Spring 2024 6.s079

Review for Quiz 1

Topics Covered

- Relational model
- SQL
- Pandas
- Regular Expression
- Text similarity
- Entity resolution
- Missing values / data cleaning
- Clustering
  - K-Means, Agglomerative, DB Scan, etc.
- Maximum Likelihood Est. (MLE)
- Expectation-Maximization (EM)
- Classifiers
  - KNN, Linear, SVM, etc.
- Feature design (one hot, etc)
- Bias / Variance
- Metrics
  - Precision, Recall, F1, etc
- Embeddings

SQL

Consider the following schemas about a social media platform:

```
Users(uid, name, email)
Posts(pid, uid, content, date) % uid refs Users.uid
Comments(cid, pid, uid, content, date) % pid refs Posts.pid,
                                         uid refs Users.uid
Likes(lid, pid, uid) % pid refs Posts.pid, uid refs Users.uid
```

SQL

Users(uid, name, email)
Posts(pid, uid, content, date) % uid refs Users.uid
Comments(cid, pid, uid, content, date) % pid refs Posts.pid,
uid refs Users.uid
Likes(lid, pid, uid) % pid refs Posts.pid, uid refs Users.uid

Complete the query

Find the top-5 most active users (user activity is defined by # posts), assuming there will not be ties and at least 5 users posted.

```
SELECT uid, name, ________ as nposts
FROM Users u, Posts p
WHERE ____________________
GROUP BY uid
_______________
______________
```

SQL

Solution

```
Users(uid, name, email)
Posts(pid, uid, content, date) % uid refs Users.uid
Comments(cid, pid, uid, content, date) % pid refs Posts.pid,
                                         uid refs Users.uid
Likes(lid, pid, uid) % pid refs Posts.pid, uid refs Users.uid
```

```
SELECT uid, name, count(*) as nposts
FROM Users u, Posts p
WHERE u.uid = p.uid
GROUP BY uid
ORDER BY nposts DESC
LIMIT 5
```

Users(uid, name, email)
Posts(pid, uid, content, date) % uid refs Users.uid
Comments(cid, pid, uid, content, date) % pid refs Posts.pid, uid refs Users.uid
Likes(lid, pid, uid) % pid refs Posts.pid, uid refs Users.uid

SQL

Complete the query

Find the number of comments for each post. For posts without comments, their count should be zero.

```
SELECT pid, p.content, ________ as ncomments
FROM Posts p ______ Comments c ON ______
GROUP BY pid
```

SQL

Users(uid, name, email)
Posts(pid, uid, content, date) % uid refs Users.uid
Comments(cid, pid, uid, content, date) % pid refs Posts.pid, uid refs Users.uid
Likes(lid, pid, uid) % pid refs Posts.pid, uid refs Users.uid

Solution

```
SELECT pid, p.content, COUNT(cid) as ncomments
FROM Posts p LEFT JOIN Comments c ON p.pid=c.pid
GROUP BY pid
```

Note: the following would not work

```
SELECT pid, p.content, COUNT(*) as ncomments
FROM Posts p LEFT JOIN Comments c ON p.pid=c.pid
GROUP BY pid
```

Posts

pid	uid	…
1	1	…
2	1	…

Comments

cid	pid	…
1	1	…
2	1	…

Posts LEFT JOIN Comments on pid

pid	cid	…
1	1	…
1	2	…
2	NULL	…

SQL

Users(uid, name, email)
Posts(pid, uid, content, date) % uid refs Users.uid
Comments(cid, pid, uid, content, date) % pid refs Posts.pid, uid refs Users.uid
Likes(lid, pid, uid) % pid refs Posts.pid, uid refs Users.uid

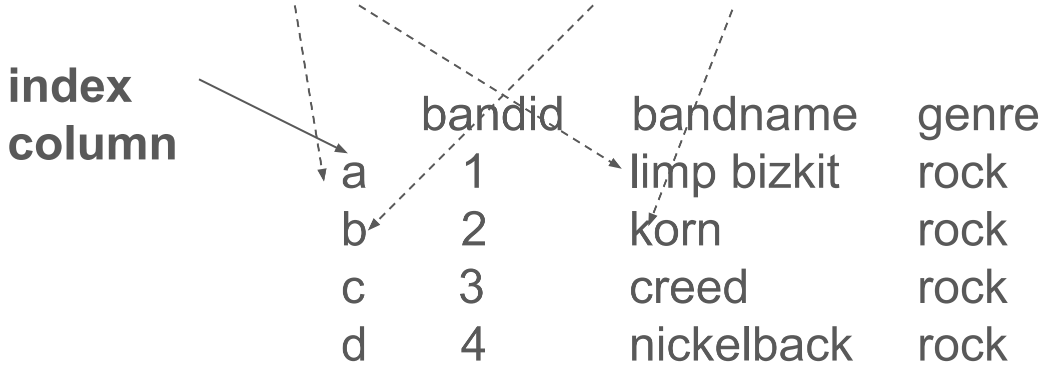
Take-Home practice

1. For each user, find the most popular post (most likes). Break ties by using post's date (more recent one wins).

It also helps to walk through lab1

Pandas

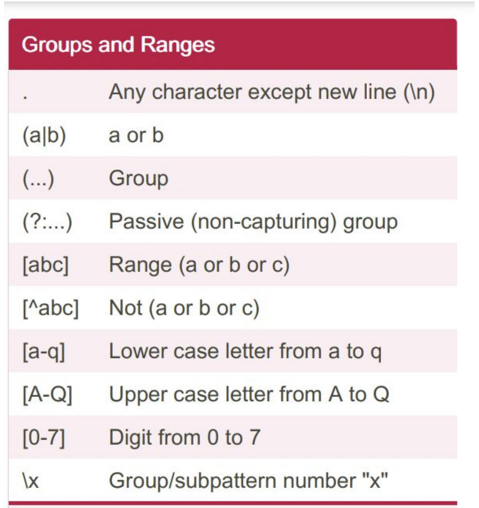
- Anything in SQL can be done in Pandas
- Pandas additionally adds
  - loc - index-based lookup
  - iloc - array-offset-based lookup
-
- `print(df.iloc[0, 1], df.loc[“b”, "bandname"])`



