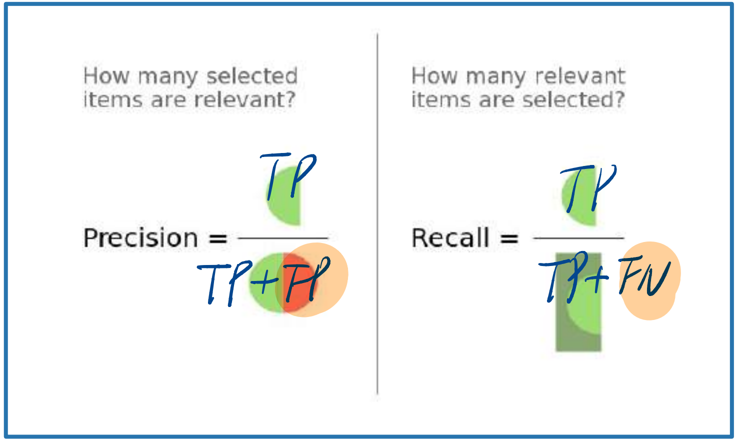


Credit: https://devopedia.org/confusion-matrix

Evaluation: Binary Classifier

- Precision and recall





Credit: https://en.wikipedia.org/wiki/F-score

Evaluation: Example

- Calculation from confusion matrix

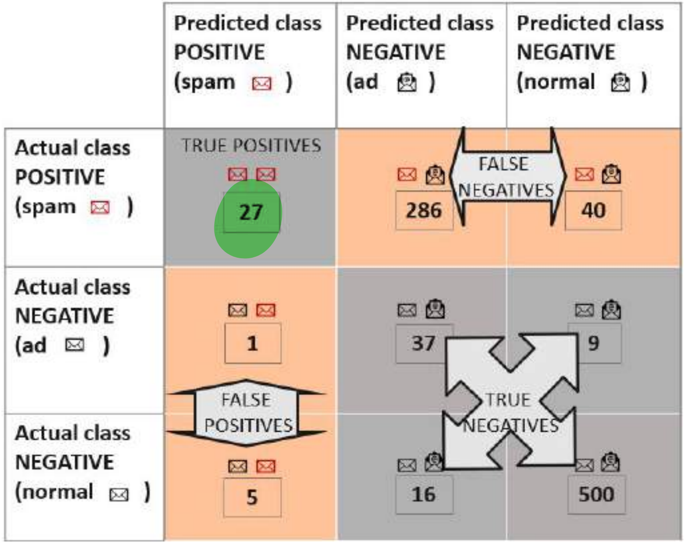
Predicted Class

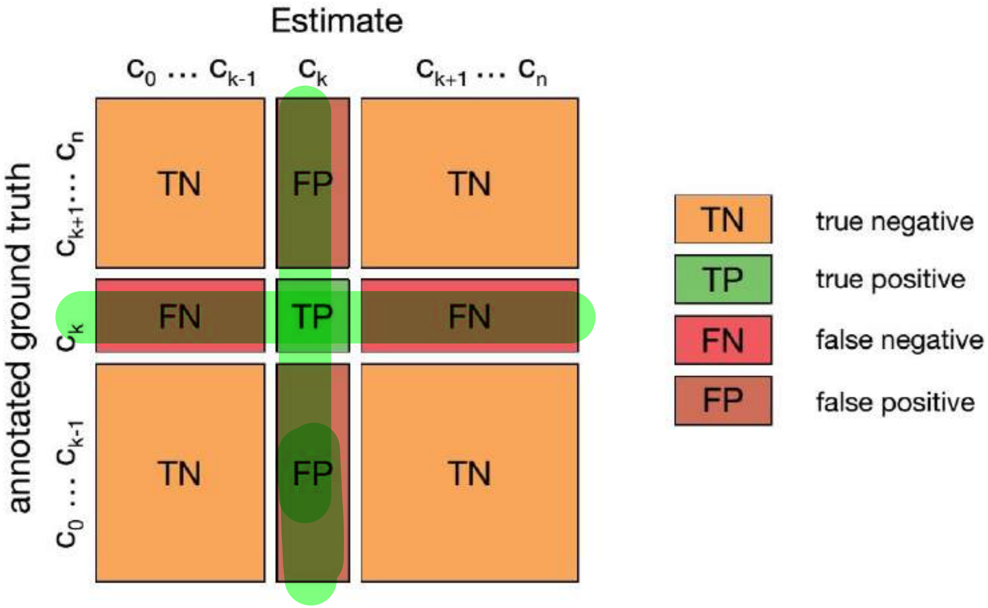
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

	Predicted class POSITIVE (spam)	Predicted class NEGATIVE (normal)
Actual class POSITIVE (spam)	TRUE POSITIVE (TP) 320	FALSE NEGATIVE (FN) 43
Actual class NEGATIVE (normal)	FALSE POSITIVE (FP) 20	TRUE NEGATIVE (TN) 538

Credit: https://towardsdatascience.com/confusion-matrix-and-class-statistics-68b79f4f510b

Evaluation: Multi-class Confusion Matrix

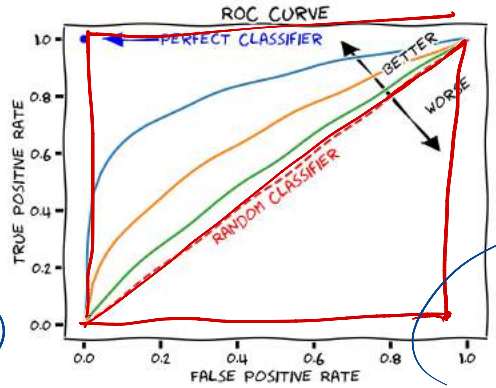




Credit: https://devopedia.org/confusion-matrix

Evaluation: Other Metrics & Curves

- ROC [Wiki]
- Area under the ROC Curve: AUC/AUROC
- A random classifier has AUROC = ½.
- A perfect classifier has AUROC = 1.
- Others: Precision-Recall Curve and AUPR



$t' = 0.975$

$TPR = 0.4 = \frac{2}{5}$

$FPR = 0$

In-class Exercise: Building ROC Curves

$\mathbb{1}[p > t]$

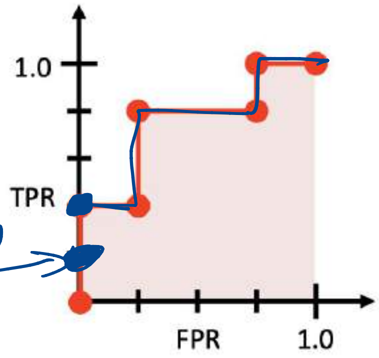
0 —— 1

$t = 0.985 \to$ $TPR = 0.2$, $FPR = 0$

Instance	Confidence in positive	Actual class
9	0.99	1
7	0.98	1
1	0.72	0
2	0.70	1
6	0.65	1
10	0.51	0
3	0.39	0
5	0.24	1
4	0.11	0
8	0.01	0

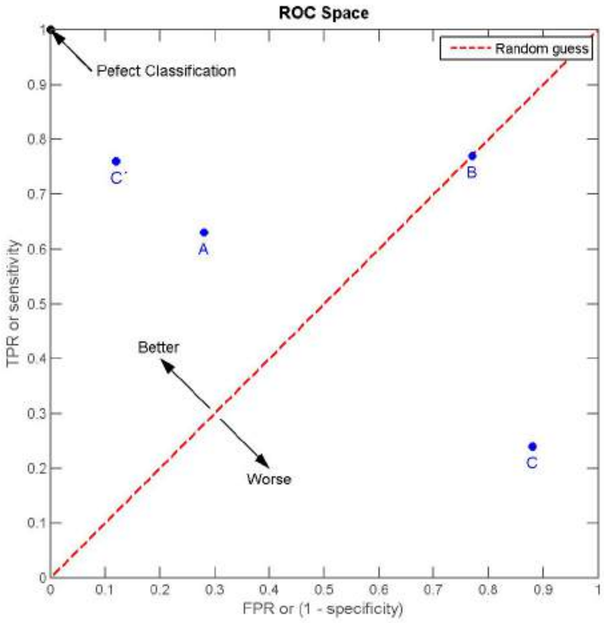
1 | 0

1 1 | 0



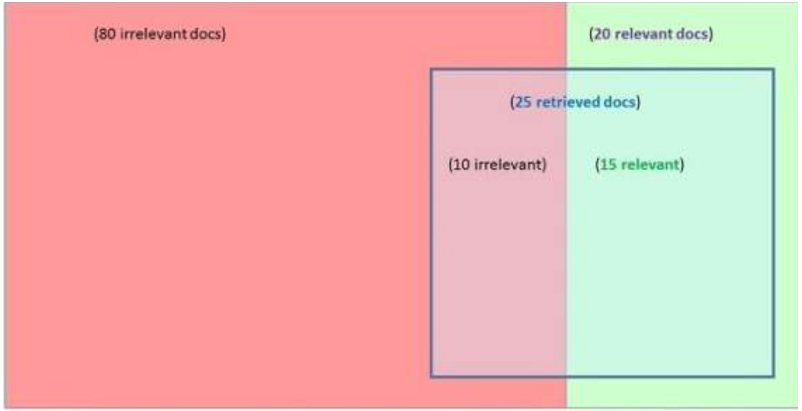
Evaluation: Model Comparison

A			B			C			C′		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112	TP=76	FP=12	88
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88	FN=24	TN=88	112
100	100	200	100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.76		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.12		
PPV = 0.69			PPV = 0.50			PPV = 0.21			PPV = 0.86		
F1 = 0.66			F1 = 0.61			F1 = 0.23			F1 = 0.81		
ACC = 0.68			ACC = 0.50			ACC = 0.18			ACC = 0.82		

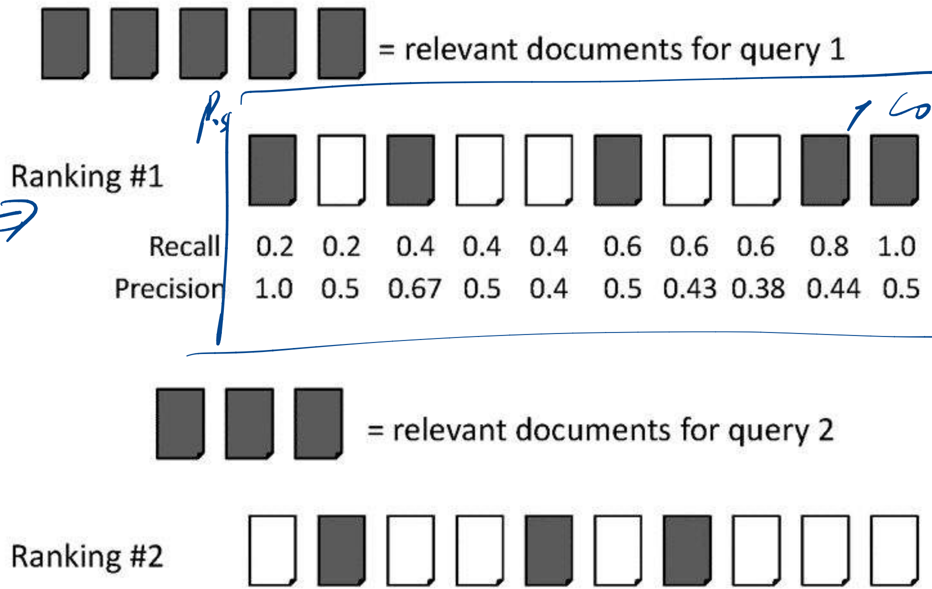


*Precision/Recall in Information Retrieval

precision@k



Precision = 15 / 25 = 0.60 Recall = 15 / 20 = 0.75



Ranking #1

Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5

Ranking #2

Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

Evaluation: sklearn Implementation

- Official Doc (stable version): https://scikit-learn.org/stable/modules/model_evaluation.html
- Some helpful demos:
  - Precision-Recall: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#sphx-glr-auto-examples-model-selection-plot-precision-recall-py
  - Confusion matrix: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metrics.confusion_matrix
  - Receiver Operating Characteristic (ROC): https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#sphx-glr-auto-examples-model-selection-plot-roc-py

Whiteboard

Thank you!

Q & A

UCLA Samueli Computer Science

CS M146 Discussion: Week 7

Kernels, SVM

Junheng Hao

Friday, 02/19/2021

Roadmap

- Announcement
- Kernels
- SVM
- PyTorch Q&A (PS3)

Announcements

- **5:00 pm PST, Feb 19 (Friday):** Weekly Quiz 7 released on Gradescope.
- **11:59 pm PST, Feb 21 (Sunday):** Weekly quiz ~~4~~ 7 closed on Gradescope!
    - Start the quiz before **11:00 pm PST, Feb 21** to have the full 60-minute time
- **Problem set 3** released on CCLE, submission on Gradescope.
    - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
    - You need to submit code, similar to PS2
    - Due on next week, **11:59pm PST, Feb 26 (Friday)**

Late Submission of PS will NOT be accepted!

About Quiz 7

- Quiz release date and time: **Feb 19, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Feb 21, 2021 (Sunday) 11:59 PM PST**
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week 7 Quiz" on GradeScope.
- Topics: **Kernels, SVM**
- Question Types
  - True/false, multiple choices
  - Some questions may include several subquestions.
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.

Quiz 6 Review: Question 1

[Point: **2**] Variables $a, b, c, d, e, f \in \mathbb{R}$ satisfy

$$
\begin{aligned}
c &= \sigma(w_1 \cdot a + w_2 \cdot b) \\
d &= \tanh(w_3 \cdot a + w_4 \cdot b) \\
e &= \mathrm{ReLU}(w_5 \cdot a + w_6 \cdot b) \\
f &= \sigma(w_7 \cdot d + w_8 \cdot e) \\
g &= \sigma(w_9 \cdot c + w_0 \cdot f)
\end{aligned}
$$

$f(d, e)$ $c(a,b)$

$g(c, f)$ $d(a,b)$

$e(a,b)$

where $w_i, i = 0, \ldots, 9$ are constants. Which of the following statements is true? *Hint: It would be helpful to draw a computational graph.*

A.

$$\frac{\partial g}{\partial a} = \frac{\partial g}{\partial c} \cdot \frac{\partial c}{\partial a} + \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial a}$$

$\times$

B.

$$\frac{\partial f}{\partial a} = \frac{\partial d}{\partial a} + \frac{\partial e}{\partial a}$$

$$\frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial a} + \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial a}$$

$$\boxed{\text{C.} \quad \frac{\partial g}{\partial b} = \frac{\partial g}{\partial c} \cdot \frac{\partial c}{\partial b} + \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial b}}$$

✓

D.

$$\frac{\partial g}{\partial b} = \frac{\partial g}{\partial c} \cdot \frac{\partial c}{\partial b} + \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial d} \cdot \frac{\partial d}{\partial b} + \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial e} \cdot \frac{\partial c}{\partial b}$$

Quiz 6 Review: RNN Example

$h^{(t-1)} = W_h \cdot h^{(t-2)} + W_x \cdot x^{(t-1)}$

$W_h \; h^{(t-3)} + \cdots$

Let's consider a simple RNN structure:

$$h^{(t)} = W_h \cdot h^{(t-1)} + W_x \cdot x^{(t)}$$
$$y^{(t)} = W_y \cdot h^{(t)}$$

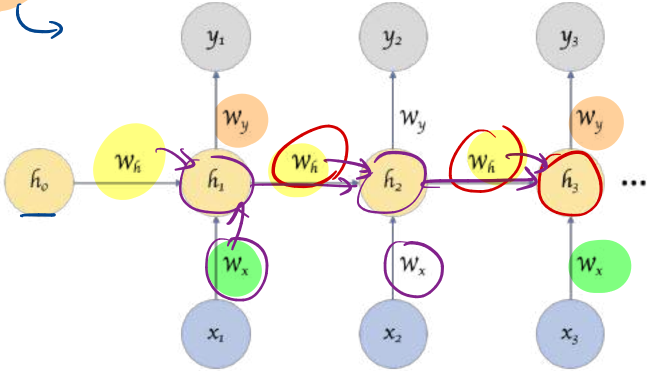
where $t = 1, 2, \ldots, T$, $x^{(t)}$ is the input and $y^{(t)}$ is the output. $h^{(0)}$ is initialized as 0. Which of the following statement(s) is/are true? Note: To make things simpler, we treat all the variables as scalars.