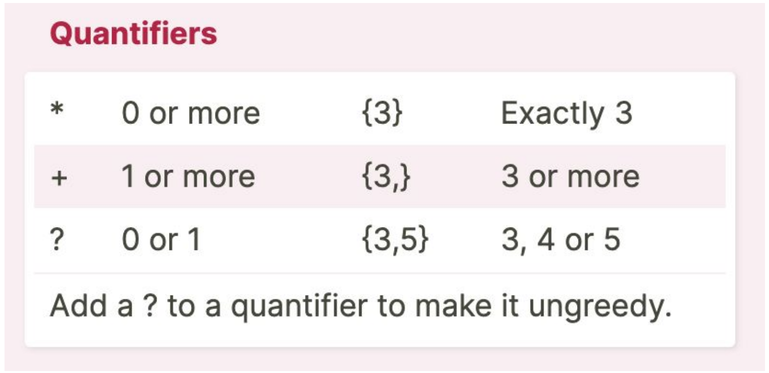
	bandid	bandname	genre
a	1	limp bizkit	rock
b	2	korn	rock
c	3	creed	rock
d	4	nickelback	rock

Regex

- Check slides for syntax
- Write a regular expression for matching
    - Posts with valid Hashtags or Mentions: only alphanumerics are allowed after hashtags/mentions
        - "Just saw Tim the Beaver at the **#MIT** football game! 🏈 **#GoBeavers @MITAthletics**"
        - "Had an amazing encounter with #   Tim the Beaver today! He even posed for a photo 📸"



Groups and Ranges	
.	Any character except new line (\n)
(a|b)	a or b
(...)	Group
(?:...)	Passive (non-capturing) group
[abc]	Range (a or b or c)
[^abc]	Not (a or b or c)
[a-q]	Lower case letter from a to q
[A-Q]	Upper case letter from A to Q
[0-7]	Digit from 0 to 7
\x	Group/subpattern number "x"



Quantifiers

*	0 or more	{3}	Exactly 3
+	1 or more	{3,}	3 or more
?	0 or 1	{3,5}	3, 4 or 5

Add a ? to a quantifier to make it ungreedy.

Regex

- Solution
  - #[a-zA-Z0-9]+|@[a-zA-Z0-9]+
- Take-Home practices
    - Posts where “Tim” shows up at least twice
    - Posts with URLs
- Have a REGEX syntax cheatsheet is helpful.
  - https://cheatography.com/davechild/cheat-sheets/regular-expressions/
- Also helpful to walk through lab 2 on sed/grep/awk

Text Similarity

- Jaccard Similarity for two sets of words
  - Jaccard(S1, S2) = |S1∩S2| / |S1 U S2|
- Cosine Similarity for vectors
  - CosSim(V1, V2) = Cos(Θ) = V1 • V2 / (||V1|| * ||V2||)
- See slides and lab 4

id	completed	age	income	YearsOfExp.

Missing Values

- Types of Missing Values
  - Missing Completely At Random (MCAR)
    - Missingness occur randomly, not correlating to any columns in the dataset
    - e.g., missingness caused by poor internet connections
  - Missing At Random (MAR)
    - Missingness correlates to certain columns in the dataset
    - e.g., Older people less likely to respond to web survey, so records where age is higher tend to have more missing values in other columns.
  - Missing Not At Random (MNAR)
    - Missingness correlates to the missing values themselves
    - E.g., Rich/poor people tend not to report income
- Missing Value handling
  - Deletion-based
  - Imputation-based

Missing Values

- Deletion-based
  - Listwise Deletion: delete records with any missing values
  - Pairwise Deletion: delete records with missing values on columns needed for analysis
- Comparison
  - Advantages
    - Pairwise deletion keeps as many records as possible
    - Listwise deletion is simple
  - Disadvantages
    - Listwise deletion reduces the sample size more aggressively
    - Pairwise deletion might have different sample size

Missing Values

- Imputation-based
  - Mean substitution
    - + Simple
    - - Reduces variability, weakens correlations, biases data
  - Regression methods
    - + Emphasize correlations present in the data
    - - Complicated and not useful when there is no correlation
  - Sampling from a reasonable distribution
  - Multiple imputation

Clustering

We primarily discussed three clustering algorithms:

- K-Means
- Agglomerative Clustering
- DBScan

Clustering – K-means

K-means clustering is simple and effective at getting an initial clustering

Algorithm:

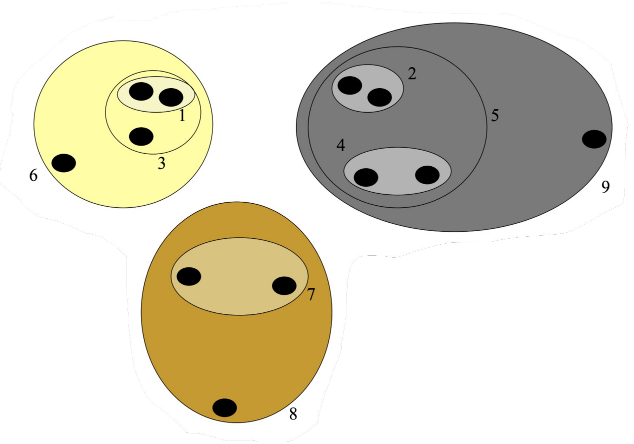
- The user chooses some integer value K (the number of clusters)
- Initialize K cluster centers at random
  - Can be random points in d-dimensional space (where d is the data dimensionality)
- Iterate:
  - Assign each data point to the nearest cluster center
  - Compute each cluster’s centroid
  - Update each cluster center to be the centroid
- Terminate on some condition (e.g. when cluster centers stay mostly static)

Clustering – Agglomerative Clustering

Agglomerative Clustering naturally creates a (nested) hierarchy of clusters

Algorithm:

- Start with each point as its own cluster (i.e. you have k = |D| clusters)
- Iterate:
    - Compute which two clusters are the closest
        - E.g., using single, complete, or average linkage
    - Merge the two closest clusters into a single cluster
- Terminate:
    - on some condition
        - e.g. when you have k' clusters
        - e.g. when min. cluster distance > some value



Clustering – DBScan

Algorithm:

```
Input: The data set D

Parameter: ε, MinPts

For each object p in D
    if p is a core object and not processed then
        C = retrieve all objects density-reachable from p
        mark all objects in C as processed
        report C as a cluster
    else mark p as outlier
    end if

End For
```

Epsilon-neighborhood: all points within a radius ε from a given point

High Density (point): epsilon-neighborhood of point contains at least **MinPts** points

Core Point: a high density point

Border Point: a non-core point in the epsilon-neighborhood of a core point

Outlier (Noise) Point: a point that is not a core point nor a border point

Clustering – DBScan (cont.)

Algorithm:

```
Input: The data set D

Parameter: ε, MinPts

For each object p in D
    if p is a core object and not processed then
        C = retrieve all objects density-reachable from p
        mark all objects in C as processed
        report C as a cluster
    else mark p as outlier
    end if

End For
```

Directly Density-Reachable: a point q is directly density-reachable from a point p iff q is within p's epsilon-neighborhood (and p is a core point)

Density-Reachable: a point q is density-reachable from p if there is a chain of points p, q1, q2, …, q such that q_(i+1) is **directly density-reachable** from q_i

Note that density-reachability is not symmetric

Clustering – Overview

K-means:

- **Pros/Cons:** easy to implement, requires user to set K
- **Works well when:** you have spherical clusters of same size and roughly equal number of points

Agglomerative Clustering:

- **Pros/Cons:** linkage provides some control over cluster shapes, requires user to set K (or equivalent)
- **Works well when:** your choice of linkage can map well to some intuition about the dataset

DBScan

- **Pros/Cons:** does better job of identifying clusters of varying shapes and sizes, don’t need to specify k, requires tuning hyperparameters, still sensitive to varying densities
- **Works well when:** you don’t know # of clusters ahead of time; clusters have contiguous shapes of similar density

Clustering – Self-Test Ideas

- Walk yourself through K-means / Agg. Clustering / DBScan on a **small** hand-crafted dataset
- Create a **small** dataset where K-means / Agg. Clustering / DBScan will not work well

Maximum Likelihood Estimation (MLE)

- **MLE Key Idea:** Given a model L(Θ, X) and dataset X = {x1, …, xn}, learn the model parameters Θ’ which maximize the (log) likelihood of observing the dataset X

- L is a likelihood function, i.e.: $\mathcal{L}(\theta \mid x) = p_\theta(x) = P_\theta(X = x)$

- If we assume X is sampled i.i.d., then: $\mathcal{L}(\theta, X) = p_\theta(X) = \prod_i^n p_\theta(x_i)$

Note: L(Θ|x) is often written as L(Θ,x) or L(Θ; x)

- The maximum likelihood estimator is: $\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \, \mathcal{L}(\theta, X)$

Maximum Likelihood Estimation (MLE) – TMI

- Suppose we have a coin with an unknown probability θ of coming up heads
- We flip the coin N times and observe N_h heads and N_t tails (N = N_h + N_t)
- If we assume L(θ, X) is a binomial distribution, i.e.:

$$P(D \mid \theta) = \binom{n_H + n_T}{n_H} \theta^{n_H} (1-\theta)^{n_T} \longrightarrow$$

Returning to our binomial distribution, we can now plug in the definition and compute the log-likelihood:

$$\begin{aligned}
\hat{\theta}_{MLE} &= \underset{\theta}{\operatorname{argmax}}\; P(D; \theta) \\
&= \underset{\theta}{\operatorname{argmax}} \binom{n_H + n_T}{n_H} \theta^{n_H} (1-\theta)^{n_T} \\
&= \underset{\theta}{\operatorname{argmax}}\; \log \binom{n_H + n_T}{n_H} + n_H \cdot \log(\theta) + n_T \cdot \log(1-\theta) \\
&= \underset{\theta}{\operatorname{argmax}}\; n_H \cdot \log(\theta) + n_T \cdot \log(1-\theta)
\end{aligned}$$

We can then solve for θ by taking the derivative and equating it with zero. This results in

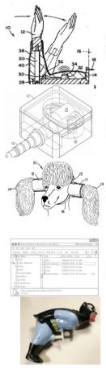
$$\frac{n_H}{\theta} = \frac{n_T}{1-\theta} \Longrightarrow n_H - n_H\theta = n_T\theta \Longrightarrow \theta = \frac{n_H}{n_H + n_T}$$

Example problem from Prof. Kilian Weinberger's excellent course notes for CS 4780: https://www.cs.cornell.edu/courses/cs4780/2023sp/lectures/lecturenote04.html

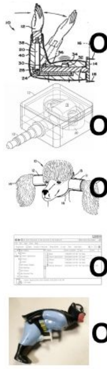
Expectation-Maximization (EM) Example

Latent (hidden) Variables

$$\boxed{Z_1 \quad Z_2 \quad Z_3 \quad Z_4 \quad Z_5}$$



$$\begin{array}{c} o_1 \\ o_2 \\ o_3 \\ o_4 \\ o_5 \end{array} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$



$$T(n) = \begin{array}{c} o_1 \\ o_2 \\ o_3 \\ o_4 \\ o_5 \end{array} \begin{pmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{pmatrix}$$

Maximum Likelihood Estimation (MLE) for EM

- **MLE Key Idea:** Given a model L(Θ, X) a dataset X = {x1, …, xn}, **and a set of latent data or missing values Z**, learn the model parameters Θ' which maximize the (log) **marginal** likelihood of the observed dataset X

- L is a likelihood function, i.e.: $L(\boldsymbol{\theta}; \mathbf{X}, \mathbf{Z}) = p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$

- If we assume X is sampled i.i.d., then: $\mathcal{L}(\theta, X, Z) = p_\theta(X, Z) = \prod_i^n p_\theta(x_i, z)$

- The maximum likelihood estimator is: $\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \mathcal{L}(\theta, X) = \sum_z p_\theta(X, Z)$

Expectation-Maximization (EM) Algorithm

```
Initialize $\theta \in \Theta$
For t = 0,1,2,…
    $\boxed{\text{E-Step:}}$   Calculate the expected value
                of the log likelihood
                function, with respect to the
                conditional distribution of Z
                given X under the current
                estimate of the parameters $\theta_t$:
```

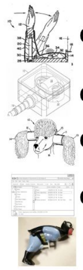
$$Q(Q|\theta_t) = E_{Z|X,\theta_t}[\log \mathcal{L}(\theta, X, Z)]$$

```
    $\boxed{\text{M-Step:}}$   Find the parameter that
                maximizes this quantity
```

$$\boldsymbol{\theta}^{(t+1)} = \boxed{\underset{\boldsymbol{\theta}}{\arg\max}}\boxed{\mathrm{E}_{\mathbf{Z}\sim p(\cdot|\mathbf{X},\boldsymbol{\theta}^{(t)})}[\log p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})]}$$

$$\theta_{t+1} = \underset{\theta}{\operatorname{argmax}}\, Q(Q|\theta_t)$$

Expectation-Maximization (EM) Example



$$\begin{array}{c} o_1 \\ o_2 \\ o_3 \\ o_4 \\ o_5 \end{array} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