A. $\frac{\partial h^{(t)}}{\partial W_h} = h^{(t-1)}$ ✗ $t=3$

B. ✓ $\frac{\partial h^{(t)}}{\partial x^{(t)}} = W_x$ ✓ $t=2$

C. ✓ $\frac{\partial y^{(t)}}{\partial W_y} = h^{(t)}$ ✓ $t=2$

D. $\frac{\partial y^{(t)}}{\partial W_x} = W_y \cdot x^{(t)}$ ✗



$x_1 \quad x_2 \;\cdots\; x_t$

$$\frac{\partial h^{(t)}}{\partial W_h} = h^{(t-1)}$$

$$\frac{\partial y^{(t)}}{\partial W_x} = \frac{\partial y^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W_x}$$

W_y

Quiz 6 Review: Question 6

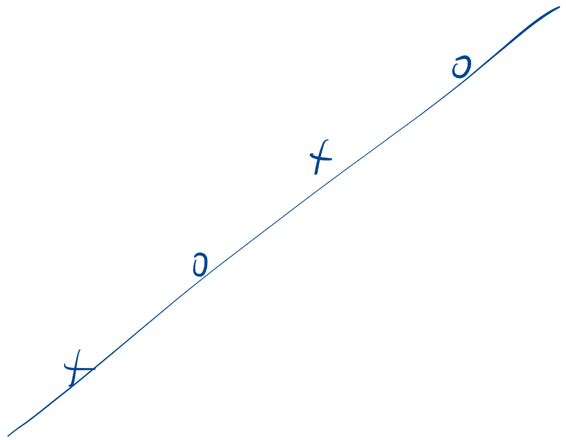
[Point: **2**] What is true about the VC-dimension of linear functions (i.e., hyperplanes) in $\mathbb{R}^2$?

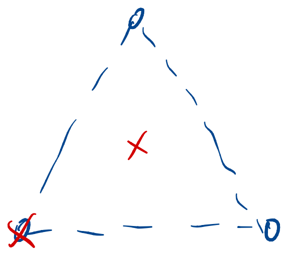
A. There is a set of 3 points that can be shattered.

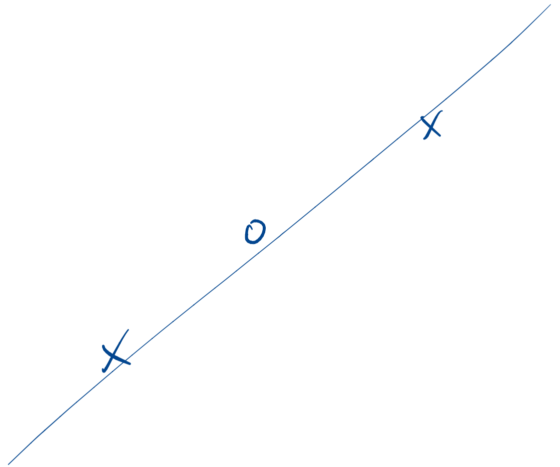
B. Every set of 3 points can be shattered.

C. There is not a set of 4 points that can be shattered.

D. Every set of 4 points can be shattered.







Kernels

- Motivation: Transformed feature space

- Basic idea: Define K, called kernel, such that:

$$K: X \times X \rightarrow \mathbb{R} \qquad \Phi(x) \cdot \Phi(y) = K(x, y)$$

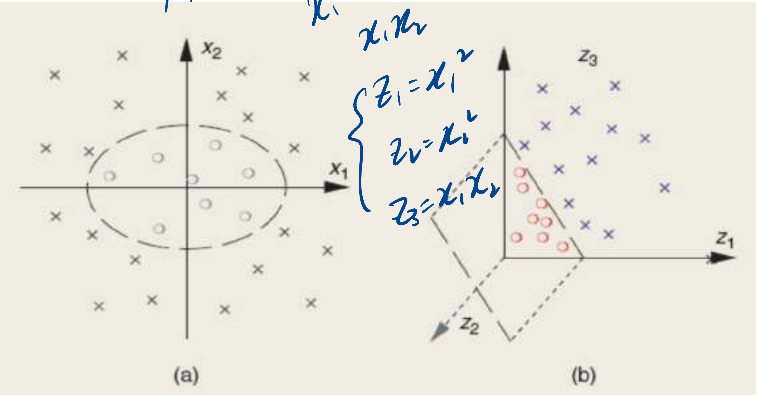
which is often as a similarity measure.

- Benefit:
  - Efficiency: is often more efficient to compute than and the dot product.
  - Flexibility: can be chosen arbitrarily so long as the existence of is guaranteed (Mercer's condition).

$\phi(x)$

$x_1^2 \; x_2^2$

$x_1 x_2$



$\begin{cases} z_1 = x_1^2 \\ z_2 = x_1^2 \\ z_3 = x_1 x_2 \end{cases}$

$\sigma(\beta^T \Phi(x))$

Credit: https://cs.nyu.edu/~mohri/icml2011-tutorial/tutorial-icml2011-1.pdf

Polynomial Kernels

- **Definition:**

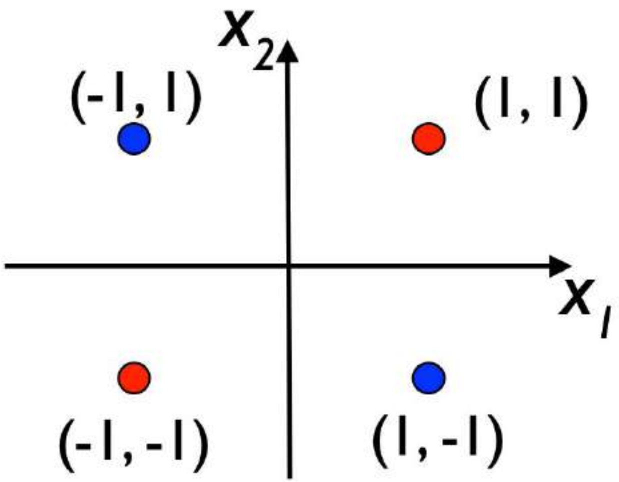
$$\forall x, y \in \mathbb{R}^N, \;\; K(x,y) = (x \cdot y + c)^d, \quad c > 0.$$

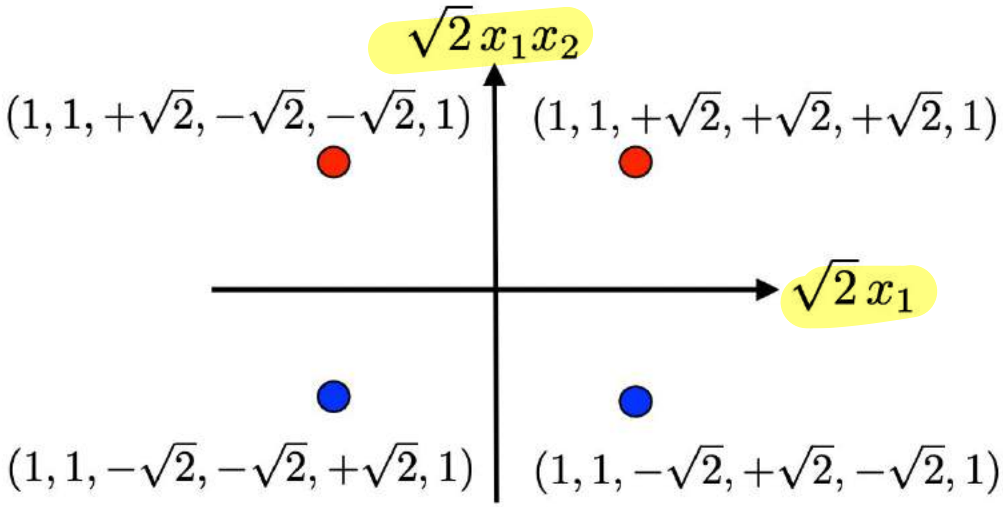
- **Example:** for $N=2$ and $d=2$,

$$K(x,y) = (x_1 y_1 + x_2 y_2 + c)^2$$

$$= \underbrace{\begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \\ \sqrt{2c}\, x_1 \\ \sqrt{2c}\, x_2 \\ c \end{bmatrix}}_{\phi(x)} \cdot \underbrace{\begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}\, y_1 y_2 \\ \sqrt{2c}\, y_1 \\ \sqrt{2c}\, y_2 \\ c \end{bmatrix}}_{\phi(y)}.$$

Kernels: XOR Example





Linearly non-separable

Linearly separable by

$$x_1x_2 = 0.$$

Other Kernel Options

Gaussian kernels:

$$K(x, y) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right), \quad \sigma \neq 0.$$

Also known as "Radial Basis Function Kernel"

$K(x,y) = \phi(x)\phi(y)$

Sigmoid Kernels:

$$K(x, y) = \tanh(a(x \cdot y) + b), \quad a, b \geq 0.$$

Note: The RBF/Gaussian kernel as a projection into infinite dimensions, commonly used in kernel SVM.

$-x^2 - x'^2 + 2xx'$

$2\sigma^2 = 1$

$$K(x, x') = \exp\left(-(x - x')^2\right)$$

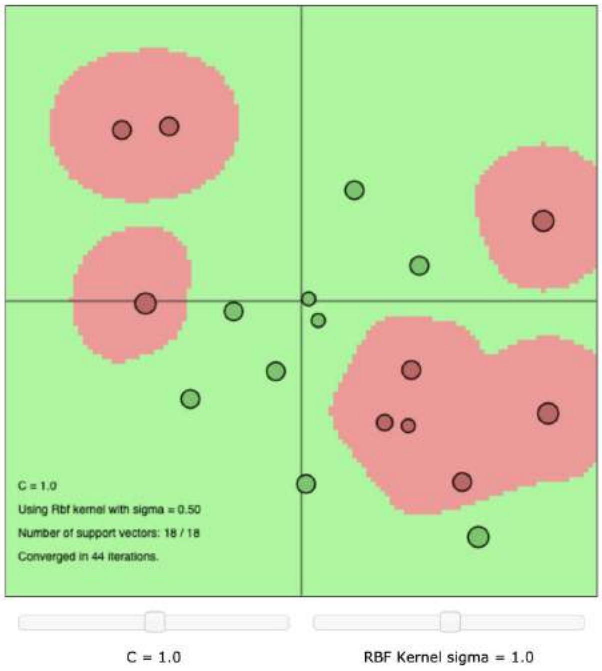
$$= \exp\left(-x^2\right)\exp\left(-x'^2\right)\underbrace{\sum_{k=0}^{\infty}\frac{2^k(x)^k(x')^k}{k!}}_{\exp(2xx')}$$

Taylor Expansion

Credit: http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf

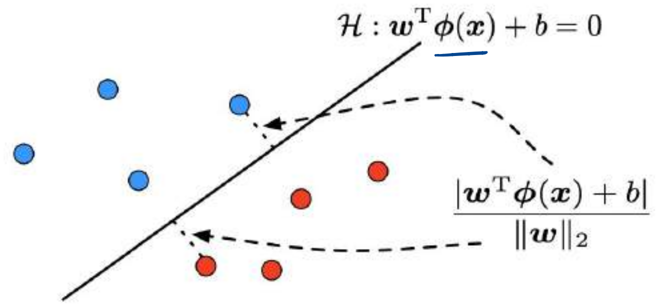
SVM: Visual Tutorials

- Links: https://cs.stanford.edu/people/karpathy/svmjs/demo/



SVM: Margin

$$\text{MARGIN}(\boldsymbol{w}, b) = \min_n \frac{y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]}{\|\boldsymbol{w}\|_2}$$



$w^T x + b = 0$

SVM: Start with the margin

$w^T x + b = 1$

$= -1$

Margin Lines

$$\mathbf{w}^T \mathbf{x}_a + \mathrm{b} = 1 \qquad \mathbf{w}^T \mathbf{x}_b + \mathrm{b} = -1 = 0$$

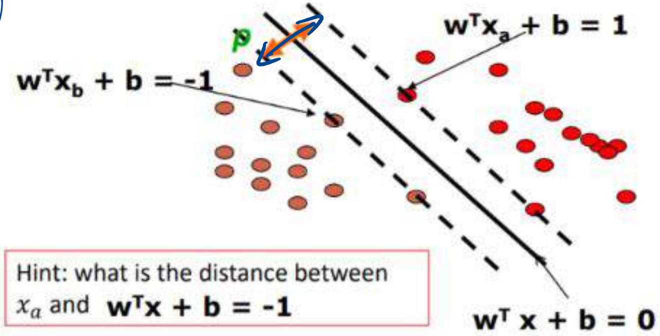
Distance between parallel lines of $ax_1+bx_2=c_1/c_2$)

$$d = \frac{|c_2 - c_1|}{\sqrt{a^2 + b^2}}$$

$\|w\|$

Margin

$$\rho = \frac{|(b+1)-(b-1)|}{\|w\|} = \frac{2}{\|w\|}$$



Hint: what is the distance between x_a and $\mathbf{w}^T\mathbf{x} + \mathbf{b} = \mathbf{-1}$

SVM: Steps to understand SVM

1. Formulation of the Linear SVM problem: maximizing margin
2. Formulation of Quadratic Programming (optimization with linear constraints) → Primal problem
3. Solving linear SVM problem with “great” math*
    a. (Generalized) Lagrange function, lagrange multiplier
    b. Identify primal and dual problem (duality) → KKT conditions
    c. Solution to *w* and b regarding alpha
4. Support Vectors, SVM Classifier Inference
5. Non-linear SVM, Kernel tricks

SVM: Math Behind

- Slides: http://people.csail.mit.edu/dsontag/courses/ml13/slides/lecture6.pdf
- Notes: https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf

**To show in hand notes*

Whiteboard for SVM Math Foundation

This slide is intentionally left blank.

Given labeled data and alpha values

- Positively labeled data points (1 to 4)

x_1 x_2

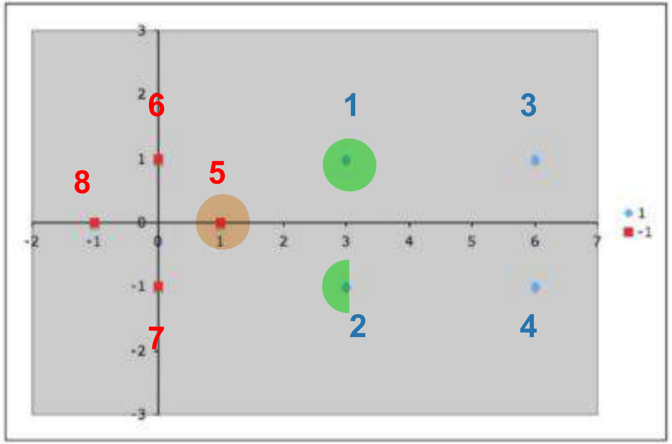
$$\left\{\begin{pmatrix}3\\1\end{pmatrix},\begin{pmatrix}3\\-1\end{pmatrix},\begin{pmatrix}6\\1\end{pmatrix},\begin{pmatrix}6\\-1\end{pmatrix}\right\}$$

- Negatively labeled data points (5 to 8)

x_5

$$\left\{\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}0\\-1\end{pmatrix},\begin{pmatrix}-1\\0\end{pmatrix}\right\}$$

- Alpha values
  - $\alpha_1 =$ 0.25
  - $\alpha_2 =$ 0.25
  - $\alpha_5 =$ 0.5
  - Others = 0



$w = \sum_j \alpha_j y_j x_j$

$j = 1, 2, 5$

$j=1$ $j=2$ $j=5$

$w = 0.25 \cdot 1 \cdot \begin{pmatrix}3\\1\end{pmatrix} + 0.25 \cdot 1 \cdot \begin{pmatrix}3\\-1\end{pmatrix} + 0.5 \times (-1) \times \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}$