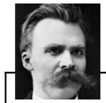
X (observed data)

Guess

True		Bogus	!Bogus
	Bogus	1	0
	!Bogus	0	1

Guess

True		Bogus	!Bogus
	Bogus	1	0
	!Bogus	0	1

	Bogus	!Bogus
Bogus	1	0
!Bogus	0	1

Guess

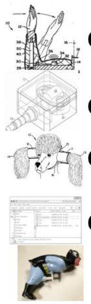
True		Bogus	!Bogus
	Bogus	1	0
	!Bogus	0	1

Guess

True		Bogus	!Bogus
	Bogus	1	0
	!Bogus	0	1

θ (unknown params)

EM Algorithm - Example

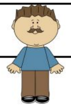


$$\begin{array}{c} o_1 \\ o_2 \\ o_3 \\ o_4 \\ o_5 \end{array} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

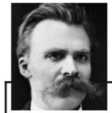
$$\underbrace{\begin{array}{cc} \text{Bogus} & \text{Not Bogus} \end{array}}_{\text{Z (latents; or missing values)}} \qquad \underbrace{\begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix}}_{L(\theta,X,Z)}$$

		Guess	
		Bogus	!Bogus
True	Bogus	1	0
	!Bogus	0	1

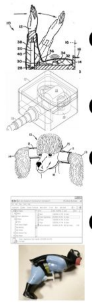
		Guess	
		Bogus	!Bogus
True	Bogus	1	0
	!Bogus	0	1

		Guess	
		Bogus	!Bogus
True	Bogus	1	0
	!Bogus	0	1

		Guess	
		Bogus	!Bogus
True	Bogus	1	0
	!Bogus	0	1

	Bogus	!Bogus
Bogus	1	0
!Bogus	0	1

EM Algorithm - Example



$$\begin{array}{c}o_1\\o_2\\o_3\\o_4\\o_5\end{array}\begin{pmatrix}1&1&1&0&0\\0&0&0&1&1\\0&1&1&0&0\\0&0&0&0&1\\0&1&1&1&0\end{pmatrix}$$

Z (latents)

L(θ,X,Z)

$$\begin{array}{cc}\text{Bogus}&\text{Not Bogus}\\\hline\end{array}$$

$$\begin{pmatrix}0.6&0.4\\0.4&0.6\\0.4&0.6\\0.2&0.8\\0.6&0.4\end{pmatrix}$$

"Correct" Labels

$$\begin{pmatrix}1\\0\\0\\0\\1\end{pmatrix}$$

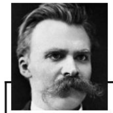
E[log(L(θ,X,Z))]

Guess

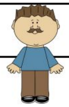
True		Bogus	!Bogus
	Bogus	1	0
	!Bogus	0	1

Guess

True		Bogus	!Bogus
	Bogus	1	0
	!Bogus	0	1



	Bogus	!Bogus
Bogus	1	0
!Bogus	0	1



Guess

True		Bogus	!Bogus
	Bogus	1	0
	!Bogus	0	1

Guess

True		Bogus	!Bogus
	Bogus	1	0
	!Bogus	0	1

Expectation-Maximization (EM) Algorithm

```
Initialize $\theta_0$
For t = 0,1,2,…
    E-Step:   Calculate the expected labels
              (e.g., bogus or not-bogus)
              given $\theta_t$

    M-Step:   Given the estimated label,
              optimize $\theta$ and set it to $\theta_{t+1}$
```

EM Algorithm – Self-Test Ideas

- Walk yourself through the example shown in lecture

Classifiers

We primarily discussed five classifiers:

- K-Nearest Neighbor (KNN)
- Support Vector Machines
- Decision Trees
- Random Forest
- (Gradient) Boosted Decision Trees

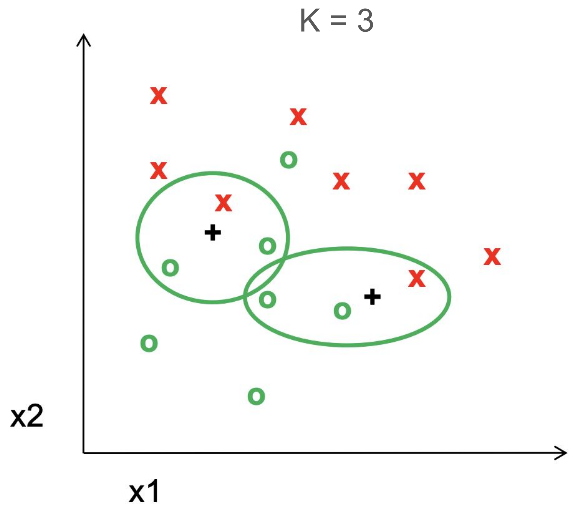
Assume we have a dataset D = {(x1, y1), (x2, y2), …, (xn, yn)}

Classifiers – KNN

Classifier: h(x) = max_freq({y_i : x_i is one of k-nearest points to x})

Notes:

- Easy to code and interpret
- O(|D|) inference time (requires evaluating distance function dist(x, x_i) on every sample in D)
- O(|D|) memory
- Does not scale well

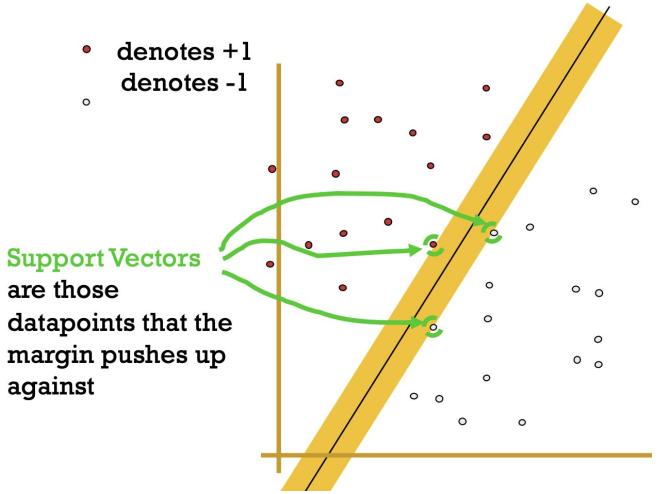


Classifiers – Support Vector Machines (SVMs)

Classifier: h(x) = sign(w^T · x + b)