α_1

Linear SVM: Example for Practice

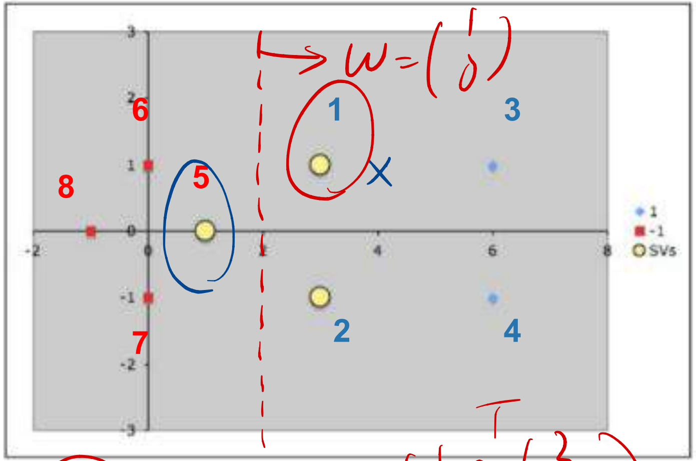
Questions

$w^T\begin{pmatrix}4\\1\end{pmatrix}+b>0$

- Which points are support vectors?
- Calculate normal vector of hyperplane: $\boldsymbol{w}$
- Calculate the bias term
- What is the decision boundary?
- Predict class of new point (4, 1)

$w^T x + b = \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} - 2 = 0$

$x_1 - 2 = 0$



$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \qquad b = \sum_{k:\alpha_k \neq 0} (y_k - \mathbf{w}^T \mathbf{x}_k)/N_k$$

$w = \begin{pmatrix}1\\0\end{pmatrix}$ $\quad b = -1 - \begin{pmatrix}1\\0\end{pmatrix}^T\begin{pmatrix}1\\0\end{pmatrix} = -2$ $\quad b = 1 - \begin{pmatrix}1\\0\end{pmatrix}^T\begin{pmatrix}3\\-1\end{pmatrix} = -2$

Predictions for new data

$$y \leftarrow \text{sign}(\vec{w} \cdot \vec{x} + b)$$

Using dual solution

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i \underbrace{(\vec{x}_i \cdot \vec{x})} + b\right]$$

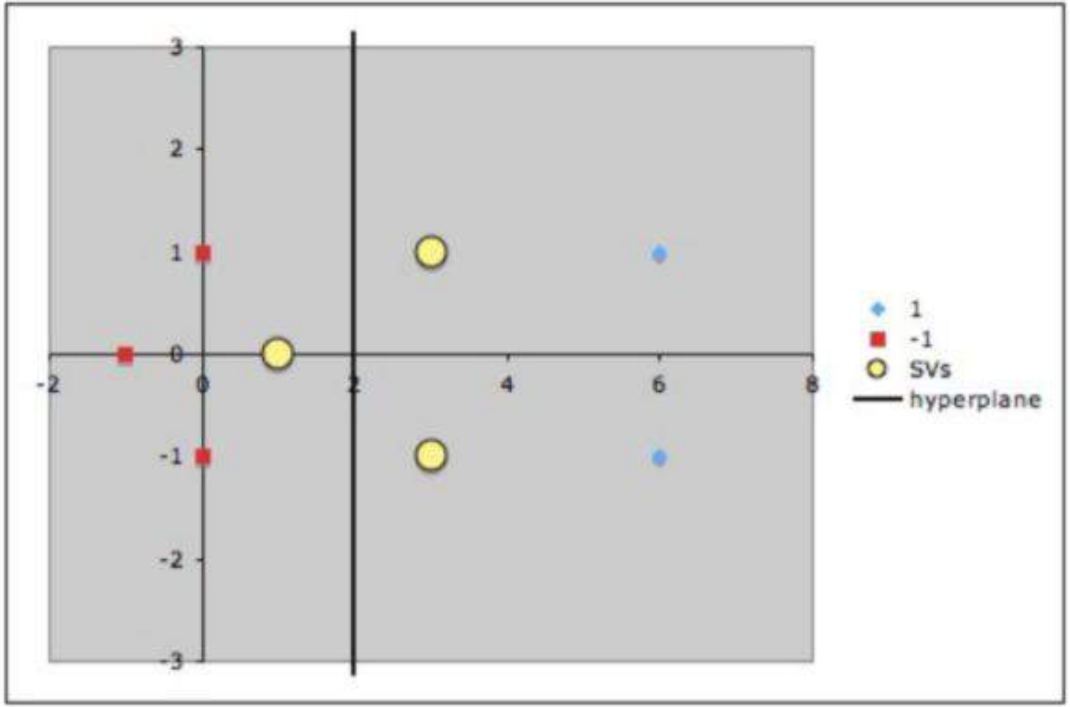
dot product of feature vectors of new example with support vectors

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

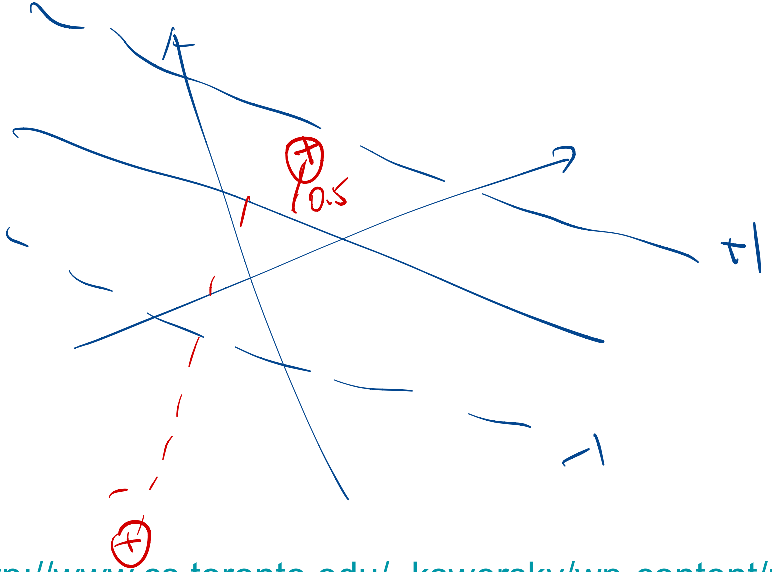
for any k where $C > \alpha_k > 0$

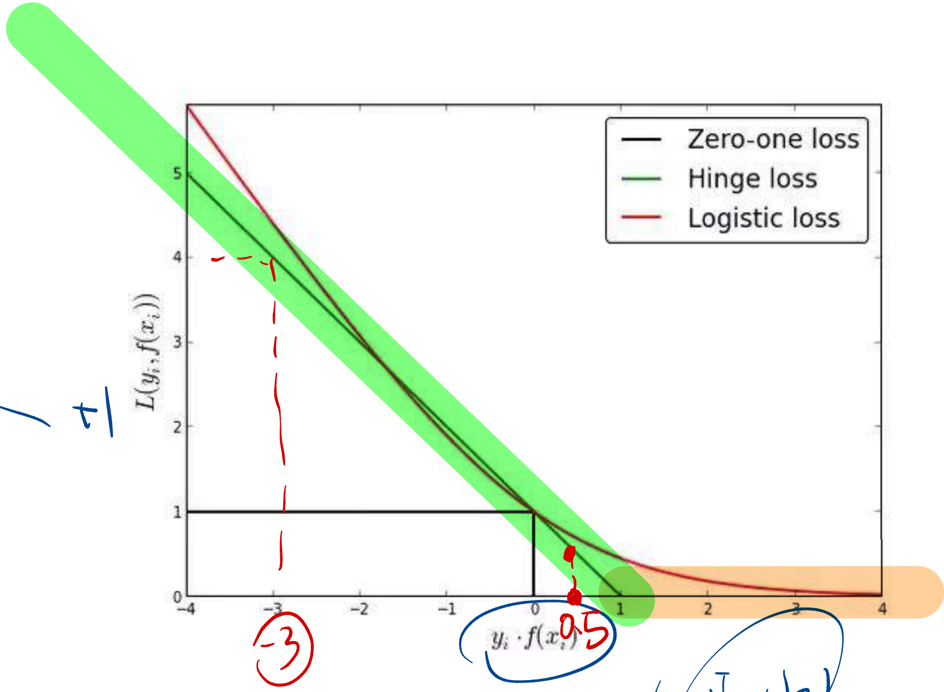
Plot



Linear SVM vs Logistic Regression

- Decision boundaries?
- Loss functions?



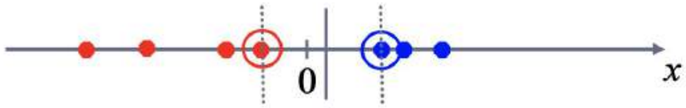


1. $(w^T x + b) \geq 1$

Reading: http://www.cs.toronto.edu/~kswersky/wp-content/uploads/svm_vs_lr.pdf

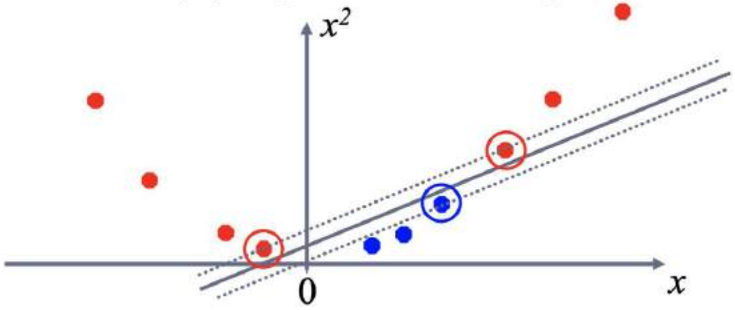
Non-linear SVM

- Datasets that are linearly separable (with some noise) work out great:



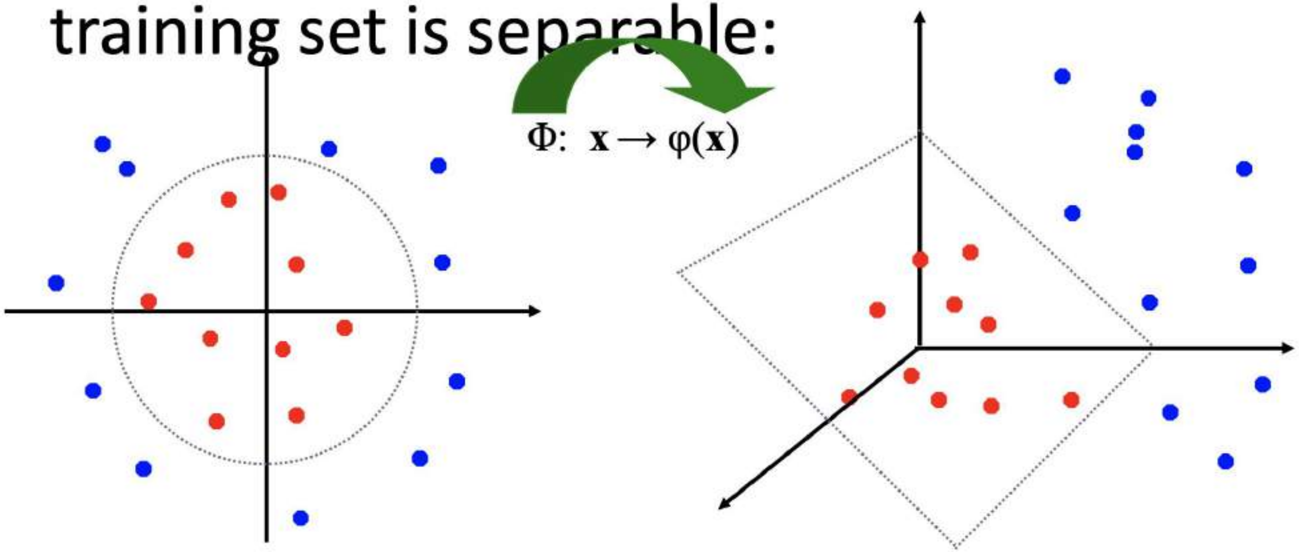
- But what are we going to do if the dataset is just too hard?

- How about … mapping data to a higher-dimensional space:



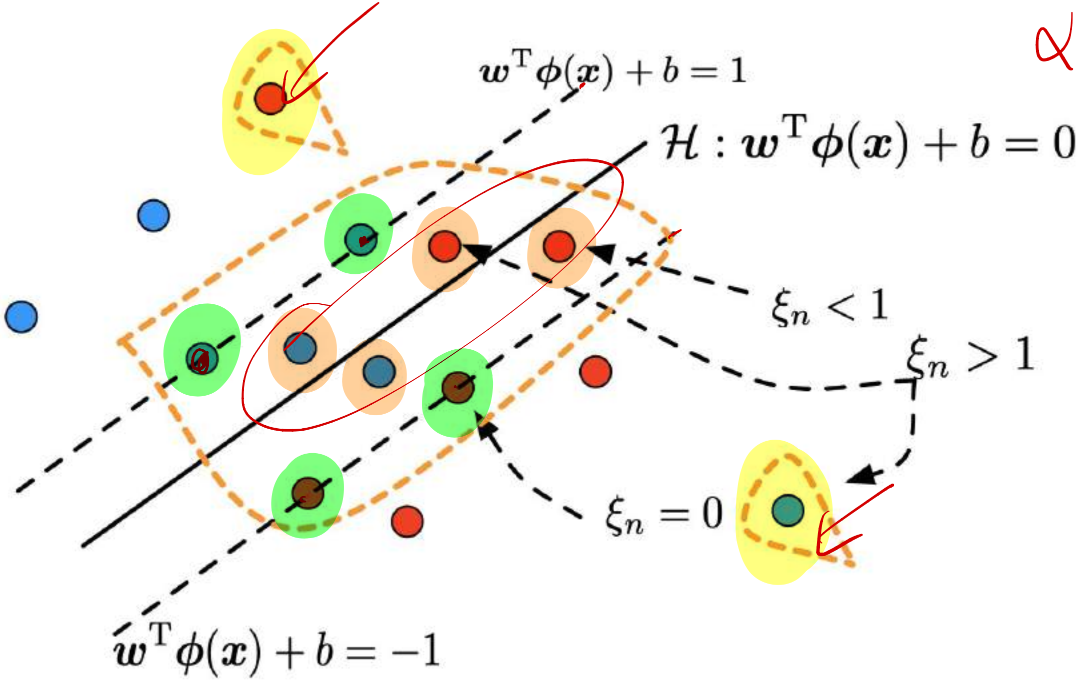
Non-linear SVM

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



Support Vectors

$\alpha_i \neq 0$



SVM: Kernel

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \boxed{\mathbf{x}_i \mathbf{x}_j}$$

$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \boxed{K(\mathbf{x}_i, \mathbf{x}_j)}$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

SVM: Kernel

- The linear SVM relies on an inner product between data vectors,

$$K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i^T x_j}$$

- If every data point is mapped into high-dimensional space via transformation, the inner product becomes,

$$K(\mathbf{x_i}, \mathbf{x_j}) = \phi^T(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$$

- Do we need to compute ***ϕ(x)*** explicitly for each data sample? → **Directly compute kernel function *K(xi, xj)***

SVM: Kernel Function Example

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^T\mathbf{z} + c)^2 = \left(\sum_{j=1}^{n} x^{(j)} z^{(j)} + c\right)\left(\sum_{\ell=1}^{n} x^{(\ell)} z^{(\ell)} + c\right) \\
&= \sum_{j=1}^{n}\sum_{\ell=1}^{n} x^{(j)} x^{(\ell)} z^{(j)} z^{(\ell)} + 2c\sum_{j=1}^{n} x^{(j)} z^{(j)} + c^2 \\
&= \sum_{j,\ell=1}^{n} (x^{(j)} x^{(\ell)})(z^{(j)} z^{(\ell)}) + \sum_{j=1}^{n} (\sqrt{2c}x^{(j)})(\sqrt{2c}z^{(j)}) + c^2,
\end{aligned}
$$

Feature mapping given by:

$$
\mathbf{\Phi}(\mathbf{x}) = [x^{(1)2}, x^{(1)}x^{(2)}, \ldots, x^{(3)2}, \sqrt{2c}x^{(1)}, \sqrt{2c}x^{(2)}, \sqrt{2c}x^{(3)}, c]
$$

SVM: Kernel Function Example

Polynomial kernel of degree h : $K(X_i, X_j) = (X_i \cdot X_j + 1)^h$

Gaussian radial basis function kernel : $K(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\sigma^2}$ ✓

Sigmoid kernel : $K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$

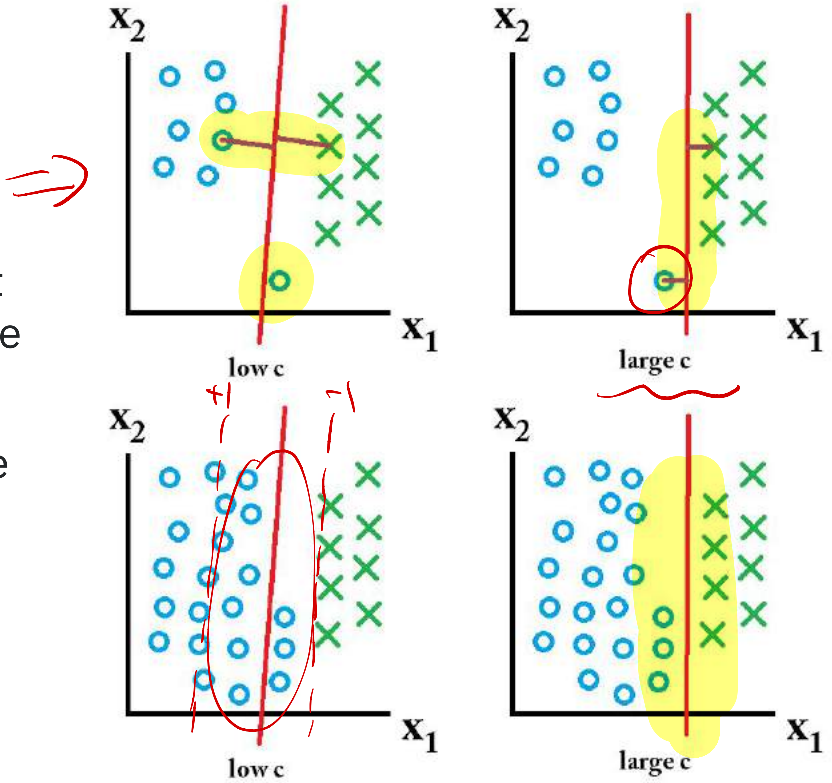
- Given the same data samples, what is the difference between linear kernel and non-linear kernel? Is the decision boundary linear (in original feature space)?