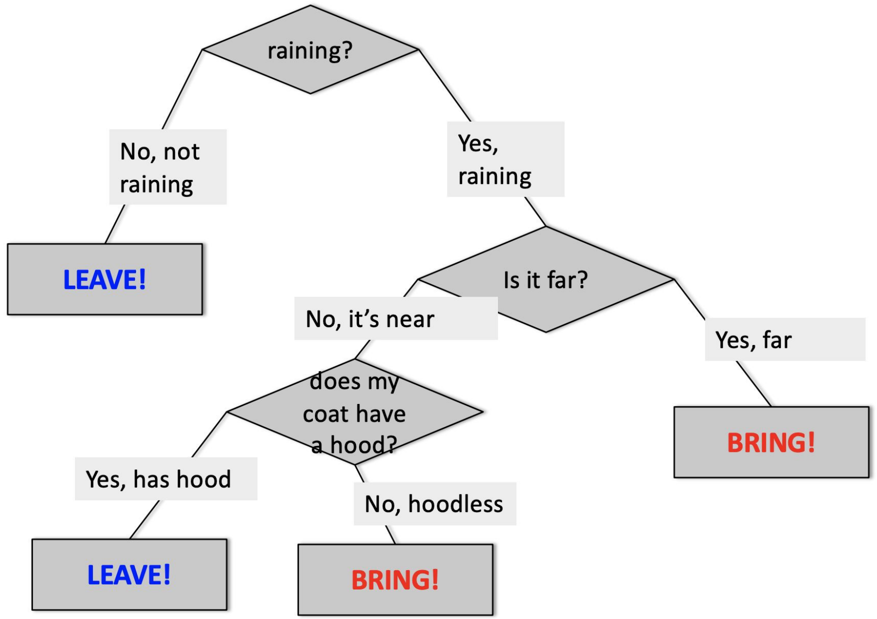
Notes:

- (L)SVM is the maximum margin linear classifier
- If data is not linearly separable, we can:
    - Use the “kernel trick” to project our data into higher-dim. space where it is linearly separable (i.e. Kernel SVM)
    - Regularize with a soft-margin classifier



Classifiers – Decision Trees

Classifier: h(x) = (traverse tree to leaf node based on x’s feature values)

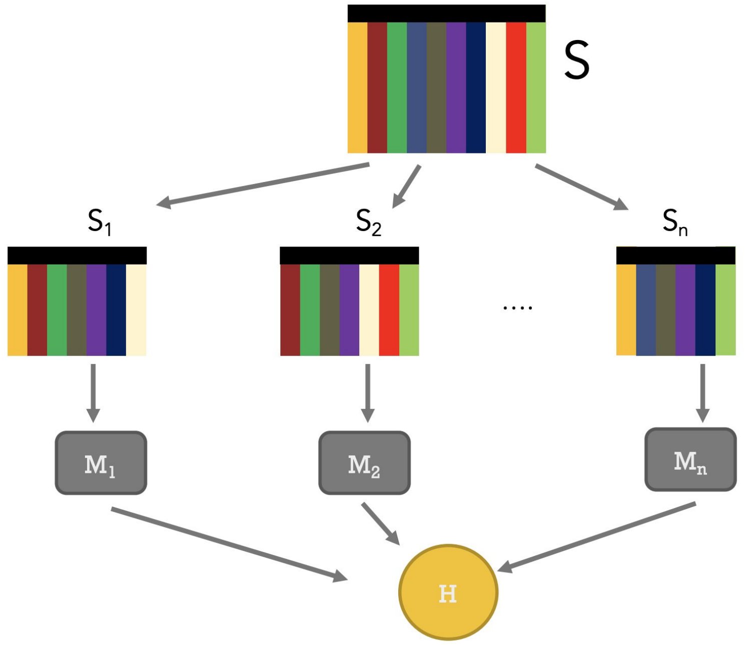


Classifiers – Random Forest

Classifier (classification): h(x) = max_freq({h_i(x) : ∀i ϵ [1, n]})

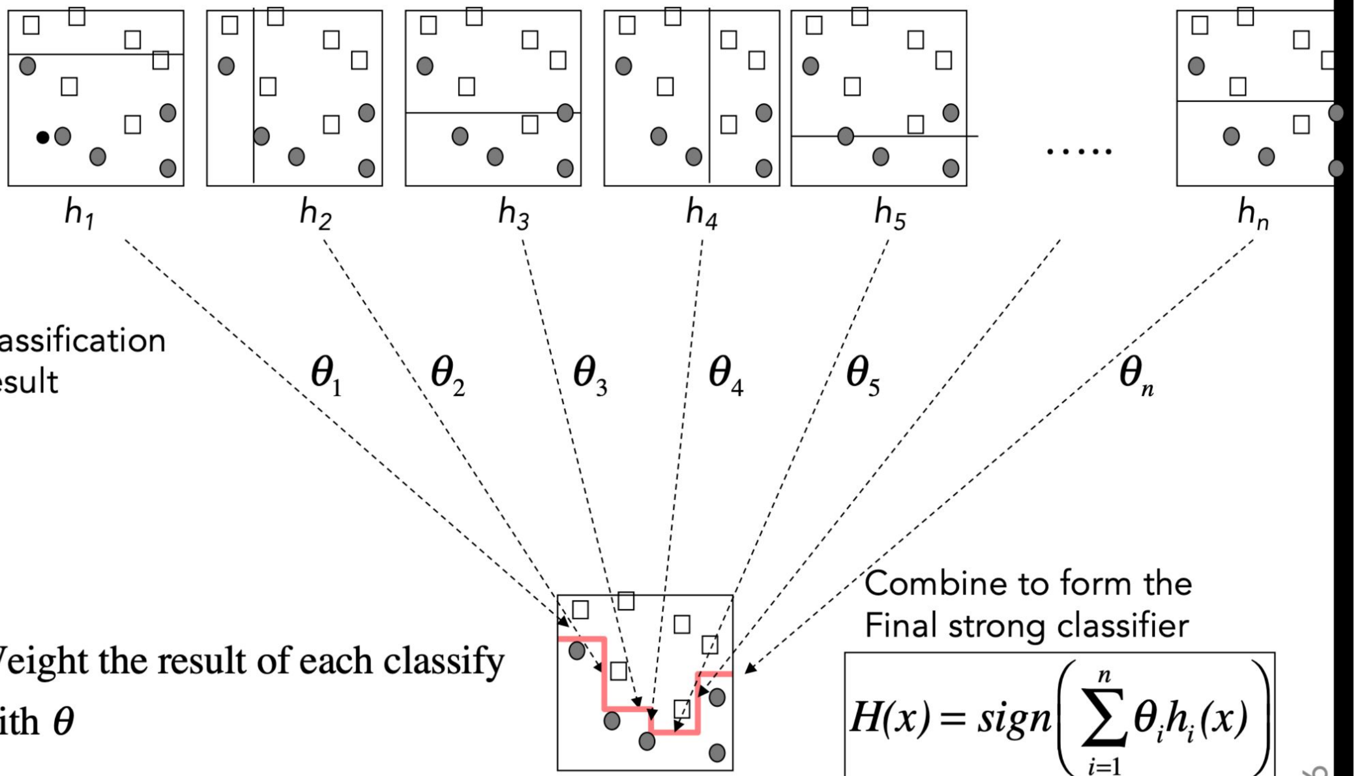
Notes:

- Sample subsets S_1, S_2, …, S_n from D with replacement
- For each subset S_i:
  - Sample k <= d features (w/out replacement)
  - Train a full decision tree h_i(x) on only those k features
- Final classifier is majority vote (or avg. for regression) over outputs from h_i(x)
- Works great out-of-the-box



ADABOOST - CORE IDEA

Take a set of weak classifiers (normally they should do better than guessing)



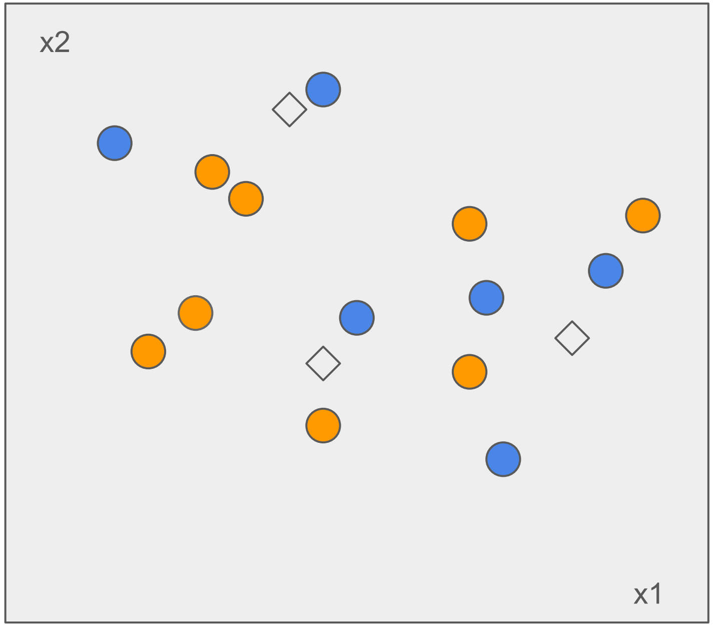
Classification Result

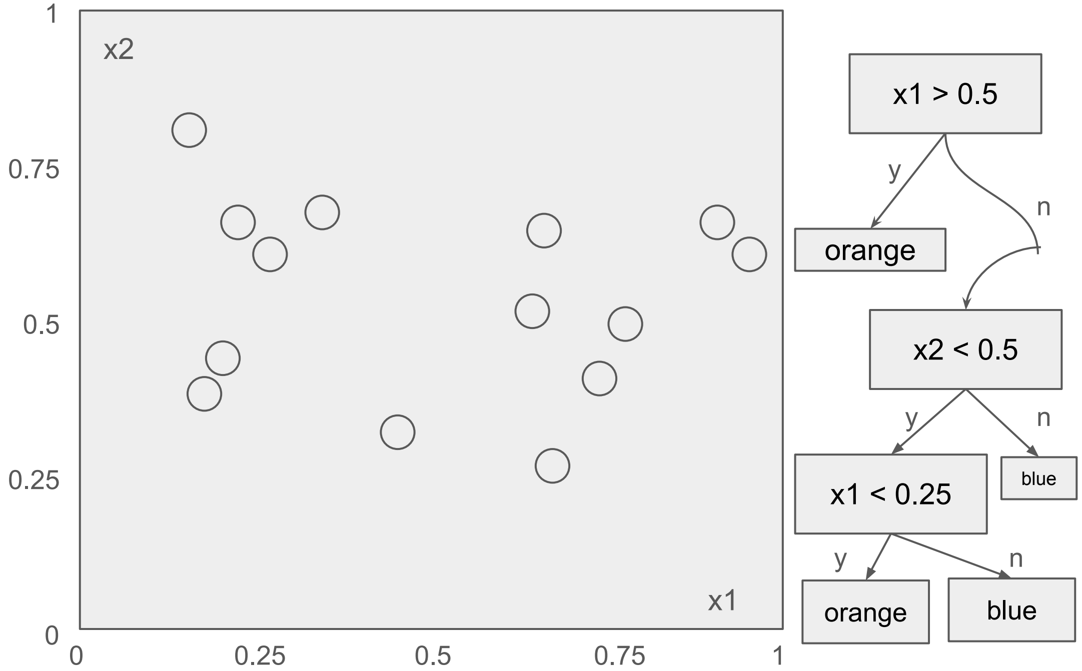
Weight the result of each classify with θ