SVM: Overfitting

- Huge feature space with kernels: should we worry about overfitting?
  - SVM objective seeks a solution with large margin.
  - Theory says that large margin leads to good generalization.
  - But everything overfits sometimes.
  - Can control by:
    - Setting C
    - Choosing a better Kernel
    - Varying parameters of the Kernel (width of Gaussian, etc.)

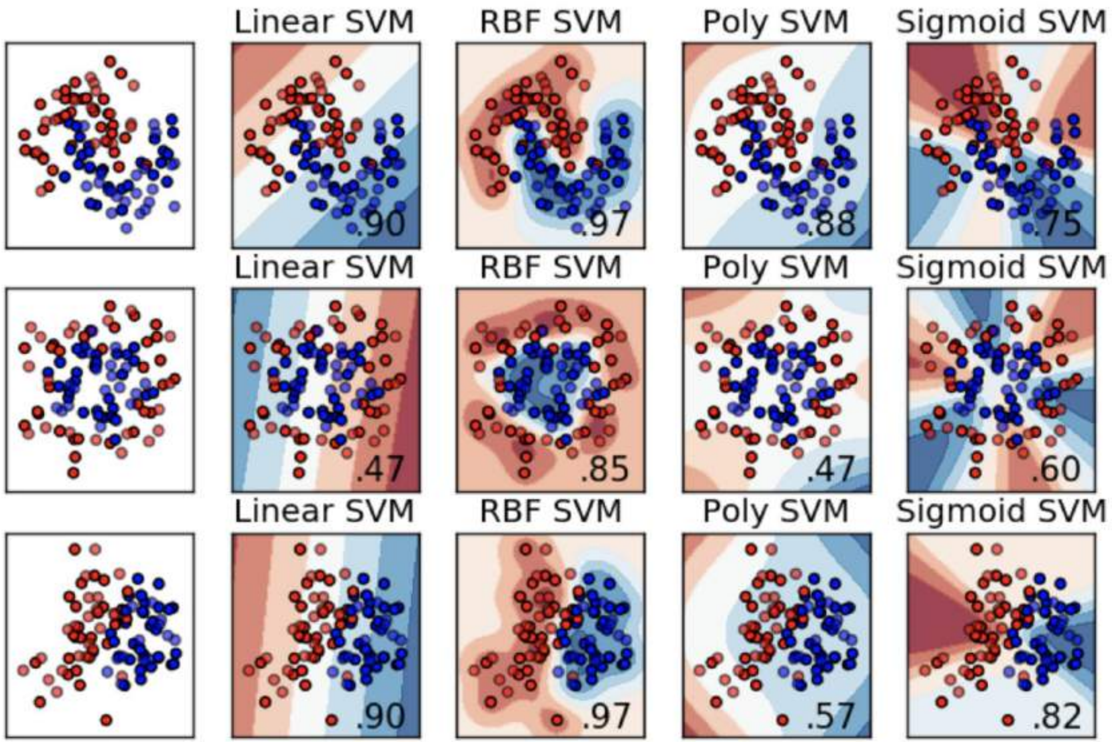
SVM: Understanding C

- The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.
- For large values of C, the optimization will choose **a smaller-margin hyperplane** if that hyperplane does a better job of getting all the training points classified correctly.
- Conversely, a very small value of C will cause the optimizer to look for **a larger-margin separating hyperplane**, even if that hyperplane misclassified more points.



Reference: https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

SVM: Demo of different kernels



Non-linear SVM: Example Practice

- Positivelv labeled data points (1 to 4)

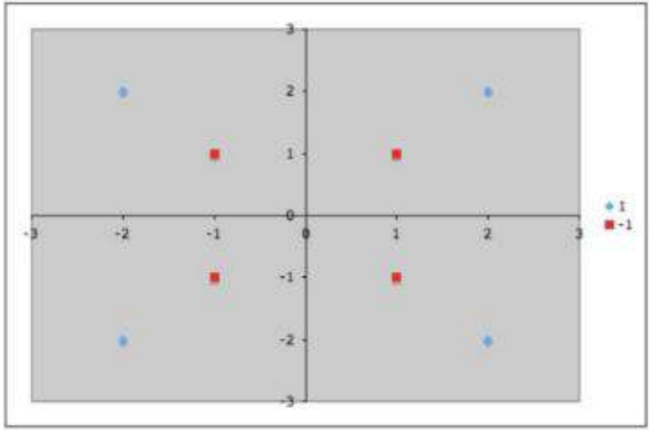
$$\left\{\begin{pmatrix}2\\2\end{pmatrix},\begin{pmatrix}2\\-2\end{pmatrix},\begin{pmatrix}-2\\-2\end{pmatrix},\begin{pmatrix}-2\\2\end{pmatrix}\right\}$$

- Negatively labeled data points (5 to 8)

$$\left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}1\\-1\end{pmatrix},\begin{pmatrix}-1\\-1\end{pmatrix},\begin{pmatrix}-1\\1\end{pmatrix}\right\}$$

- Non-linear mapping

$$\Phi_1\begin{pmatrix}x_1\\x_2\end{pmatrix}=\begin{cases}\begin{pmatrix}4-x_2\\4-x_1\end{pmatrix} & \text{if } \sqrt{x_1^2+x_2^2}>2\\ \begin{pmatrix}x_1\\x_2\end{pmatrix} & \text{otherwise}\end{cases}$$



Non-linear SVM Example

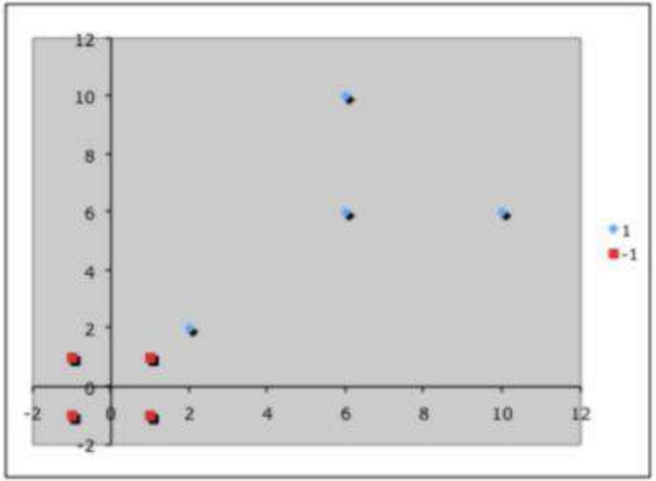
- New positively labeled data points (1 to 4)

$$\left\{\begin{pmatrix}2\\2\end{pmatrix},\begin{pmatrix}6\\2\end{pmatrix},\begin{pmatrix}6\\6\end{pmatrix},\begin{pmatrix}2\\6\end{pmatrix}\right\}$$

- New negatively labeled data points (5 to 8)

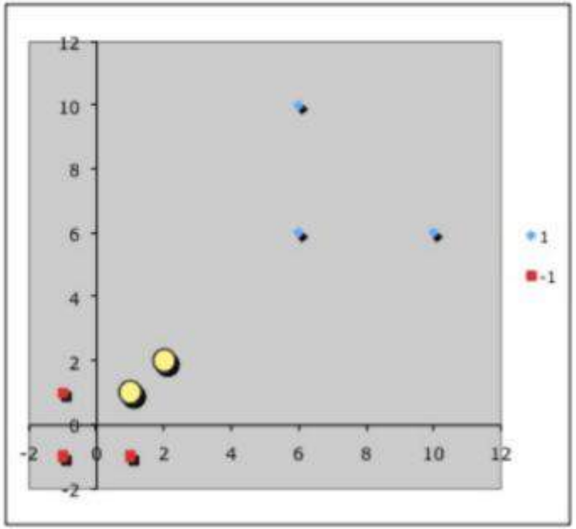
$$\left\{\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}1\\-1\end{pmatrix},\begin{pmatrix}-1\\-1\end{pmatrix},\begin{pmatrix}-1\\1\end{pmatrix}\right\}$$

- Alpha values
  - $\alpha_1 = \boxed{1.0}$
  - $\alpha_5 = \boxed{1.0}$
  - Others = 0

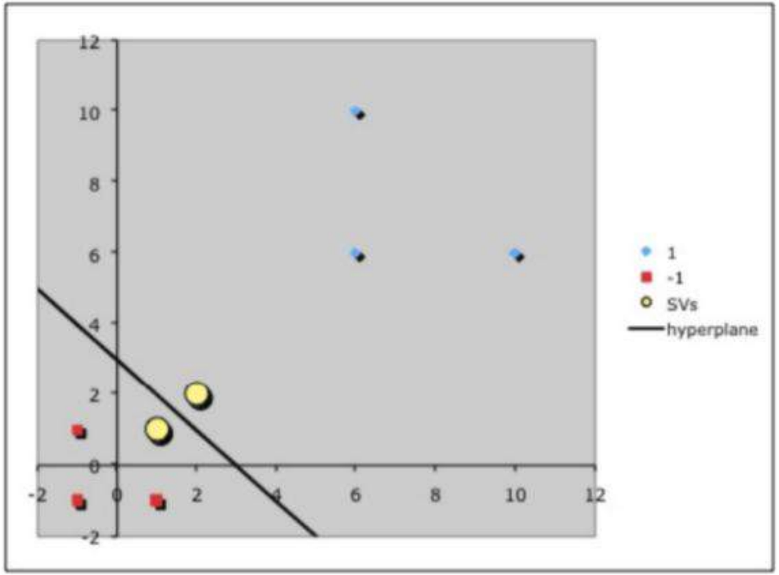


Non-linear SVM Example

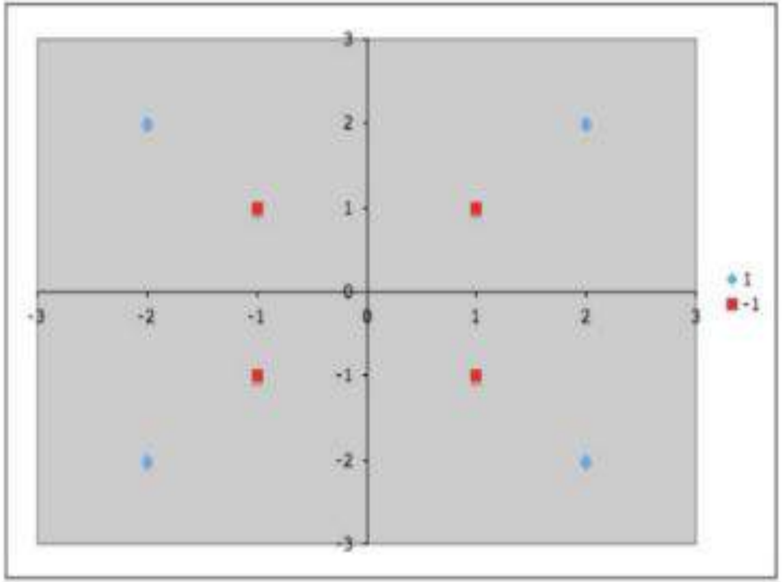
- Which points are support vectors?
- Calculate normal vector of hyperplane: $\boldsymbol{w}$
- Calculate the bias term
- What is the decision boundary?
- Predict class of new point (4, 5)



Plot



12
10
8
6
4
2
0
-2
-2
0
2
4
6
8
10
12
1
-1
SVs
hyperplane



3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
1
-1

Non-linear SVM Solution

- Decision Boundary

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i K(x_i, x) + b\right]$$

UCLA Samueli Computer Science

Thank you!

Q & A

SVM Dual Problem: How to optimize?

- The answer is Sequential Minimal Optimization (SMO) Algorithm.
- Basic idea: optimization problem of multiple variables is decomposed into a series of subproblems each optimizing an objective function of a small number of variables, typically only one, while all other variables are treated as constants that remain unchanged in the subproblem.
- Formulation:

$$
\begin{aligned}
\textbf{maximize:} \quad & L(\alpha_i, \alpha_j) = \alpha_i + \alpha_j - \frac{1}{2}\left(\alpha_i^2 \mathbf{x}_i^T \mathbf{x}_i + \alpha_j^2 \mathbf{x}_j^T \mathbf{x}_j + 2\alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j\right) \\
& -\alpha_i y_i \left(\sum_{n \neq i} \alpha_n y_n \mathbf{x}_n^T\right) \mathbf{x}_i - \alpha_j y_j \left(\sum_{n \neq j} \alpha_n y_n \mathbf{x}_n^T\right) \mathbf{x}_j \\
& = \alpha_i + \alpha_j - \frac{1}{2}\left(\alpha_1^2 K_{ii} + \alpha_2^2 K_{jj} + 2\alpha_i \alpha_j y_i y_j K_{ij}\right) \\
& -\alpha_i y_i \sum_{n \neq i,j} \alpha_n y_n K_{ni} - \alpha_j y_j \sum_{n \neq i,,j} \alpha_n y_n K_{nj} \\
\textbf{subject to:} \quad & 0 \leq \alpha_i, \alpha_j \leq C, \qquad \sum_{n=1}^{N} \alpha_n y_n = 0
\end{aligned}
$$

Whiteboard

- Content

UCLA Samueli Computer Science

CS M146 Discussion: Week 9

Naive Bayes, Clustering (K-Means, Gaussian Mixture Model)

Junheng Hao

Friday, 03/05/2021

Roadmap

- Announcement
- Naive Bayes
- K-Means
- Gaussian Mixture Model

Announcements

- **5:00 pm PST, Mar 5 (Friday):** Weekly Quiz 9 released on Gradescope.
- **11:59 pm PST, Mar 7 (Sunday):** Weekly Quiz 9 closed on Gradescope!
  - Start the quiz before **11:00 pm PST, Mar 7** to have the full 60-minute time
- **Grading update:** Lowest two quiz scores are dropped. The rest ~~7~~ 6 quizzes are counted into final grading.
- **Problem set 4** released on CCLE, submission on Gradescope.
  - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
  - You need to submit code and the results required by the problem set
  - Due on **next Friday, 11:59pm PST, Mar 12 (Friday)**

Late Submission of PS will NOT be accepted!

About Quiz 9

- Quiz release date and time: **Mar 5, 2021 (Friday) 05:00 PM PST**
- Quiz due/close date and time: **Mar 7, 2021 (Sunday) 11:59 PM PST**
- You will have up to **60 minutes** to take this exam. → Start before **11:00 PM** Sunday
- You can find the exam entry named "Week 9 Quiz" on GradeScope.
- Topics: **Naive Bayes, Clustering**
- Question Types
  - True/false, multiple choices
  - Some questions may include several subquestions.
- Some light calculations are expected. Some scratch paper and one scientific calculator (physical or online) are recommended for preparation.
- **Note: This is the last quiz in this quarter. Highest ~~7~~ 6 quiz scores are counted for final grading.**

Prof. Sankararaman's post on updated quiz grading:
https://campuswire.com/c/GB5E561C3/feed/438

Updates: Final Exam

- Open book and open notes, on GradeScope: “quiz”-like exam
- Start attempting the exam from **8:00 am PST on March 15**; Submit your exam before **8:00am PST March 16** (No extensions). → 24h time window
- Exam duration: **3 hours** (time limit after start the exam)
- Type: True/false and multiple choice questions (free text boxes are given for justification)
- The instructors will be available to provide clarifications on CampusWire (visible for everyone) from 8:00am-11:00am on March 15. Later questions on Campuswire may not be answered.
- Some calculations are expected.

MUST READ: Official post about final exam on Campuswire:
https://campuswire.com/c/GB5E561C3/feed/437

Naive Bayes: Model Summary

- Defines a joint distribution

$P(X, Y)$ prior

$$P(X = x, Y = c) = P(Y = c) \prod_{d=1}^{D} P(X_d = x_d | Y = c)$$

input class label

$$= P(Y = c) \prod_{k} P(k|Y = c)^{z_k} = \pi_c \prod_{k} \theta_{ck}^{z_k}$$

- Learning problem formulation

Training data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N} \rightarrow \mathcal{D} = \{(\{z_{nk}\}_{k=1}^{K}, y_n)\}_{n=1}^{N}$

Constraints $\sum_c \pi_c = 1$ $\sum_k \theta_{ck} = \sum_k P(k|Y = c) = 1$

Objective $\mathcal{L} = \log P(\mathcal{D}) = \log \prod_{n=1}^{N} \pi_{y_n} P(x_n|y_n) \longrightarrow (\pi_c^*, \theta_{ck}^*) = \arg\max \sum_n \log \pi_{y_n} + \sum_{n,k} z_{nk} \log \theta_{y_n k}$

MAP

- Solution

$$\pi_c^* = \frac{\text{\#of data points labeled as c}}{\text{N}}$$

$$\theta_{ck}^* = \frac{\text{\#of times word k shows up in data points labeled as c}}{\text{\#data points labeled as c}}$$

$\sum_c \pi_c^* = 1$

$\#k$

$\theta_{ck} = {}^{\#C} \Big[\qquad \Big]$

Naive Bayes: Derivation

A short derivation of the maximum likelihood estimation

To maximize

$$\sum_{n:y_n=c,k} z_{nk} \log \theta_{ck}$$

We use the Lagrangian multiplier

$$\sum_{n:y_n=c,k} z_{nk} \log \theta_{ck} + \lambda \left(\sum_k \theta_{ck} - 1 \right)$$

Taking derivatives with respect to θ_{ck} and then find the stationary point

$$\left(\sum_{n:y_n=c} \frac{z_{nk}}{\theta_{ck}} \right) + \lambda = 0 \rightarrow \theta_{ck} = -\frac{1}{\lambda} \sum_{n:y_n=c} z_{nk}$$

Apply constraint $\sum_k \theta_{ck} = 1$, plug in expression above for θ_{ck}, solve for λ, and plug back into expression for θ_{ck}:

$$\theta_{ck} = \frac{\sum_{n:y_n=c} z_{nk}}{\sum_k \sum_{n:y_n=c} z_{nk}}$$

Naive Bayes: Example

- #docs in Class 1: 25, #docs in Class 2: 75

Index	Word	Count in Class 1	Count in Class 2
1	I	1	9
2	like	2	1
3	machine	3	9
4	learning	4	1

Naive Bayes: Learning (Step 1)

- How to obtain the parameters in Naive Bayes classifier? *(shown in class)*

$C = 1$

$C: 2$

$\boxed{\pi_1 + \pi_2 = 1}$

$\pi_1 = \frac{25}{25+75} = \frac{1}{4}$

$\pi_2 = \frac{75}{25+75} = \frac{3}{4}$

Index	Word	# in C1	# in C2
1	I	1	9
2	like	2	1
3	machine	3	9
4	learning	4	1
	total	10	20

θ_{ck}

#C #word

$\Sigma = 1$ $\Sigma = 1$

$$\begin{array}{l c} & \begin{array}{cc} C_1 & C_2 \end{array} \\ \begin{array}{l} \text{"I"} \\ \text{"like"} \\ \text{"machine"} \\ \text{"learning"} \end{array} & \begin{bmatrix} \frac{1}{10} & \frac{9}{20} \\ \frac{2}{10} & \frac{1}{20} \\ \frac{3}{10} & \frac{9}{20} \\ \frac{4}{10} & \frac{1}{20} \end{bmatrix} \end{array}$$

$\boxed{\sum_k \theta_{ck} = 1}$

C1

C2

Naive Bayes: Predicting

- Predicting new document: {I:3, like:1, machine:5, learning:1} *(shown in class)*

Index	Word	# in C1	# in C2
1	I	1 +1	9 +1
2	like	2 +1	1 +1
3	machine	3 +1	9 +1
4	learning	4 +1	1 +1
5	love	0 +1	0 +1

$$P(X, Y=1) = \underbrace{P(Y=1)}_{\pi_1} \cdot \prod_k \theta_{1k}^{z_k} = \frac{1}{4} \times \left(\frac{1}{10}\right)^3 \times \left(\frac{2}{10}\right)^1 \times \left(\frac{3}{10}\right)^5 \times \left(\frac{4}{10}\right)^1$$

$$\Lambda$$

$$P(X, Y=2) = \underbrace{P(Y=2)}_{\pi_2} \cdot \prod_k \theta_{2k}^{z_k} = \frac{3}{4} \times \left(\frac{9}{20}\right)^3 \times \left(\frac{1}{20}\right)^1 \times \left(\frac{9}{20}\right)^5 \times \left(\frac{1}{20}\right)^1$$

- One further question: How to predict new document: {I:3, like:1, machine:5, learning:1, love:2} → Label Smoothing

$$\theta_{ck} = 0 \qquad c=1 \;\; k=5 \qquad c=2 \;\; \text{"love"}$$

$$\theta^*_{ck} = \begin{array}{c|cc} & C_1 & C_2 \\ \hline d_1 & \frac{2}{15} & \\ \vdots & \frac{3}{15} & \\ & \frac{4}{15} & \\ & \frac{5}{15} & \\ k_5 & \frac{1}{15} & \end{array}$$

$$P(\text{"Love"} \mid Y=1) = \cancel{0} \quad \frac{1}{15}$$

$$P(\text{"love"} \mid Y=2) = \cancel{0} \quad \frac{1}{15}$$

Naive Bayes: Summary, Pros and Cons

- A linear classifier (same as logistic regression)
- Generative model, modeling joint distribution (probabilities) → *What is the model assumption?*
- Pros:
  - Fast and simple compared to other complicated algorithms, easy training
  - Works well with high-dimension data such as text classification
- Cons:
  - Strong assumptions (feature independency)
  - Not fit to regression
  - Smoothing is somewhat required for generalization

Generative vs Discriminative Models

- Training classifiers involve estimating *f: X → Y*, or P(Y|X)

- Generative classifiers → *"distribution"*
  - Assume some functional form for P(Y), P(X|Y)
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X)
  - Actually learn the underlying structure of the data

- Discriminative Classifiers → *"boundary"*
  - Assume some functional form for P(Y|X)
  - Estimate parameters of P(Y|X) directly from training data.
  - Learn the mappings directly from the points to the classes

- Generative models
  - Naive Bayes
  - HMM

- Discriminative models
  - Logistic Regression
  - Neural Network / Perceptron
  - SVM

Q: With the aim of classification only, which type of models may less expensive?

Credit: https://stats.stackexchange.com/questions/12421/generative-vs-discriminative

Naive Bayes vs Logistic Regression

- Compare the learning and prediction procedure on Naive Bayes and Logistic Regression in spam classification example

doc $\{ w_1 : c_1 \quad w_2 : c_2 \quad \cdots w_k : c_k \}$

\# n

NB

$\pi_c \quad \theta_{ck}$

$P(Y=1, x) \gtrless P(Y=2, x)$

LR

$\sigma(\beta^T X) \quad \beta_0 \ \beta_1 \cdots \beta_k$

$\beta_1 - c_1 \quad \beta_k - c_k$

β^*

$= \sigma(\underbrace{\beta^* \cdot x_{test}}_{= 0})$

threshold $= 0.5$

MLE vs MAP

- From Bayes rule:

Likelihood
How probable is the evidence given that our hypothesis is true?

Prior
How probable was our hypothesis before observing the evidence?

$$P(H \mid e) = \frac{P(e \mid H)\, P(H)}{P(e)}$$

Posterior
How probable is our hypothesis given the observed evidence? (Not directly computable)

Marginal
How probable is the new evidence under all possible hypotheses?
$P(e) = \sum P(e \mid H_i)\, P(H_i)$

- Comparing MAP and MLE: [Link]

$$\begin{aligned}
\theta_{MLE} &= \arg\max_\theta \log P(X|\theta) \\
&= \arg\max_\theta \log \prod_i P(x_i|\theta) \\
&= \arg\max_\theta \sum_i \log P(x_i|\theta)
\end{aligned}$$

$\arg\max_\theta P(\theta|X)$

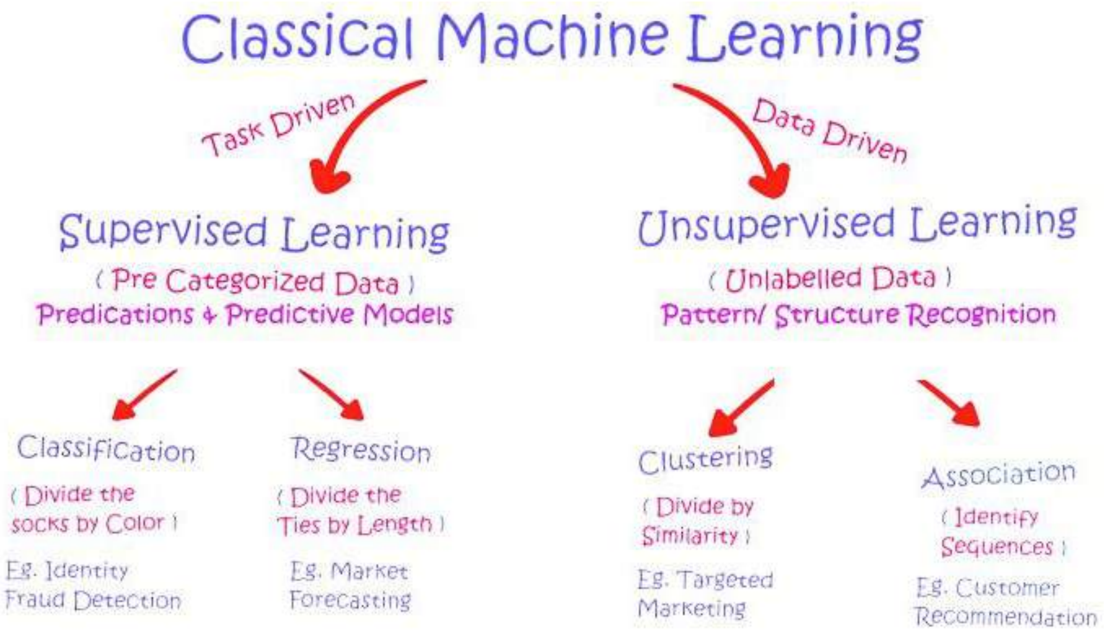
$$\begin{aligned}
\theta_{MAP} &= \arg\max_\theta P(X|\theta) P(\theta) \\
&= \arg\max_\theta \log P(X|\theta) + \log P(\theta) \\
&= \arg\max_\theta \log \prod_i P(x_i|\theta) + \log P(\theta) \\
&= \arg\max_\theta \sum_i \log P(x_i|\theta) + \log P(\theta)
\end{aligned}$$

Clustering

- Clustering: Input / Output / Goal of clustering analysis
  - Large amount of unlabeled data in real life
- Supervised learning v.s. unsupervised learning
- Unsupervised learning cases: Clustering and dimension reduction
- Clustering algorithm examples in this course:
  - K-means
  - Gaussian Mixture models
- Dimension reduction algorithm examples in this course:
  - PCA

Supervised learning / Unsupervised learning

	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction



Applications of ML Categories

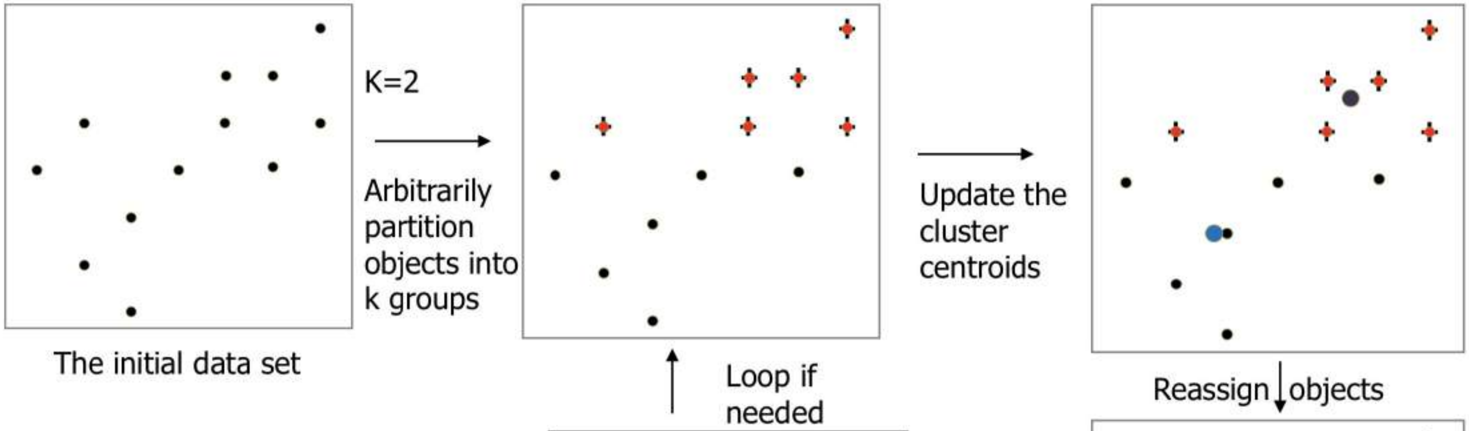
Supervised learning	Unsupervised learning	Semi-supervised
Personalized marketing	Customer grouping or clustering, e.g. discovering groups of similar visitors to a website or discovering that a group of patients respond to the same treatment	Used in applications where labeled data is scarce/ expensive
Recommendation engines	Anomaly detection or finding outliers in the data for better fraud detection or security incident identification	Speech analytics
Insurance / credit underwriting decisions	Product affinity/association rule engine, e.g. discovering which two products sell best together	Image classification
Fraud detection		Web content classification
Spam filtering		Medical predictions
Demand sensing		Protein sequence classification
Predictive maintenance		
Sales performance prediction		
People analytics		

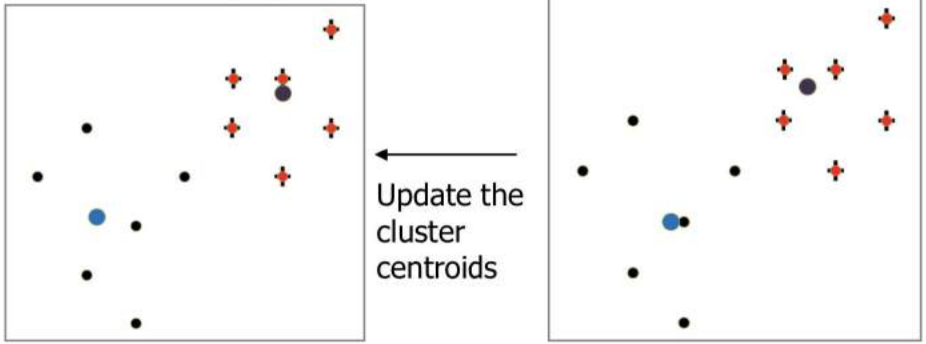
Other “learning”: Self-supervised learning, reinforcement learning

K-means

- Demo 1: http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html
- Demo 2: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

K-means: Steps





- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

K-means

- Distortion measure

$$J(\{r_{nk}\}, \{\boldsymbol{\mu}_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

- Key idea of K-means algorithms:
  - Step 1: Partition into k non-empty subsets (select K points as initial centroids)
  - Step 2: Iteration: Update mean point and assign object to cluster again
  - Step 3: Stop when converge
- Partition-based clustering methods
- Can be considered as a special case of Gaussian Mixture Model (GMM)

K-means

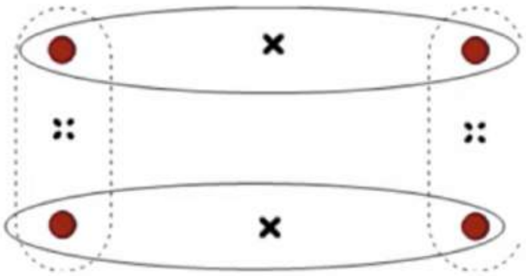
- Q1: Will K-means converge?
- Q2: Will different initialization of K-means generate different clustering results?