Combine to form the Final strong classifier

$$\boxed{H(x) = sign\left(\sum_{i=1}^{n} \theta_i h_i(x)\right)}$$

Classifiers – Knowledge Check

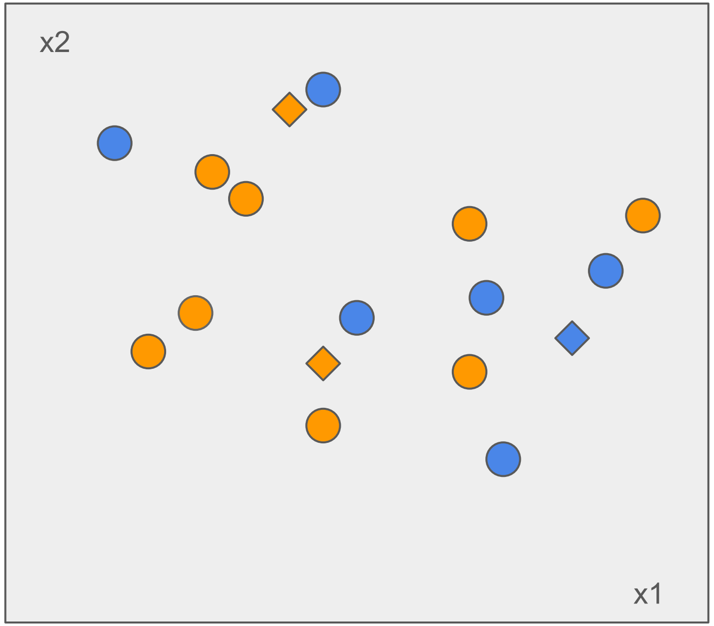


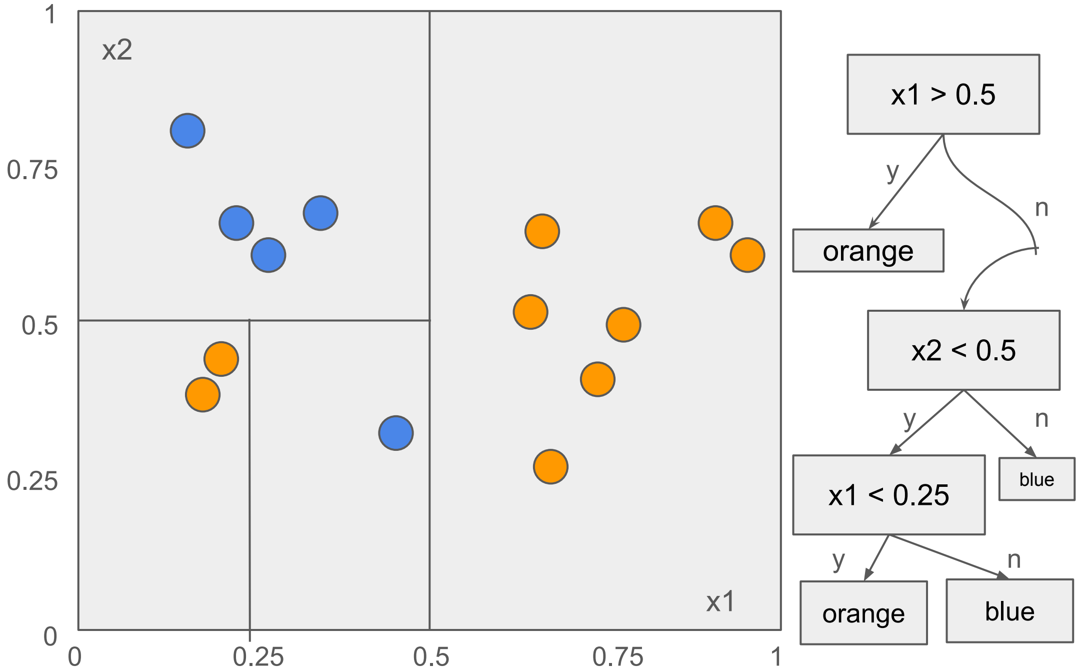


Q: Under KNN w/K=3, what is the color label for each rhombus (star)

Q: Draw the classification boundary for the decision tree on the right

Classifiers – Knowledge Check (answer)





Q: Under KNN w/K=3, what is the color label for each rhombus (star)

Q: Draw the classification boundary for the decision tree on the right

Feature Design

- Review slides 38-45 in Lecture 8 (Machine Learning II)
- Use slide 43 as a self-test to see if you can feature engineer / vectorize the data in the table

Bias and Variance

Bias: the error often associated with your choice of model

- E.g. choosing a linear model to fit quadratic data will suffer from high bias
- No matter how much data you sample from the quadratic curve, your model's predictions will still be generally bad, and your error will be high

Variance: the error from your model being too sensitive to training data

- In practice, large variance is often a result of overfitting, which can be remedied with effective regularization
- (Can also be a symptom of underfitting, although models that underfit are usually a victim of high-bias)

Bias and Variance (extra)

Note: you do not need to know the bias-variance tradeoff / decomposition for the quiz, this is just meant to supplement your understanding of the discussion in class.

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}$$

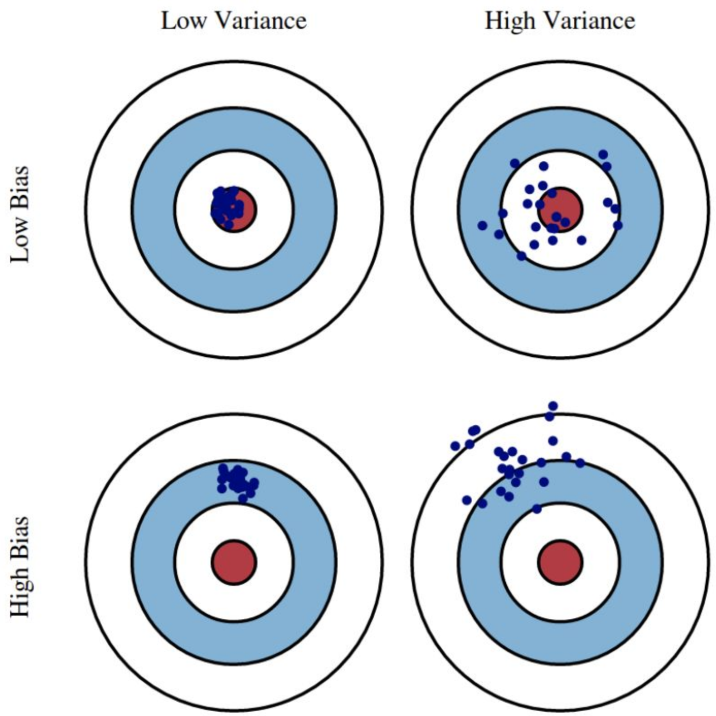
Variance: Captures how much your classifier changes if you train on a different training set. How "over-specialized" is your classifier to a particular training set (overfitting)? If we have the best possible model for our training data, how far off are we from the average classifier?

Bias: What is the inherent error that you obtain from your classifier even with infinite training data? This is due to your classifier being "biased" to a particular kind of solution (e.g. linear classifier). In other words, bias is inherent to your model.

Noise: How big is the data-intrinsic noise? This error measures ambiguity due to your data distribution and feature representation. You can never beat this, it is an aspect of the data.

[1] Content from Prof. Kilian Weinberger’s course notes for CS 4780: https://www.cs.cornell.edu/courses/cs4780/2023sp/lectures/lecturenote12.html

Bias and Variance (cont.)