K-means

- Q1: Will K-means converge?
- A1: Yes.

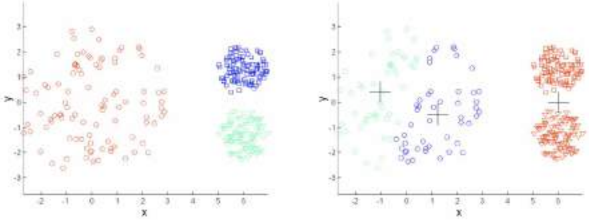
$$J = \sum_{j=1}^{k} \sum_{C(i)=j} d(x_i, c_j)^2$$

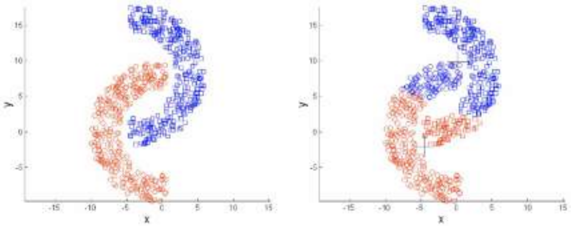
- Q2: Will different initialization of K-means generate different clustering results?
- A2: Yes. Initialization matters!



K-means: Discussion

- Efficiency: **O(tkn)** normally k,t are much smaller than n → efficient
- Can terminate at a local optimum
- Need to specify **k** (or take time to find best **k**)
- Sensitive to noisy data and outliers → K-medoids
- Different sizes and variances
- Not suitable to discover clusters with non-convex shapes
- Many variants of K-means:
  - K-means++, Genetics K-means, etc.

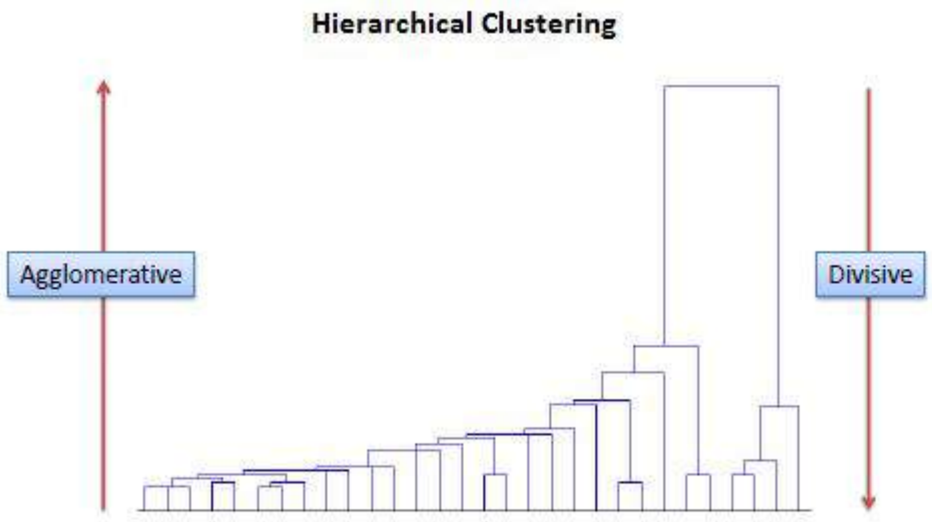




*Hierarchical Clustering

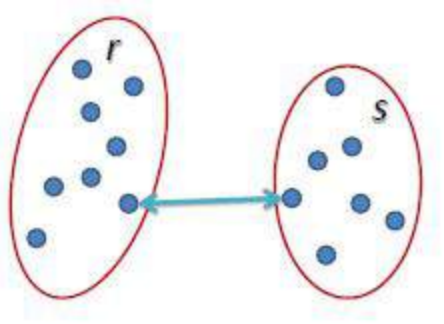
- Method
  - Divisive (Top-down)
  - Agglomerative (Bottom-up)

- Distance metrics
  - Single linkage
  - Complete linkage
  - Average linkage
  - Centroid
  - Medoid



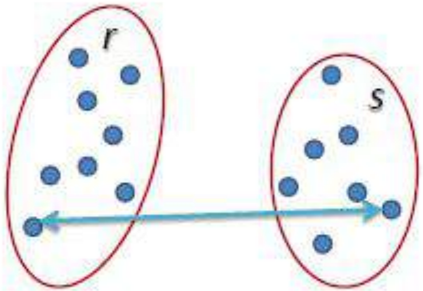
*Hierarchical Clustering

- Single Linkage



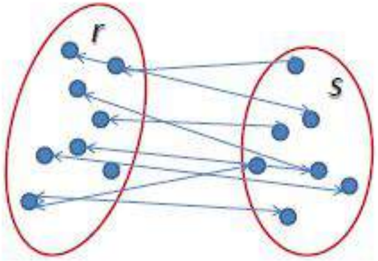
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

- Complete Linkage



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

- Average Linkage



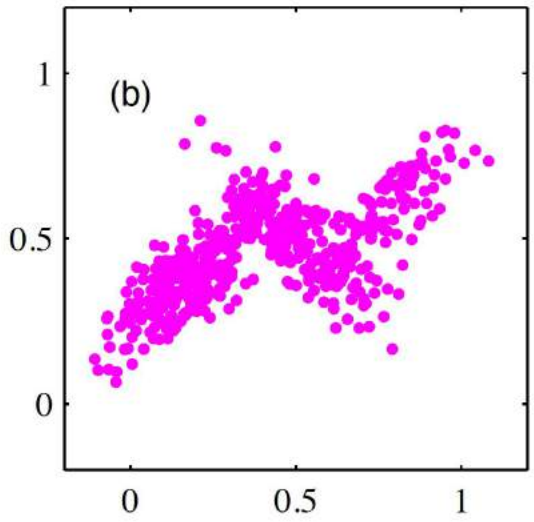
$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Gaussian Mixture Model

Probabilistic interpretation of clustering?

We can impose a probabilistic interpretation of our intuition that points stay close to their cluster centers

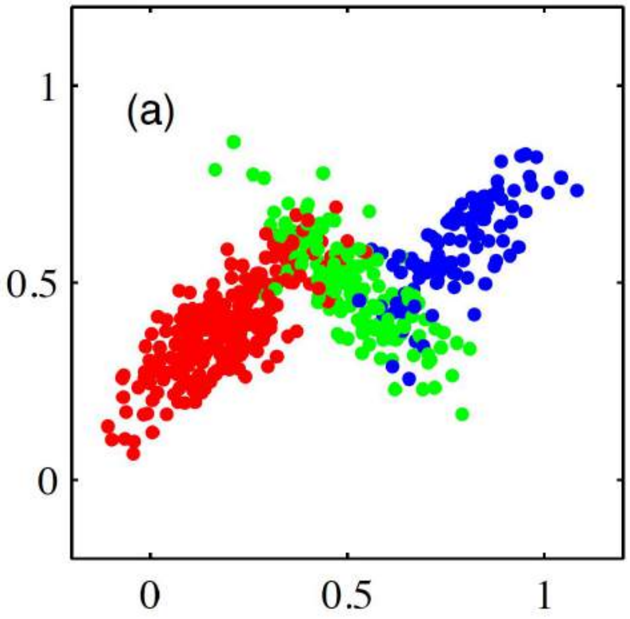
How can we model p(x) to reflect this?



Gaussian Mixture Model

Intuition

- We can model each region with a distinct distribution
- Common to use Gaussians, i.e.,
- Gaussian mixture models (GMMs) or mixture of Gaussians (MoGs).
- We don't know cluster assignments (label) or parameters of Gaussians or mixture components



Gaussian Mixture Model

Gaussian mixture models: formal definition

$\Theta = \{ \omega_k \quad \underline{\mu_k \quad \Sigma_k} \}$

A Gaussian mixture model has the following density function for $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

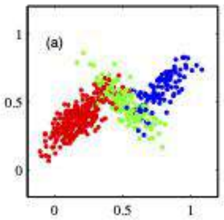
- K: the number of Gaussians — they are called (mixture) components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th component
- ω_k: mixture weights – they represent how much each component contributes to the final distribution. It satisfies two properties:

$$\forall\ k,\ \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

The properties ensure $p(\boldsymbol{x})$ is a properly normalized probability density function.

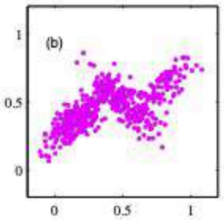
Gaussian Mixture Model

GMMs: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

$$p(\boldsymbol{x}|z = red) = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z = blue) = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z = green) = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$



The marginal distribution is thus

$$p(\boldsymbol{x}) = p(red)N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p(blue)N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$+ p(green)N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

GMM: Incomplete Data and EM algorithm

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$$

Incomplete Data

Our data contains observed and unobserved random variables, and hence is incomplete

- Observed: $\mathcal{D} = \{\boldsymbol{x}_n\}$
- Unobserved (hidden): $\{\boldsymbol{z}_n\}$

Goal Obtain the maximum likelihood estimate of $\boldsymbol{\theta}$:

$$\widehat{\boldsymbol{\theta}} = \arg\max \ell(\boldsymbol{\theta}) = \arg\max \log P(\mathcal{D}) = \arg\max \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta})$$

$$= \arg\max \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})$$

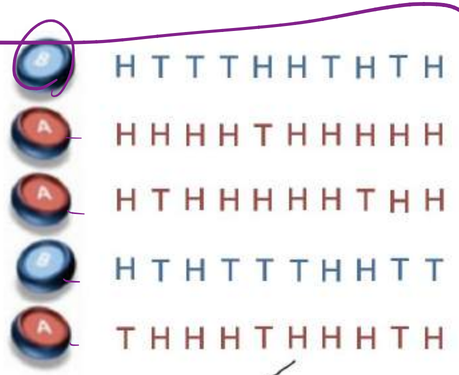
The objective function $\ell(\boldsymbol{\theta})$ is called the *incomplete* log-likelihood.

Typical EM iterations

1. Initialize θ with some values (random or otherwise)
2. Repeat
   a. **E-Step:** Compute γnk using the current θ
   b. **M-Step:** Update θ using the γnk we just computed
3. Until Convergence

EM Algorithms: Coin example (only M-step)

a Maximum likelihood



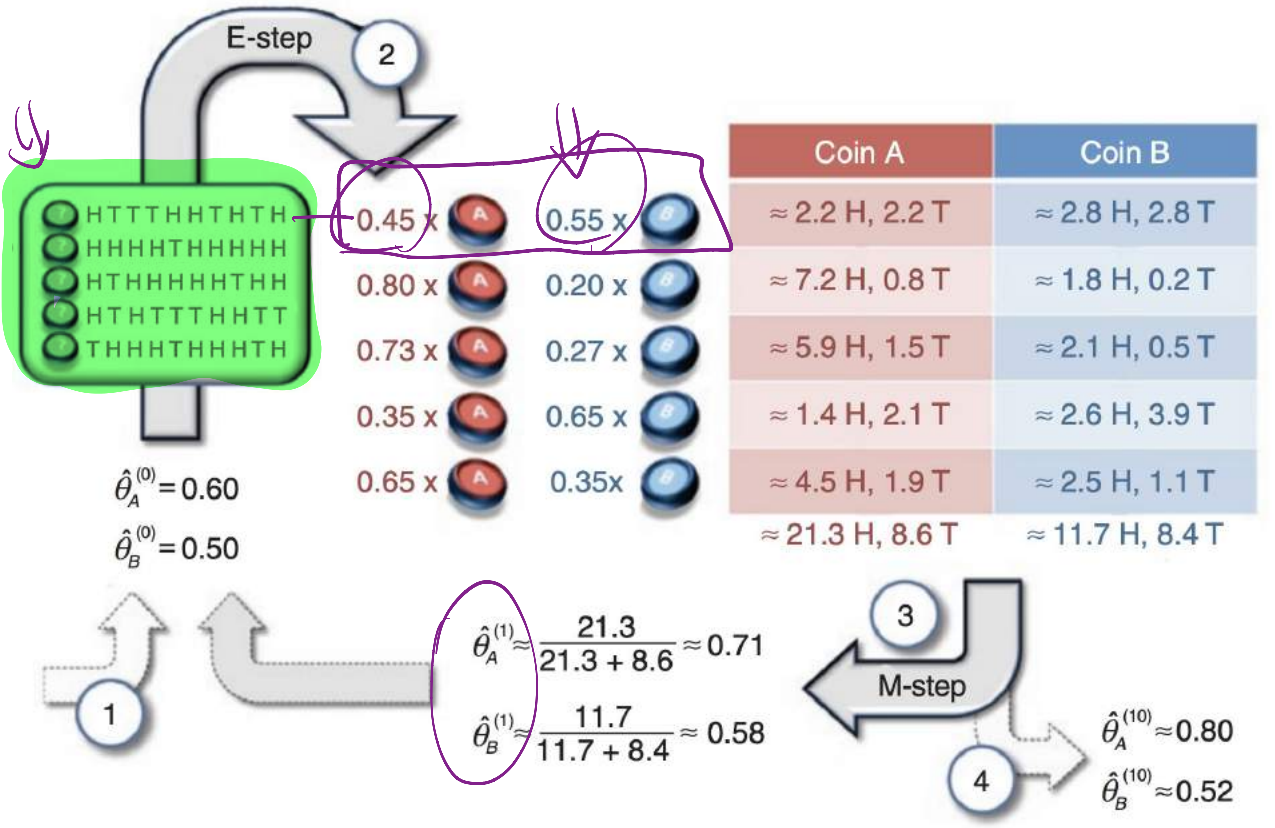
5 sets, 10 tosses per set

Coin A	Coin B
	5 H, 5 T
9 H, 1 T	
8 H, 2 T	
	4 H, 6 T
7 H, 3 T	
24 H, 6 T	9 H, 11 T

$$\hat{\theta}_A = \frac{24}{24+6} = 0.80$$

$$\hat{\theta}_B = \frac{9}{9+11} = 0.45$$

EM Algorithms: Coin example (EM)



GMM: E-Step

E-step: Soft cluster assignments

We define γ_{nk} as $p(z_n = k|\boldsymbol{x}_n, \boldsymbol{\theta})$

- This is the posterior distribution of z_n given $\boldsymbol{x}_n$ and $\boldsymbol{\theta}$
- Recall that in complete data setting γ_{nk} was binary
- Now it's a "soft" assignment of $\boldsymbol{x}_n$ to k-th component, with $\boldsymbol{x}_n$ assigned to each component with some probability

#k

$\gamma_{nk} = \begin{bmatrix} 0.2 & 0.5 & 0.3 \\ & & \end{bmatrix}$ #1

#n

Given $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, we can compute γ_{nk} using Bayes theorem:

$$\begin{aligned}
\gamma_{nk} &= p(z_n = k|\boldsymbol{x}_n) \\
&= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} \\
&= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^K p(\boldsymbol{x}_n|z_n = k')p(z_n = k')} \quad = \frac{\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\omega_k}{\sum_{k'=1}^K \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})\omega_{k'}}
\end{aligned}$$

GMM: M-Step

M-step: Maximimize complete likelihood

Recall definition of complete likelihood from earlier:

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \underset{\sigma^2}{\boldsymbol{\Sigma}_k}) \right\}$$

Previously γ_{nk} was binary, but now we define $\gamma_{nk} = p(z_n = k|\boldsymbol{x}_n)$ (E-step)

We get the same simple expression for the MLE as before!

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

Intuition: Each point now contributes some fractional component to each of the parameters, with weights determined by γ_{nk}

GMM: Example (M-Step calculation)

$k=2$

Consider clustering ID data with a mixture of **2 gaussian**. You're given the 1-D data points **x = [1 2 20 40]**. Suppose the E step is the following matrix : 2

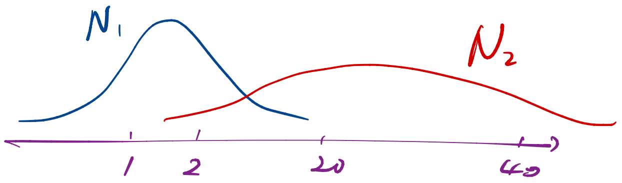
$\#C_i$

$$\gamma_{nk} = \#n \quad \begin{bmatrix} 0.5 & 0.5 \\ 0.2 & 0.8 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

4

$k=1$

$$w_k = \sum_{i=1}^{N} \frac{\gamma_{ik}}{N}$$



$$\mu_k = \frac{\sum_{i=1}^{N} \gamma_{ik} x_i}{\sum_{i=1}^{N} \gamma_{ik}}$$

→ What's the mixing weights after M-step?

→ What's the new values of means after M-step?

$w_1 + w_2 = 1$

$k=2$

$w_k =$

$$w_1 = \frac{0.5+0.2+0+1}{4} =$$

$$w_2 = \frac{0.5+0.8+1+0}{4} =$$

$\mu_k =$

$$\mu_1 = \frac{0.5\times 1 + 0.2\times 2 + 0\times 3 + 1\times 40}{0.5 + 0.2 + 0 + 1}$$

μ_2

GMM: Cheatsheet

Expectation (E) Step:

Calculate $\forall i, k$

$$\hat{\gamma}_{ik} = \frac{\hat{\phi}_k \mathcal{N}(x_i \mid \hat{\mu}_k, \hat{\sigma}_k)}{\sum_{j=1}^{K} \hat{\phi}_j \mathcal{N}(x_i \mid \hat{\mu}_j, \hat{\sigma}_j)},$$

where $\hat{\gamma}_{ik}$ is the probability that x_i is generated by component C_k. Thus, $\hat{\gamma}_{ik} = p(C_k | x_i, \hat{\phi}, \hat{\mu}, \hat{\sigma})$.

Maximization (M) Step:

Using the $\hat{\gamma}_{ik}$ calculated in the expectation step, calculate the following in that order $\forall k$:

W_k

- $\hat{\phi}_k = \sum_{i=1}^{N} \frac{\hat{\gamma}_{ik}}{N}$
- $\hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} x_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$
- $\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} (x_i - \hat{\mu}_k)^2}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}.$

GMM: Questions

- How does GMM relate to K-means? What are the similarities and differences?
- Will the GMM optimization process converge? (connected to K-means)

General EM Algorithms

- The EM algorithm is used to find **(local) maximum likelihood parameters** of a statistical model in cases where the equations cannot be solved directly.
- Typically these models involve latent variables in addition to unknown parameters and known data observations.
- Example applied cases: K-Means, GMM
- Reading: http://ai.stanford.edu/~chuongdo/papers/em_tutorial.pdf

Thank you!

Q & A

UCLA Samueli Computer Science

Evaluation of Instruction

Reminder: You have until **Saturday, March 13 8:00 AM PST** to complete confidential evaluations for CSM146 and Dis 1C (Junheng).

UCLA Samueli Computer Science

CS M146 Discussion: Week 10
PCA, HMM, Final Review

Junheng Hao
Friday, 03/12/2021

Roadmap

- Announcement
- PCA
- HMM
- Q & A

Announcement

- **There is no quiz in Week 10.**
- **Problem set 4** released on CCLE, submission on Gradescope.
  - Please assign pages of your submission with corresponding problem set outline items on GradeScope.
  - You need to submit code and the results required by the problem set
  - Due on **today 11:59pm PST, Mar 12 (Friday)**
- **Final Exam: March 15 (Next Monday)**
  - "Quiz-like" exam, submission through GradeScope

Late Submission of PS and final exam will NOT be accepted!

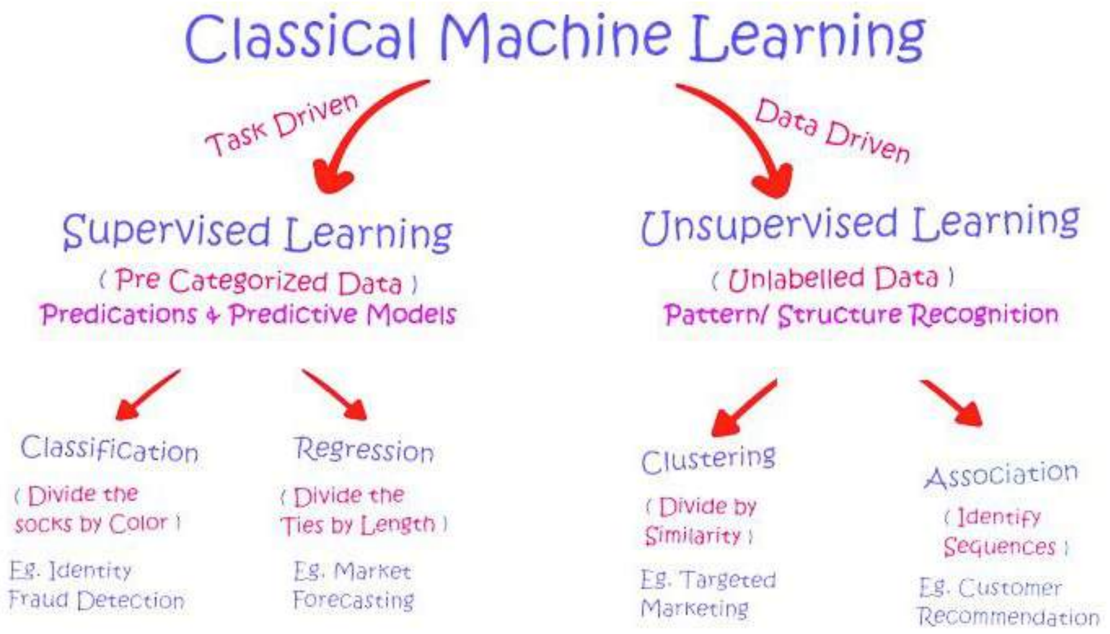
Updates: Final Exam

- Open book and open notes, on GradeScope: “quiz”-like exam
- Start attempting the exam from **8:00 am PST on March 15**; Submit your exam before **8:00am PST March 16** (No extensions). → 24h time window
- **You must start before 5:00am PST March 16 to use the full 3 hours. No late submission time.**
- Exam duration: **3 hours** (time limit after start the exam)
- Type: True/false and multiple choice questions (free text boxes are given for justification)
- The instructors will be available to provide clarifications on CampusWire (visible for everyone) from 8:00am-11:00am on March 15. Later questions on Campuswire may not be answered.
- Some calculations are expected.

MUST READ: Official post about final exam on Campuswire:
https://campuswire.com/c/GB5E561C3/feed/437

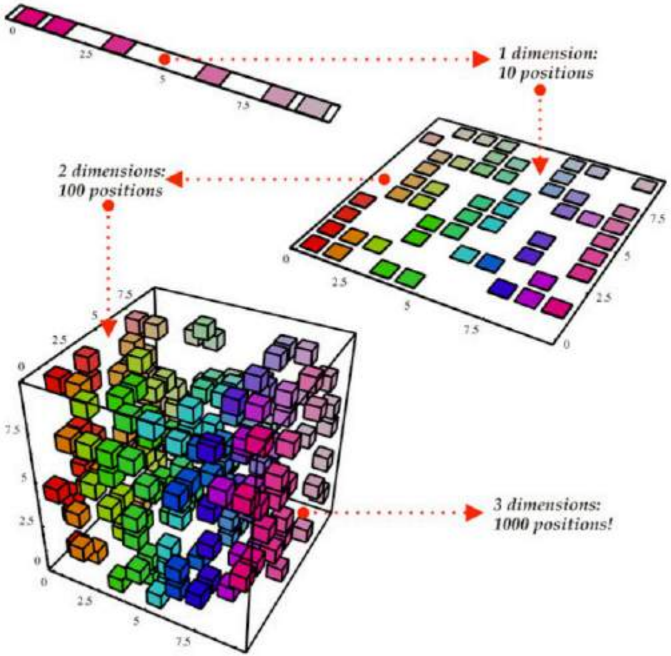
Supervised learning / Unsupervised learning

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

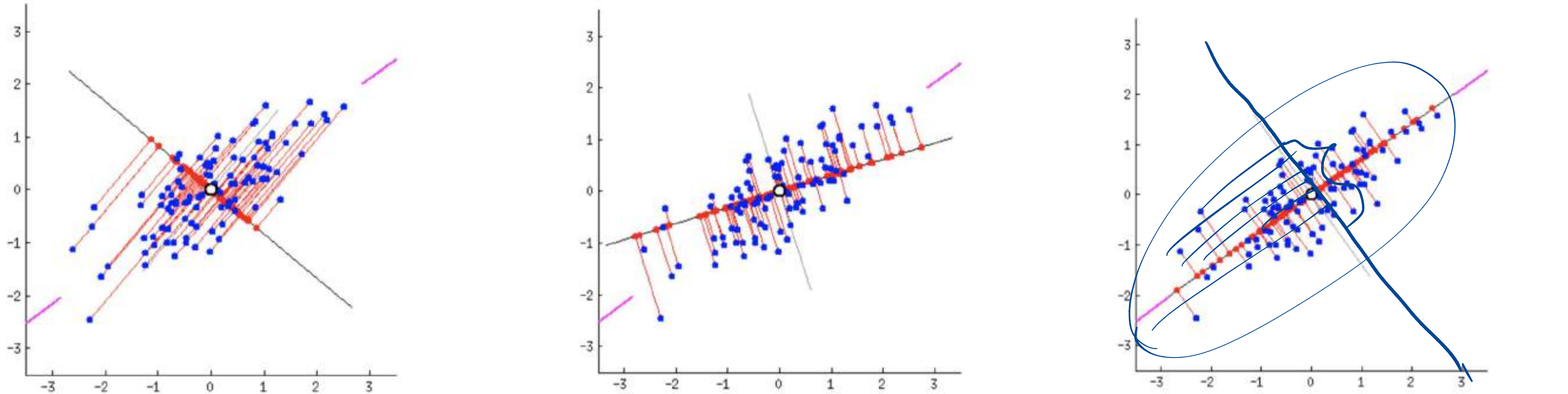


Large feature space and dimension reduction

- Disadvantages of having a large feature space
  - More data is required
  - Redundant features and more noise → Model overfitting
  - Algorithm's simplicity and fewer assumptions [Occam's razor]
- Straightforward dimensionality reduction
  - Feature elimination
  - Feature extraction



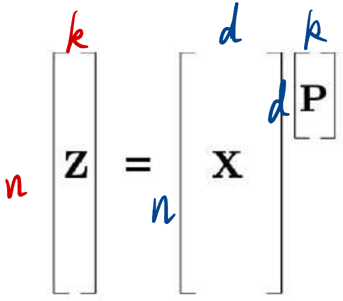
PCA



PCA: Formulation

764 → 3

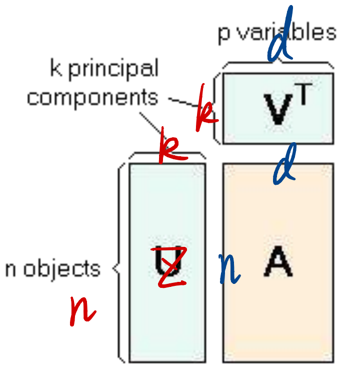
- Dimension reduction as matrix decomposition



$d \gg k$

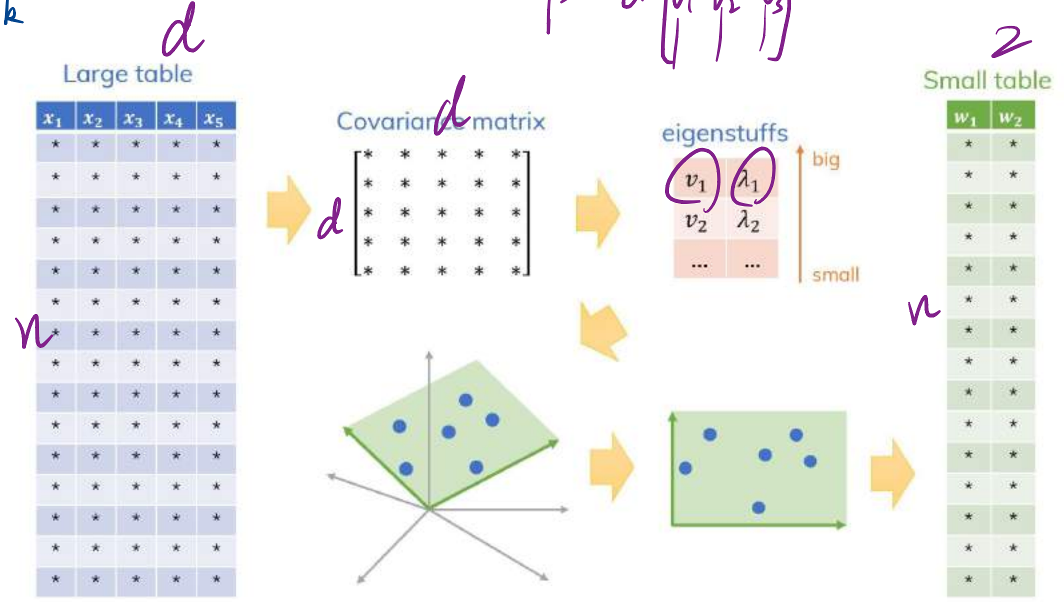
$Z = XP$

$P \quad d \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}$ (3 k)



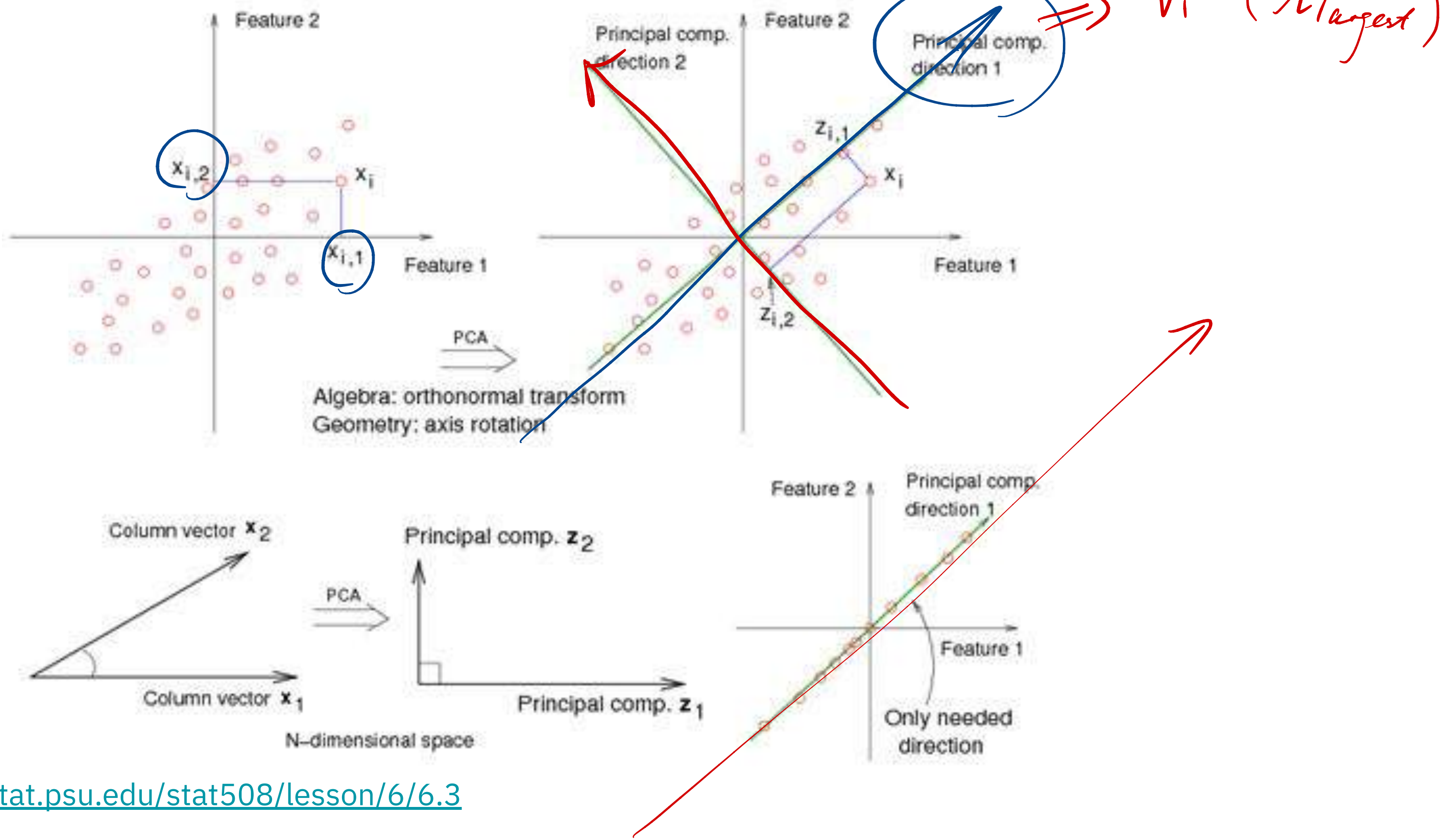
$X_{const} = Z P^T$

$A = U{\cdot}V^T$



PCA: Geometric Interpretation

$\vec{v}_1$ ($\lambda_{largest}$)



Credit: https://online.stat.psu.edu/stat508/lesson/6/6.3

PCA: Formulation in 1-dim (k=1)

- Demo shown in whiteboard

$$Z = Xp \qquad p \in \mathbb{R}^{d \times 1}$$

$$\sigma_z^2 = \frac{1}{n} \sum_{i=1}^{n} (z^{(i)})^2$$

$$= \frac{1}{n} (z^T z)$$

$$= \frac{1}{n} (Xp)^T (Xp)$$

$$(\|p\|_2 = 1)$$

$$\boxed{= \frac{1}{n} p^T \underbrace{(X^T X)}_{d \times d} p}$$

PCA Approach: Summarization

Steps:

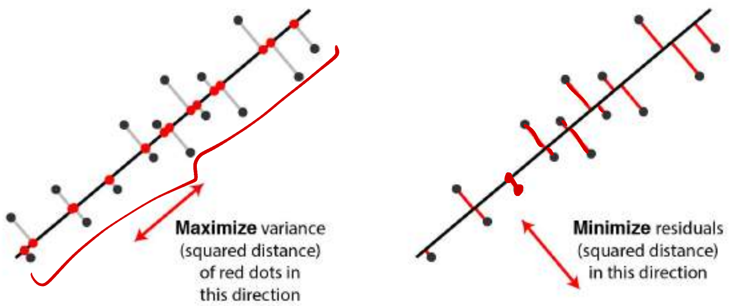
- Take the whole dataset consisting of ***d***-dimensional samples
- Compute the d-dimensional mean vector (i.e., the means for every dimension of the whole dataset)
- Compute the covariance matrix of the whole data set
  d×d
- Compute eigenvectors and corresponding eigenvalues
- Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a ***d×k*** dimensional matrix (where every column represents an eigenvector)
  p
- Use this ***d×k*** eigenvector matrix to transform the samples onto the new subspace.

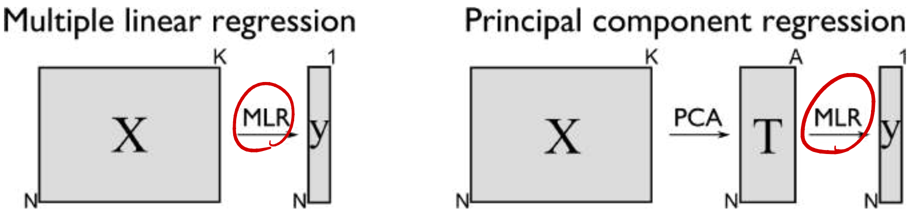
Credit: https://sebastianraschka.com/Articles/2014_pca_step_by_step.html

PCA: Example

- Demo shown at: https://sebastianraschka.com/Articles/2014_pca_step_by_step.html

PCA vs Linear Regression





HMM: Concepts

- Markov Process: the next state depends on the current state

$$P(x_{t+1}|x_1,\ldots,x_t) = P(x_{t+1}|x_t)$$

$$P(x_1,\ldots,x_t) = P(x_1)P(x_2|x_1)\ldots P(x_t|x_{t-1})$$

$P(x_1) \cdot P(x_2|x_1)\ P(x_3|x_2,x_1) = P(x_1,x_2,x_3)$

$P(x_3|x_2)$

- Initial probability

$\in R^K$

$$\pi_i = P(X_1 = i)$$

$\sum \pi_i = 1$

- Transition probability

$$q_{ij} = P(X_{t+1} = i|X_t = j)$$

$\in R^{K\times K}$

- Emission symbols

$$e_i(b) = P(Y_t = b|X_t = i)$$

$\in R^{K\times B}$

HMM: Computing probability of a sequence

- Assume uniform probability of starting in each states and transition probability matrix

current

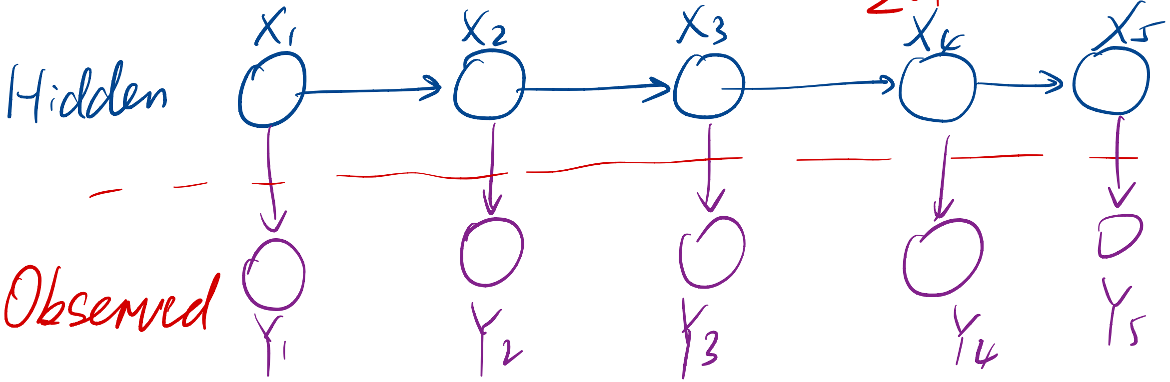
$$\text{next}\,\boldsymbol{Q} = \begin{pmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{pmatrix}$$

$\Sigma = 1$

$(X_1, X_2, X_3) = (1, 1, 3)$

$P(X_1, X_2, X_3)$

$= \pi(P(X_1)) \cdot T_{1\to1} \cdot T_{1\to3}$



HMM: Most probable path

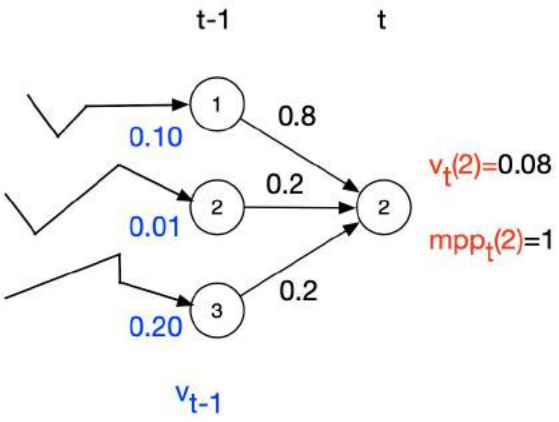
- Problem: Given a sequence of observations, what is the most probable sequence of hidden states?
- Solution: Viterbi Algorithm

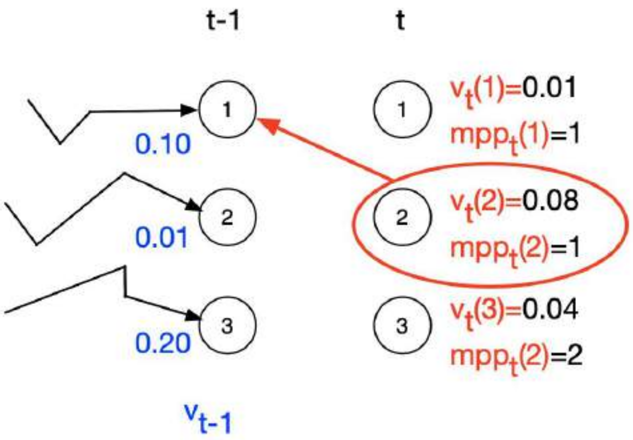
$Y_1, Y_2, \dots Y_n$

$\rightarrow X_1, \dots X_2 \quad X_n$

π $\quad E$

Q





HMM: Viterbi/Dynamic Programming

- The most probable path with last two states (*l, k*) is the most probable path with state *l* at time (*t*–1) followed by a transition from state *l* to state *k* and emitting the observation at time *t*.

$$
\begin{aligned}
& v_{t-1}(l)P(X_t = k|X_{t-1} = l)P(y_t|X_t = k) \\
& = v_{t-1}(l)q_{lk}e_t(y_t)
\end{aligned}
$$

- Maximization process

$$v_t(k) = \max_l v_{t-1}(l)q_{kl}e_t(y_t)$$

$$mpp_t(k) = l^*$$

$$l^* = \arg\max_l v_{t-1}(l)q_{kl}e_t(y_t)$$

HMM: Viterbi Algorithm Example [Link]

ABB

Transition probability distribution

Next

Current	A	B	End
Start	0.7	0.3	0
A	0.2	0.7	0.1
B	0.7	0.2	0.1

Current

$V_1(A) = 0.7$

$V_1(B) = 0.3$

$V_2(A) = 0.063$

$V_2(B) = 0.196$

Emission probability distribution

State	*S*	x	y
Start	1	0	0
A	0	0.4	0.6
B	0	0.3	0.7

Given a sequence of < *S*, x, y, y, *E* >

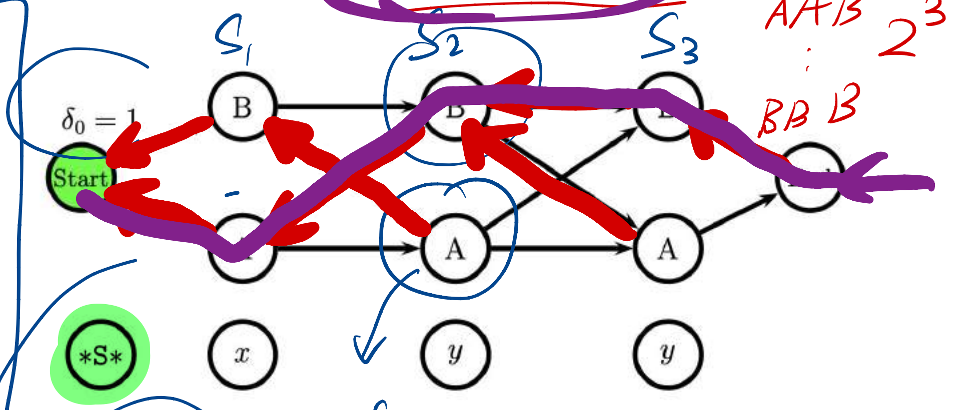
AAA

AAB

⋮

BBB

2^3



$V_2(A) = \max\{P(A|S_1) \cdot P(\text{"}x\text{"}|S_1) \cdot V_1(S_1)\}$

$S_1 = A$

$0.2 \times 0.4 \times 0.7$

0.056

$S_1 = B$

$0.7 \times 0.3 \times 0.3$

0.063

$$V_t(S) = \max_{S_{t-1}} P(S_t|S_{t-1}) \cdot P(w_{t-1}|S_{t-1}) \cdot V_{t-1}(S_{t-1})$$

$mpp_t(S) = \arg\max$

*HMM: Learning Parameters

- Solution: Baum–Welch algorithm

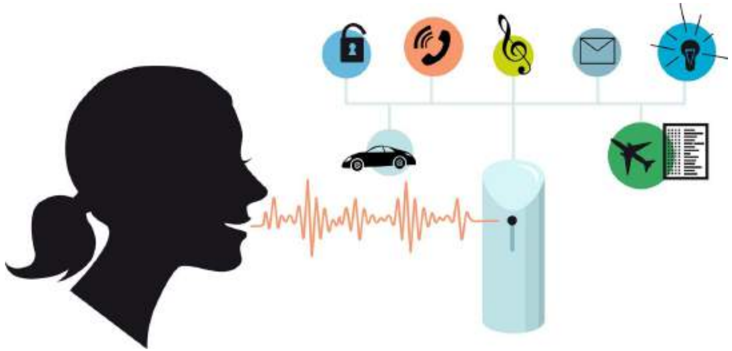
E

M

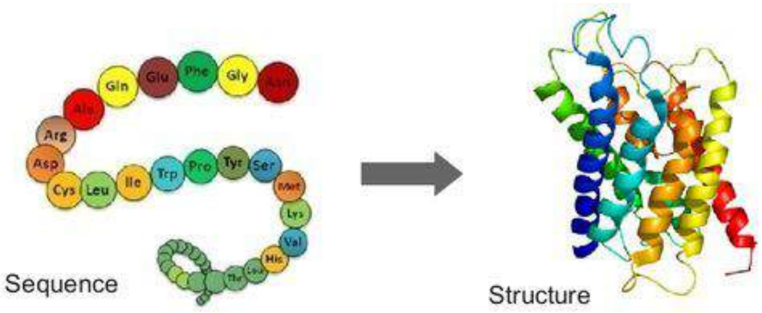
Estimated parameters to generate observations $\leftrightarrow$ Updated parameters from (multiple) observations

HMM: Applications and restrictions

Speak Recognition



Protein Structure Prediction



CSM 146: Summary

	Supervised Learning	Unsupervised Learning
Model	Decision tree, kNN Neural nets,	K-means, GMM PCA, HMM
Loss Function	0/1, square, hinge, exponential, cross entropy (log)	
Optimization	MLP, MAE, SVM (dual problem, constrained) Gradient descent (batch / stochastic) EM algorithms	
Theory	PAC learning, VC-dimension	
Others	Convexity/concavity, hyperparameters, overfitting and underfitting, inductive biases, regularizations	

Advanced Classes in AI/ML+

1. CS174A: Introduction to Computer Graphics (Prof. Asish Law, etc)
2. CS247: Advanced data mining (Prof. Yizhou Sun)
3. CS240: Big data seminar / Graph neural network (Prof. Yizhou Sun / Wei Wang)
4. CS260: Machine learning algorithms (Prof. Quanquan Gu/ Prof/ Cho-Jui Hsieh)
5. CS22X: Algorithms in Bioinformatics / Advanced Computational Genetics / Computational Methods in Medicine (Prof. Sankararaman / Prof. Eskin)
6. CS263: Natural language processing (Prof. Kai-Wei Chang / Nanyun Peng)
7. CS269: Seminars in deep learning foundations / natural language processing, etc. (Prof. Quanquan Gu / Kai-Wei Chang / Nanyun Peng)

Other courses are taught in EE, Stats departments:

1. ECE 236B/C: Convex Optimization (Prof. Vandenberghe)
2. ECE 239AS Reinforcement Learning Theory and Applications / Neural networks (Prof. Lin F. Yang / Jonathan Kao)

End-of-quarter Congratulations!

UCLA Samueli Computer Science

Thank you

for learning with us in winter 2021.

Good luck

as certified ML experts!

UCLA Samueli Computer Science

Evaluation of Instruction

Reminder: You have until **Saturday, March 13 8:00 AM PST** to complete confidential evaluations for CSM146 and Dis 1C (Junheng).

Math Backup: Eigendecomposition

$$
\begin{array}{cccc}
\mathbf{A} & \mathbf{Q} & \mathbf{\Lambda} & \mathbf{Q}^{-1}
\end{array}
$$

$$
\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}
=
\underbrace{\begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ | & | & | \end{bmatrix}}_{\text{Eigen vectors of } \mathbf{A}}
\underbrace{\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}}_{\text{Eigen values of } \mathbf{A}}
\underbrace{\begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ | & | & | \end{bmatrix}^{-1}}_{\text{Eigen vectors of } \mathbf{A}}
$$