[1] Content from Prof. Kilian Weinberger's course notes for CS 4780: https://www.cs.cornell.edu/courses/cs4780/2023sp/lectures/lecturenote12.html

Bias and Variance (cont.)

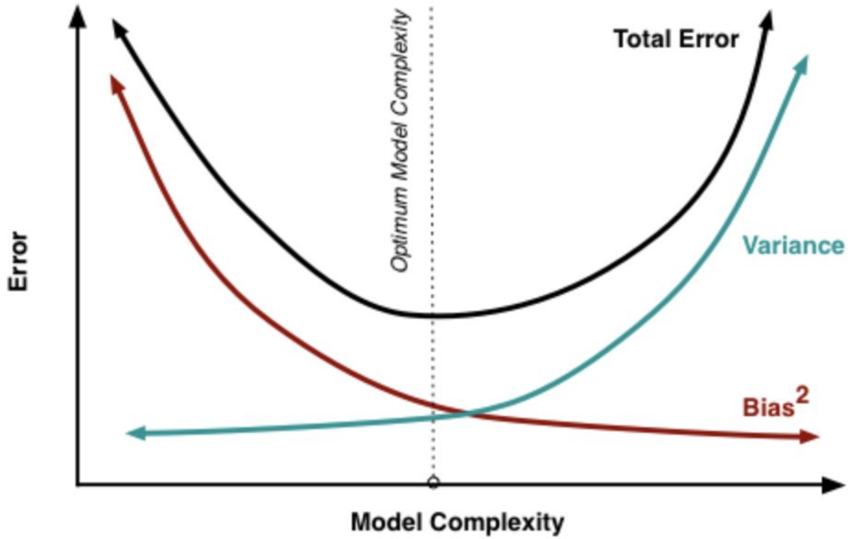
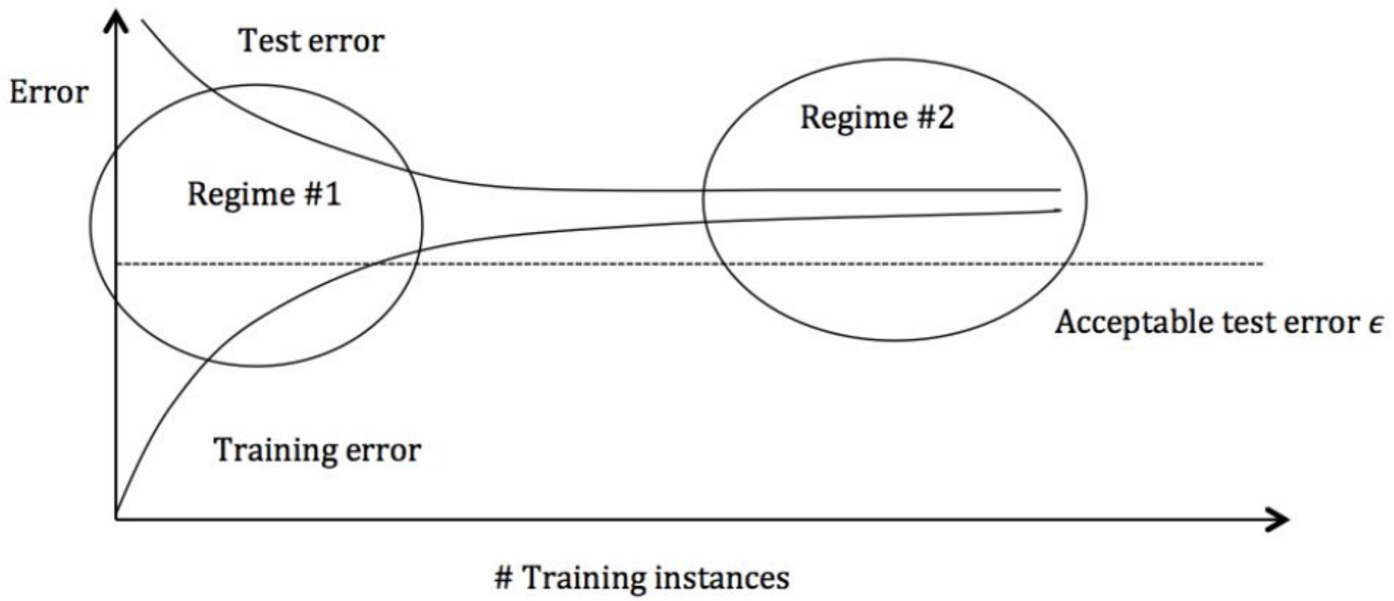


Fig 2: The variation of Bias and Variance with the model complexity. This is similar to the concept of overfitting and underfitting. More complex models overfit while the simplest models underfit.
Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

[1] Content from Prof. Kilian Weinberger's course notes for CS 4780: https://www.cs.cornell.edu/courses/cs4780/2023sp/lectures/lecturenote12.html

Bias and Variance – Knowledge Check



- Does Regime #1 suffer from high bias or high variance?
- Does Regime #2 suffer from high bias or high variance?

[1] Content from Prof. Kilian Weinberger's course notes for CS 4780: https://www.cs.cornell.edu/courses/cs4780/2023sp/lectures/lecturenote12.html

Bias and Variance – Self-Test Ideas

- Review slides 38-46 from Lecture 9 and try to answer the questions yourself
- You can check your work by watching the lecture recording

Metrics

- **Precision:** tp/(tp + fp)
  - What pct. of the time your model is correct when it makes a prediction
- **Recall:** tp/(tp + fn)
  - What pct. of “true” examples your model makes a (correct) prediction for
- **F1:** the harmonic mean of precision and recall
  - A metric which averages
- **Kendall’s Tau:** a measurement of how accurate a sort order is (see slides)
- **ROC / Precision-Recall Curves:** measures model performance across a range of thresholds
  - Area under the curve (AUC) for both plots can measure which model performs better across all thresholds
  - Typically you care more about Precision @ Fixed Recall or Recall @ Fixed precision and evaluate multiple models accordingly

Metrics – Knowledge Check

- **Q:** on which category does the model have the lowest recall?

- **Q:** on which category does the model have the highest precision?

- **Q:** what is the model's F1 score for predicting category B?

Groundtruth Label	**Model Prediction**
A	A
A	B
B	C
B	A
B	B
C	C
C	A
C	B
C	C
C	A

Metrics – Self-Test Ideas

- Create **small** fake datasets and test your ability to compute these metrics (or reason about their ROC / precision-recall curves) correctly
- You can use Python libraries like sklearn, scipy, etc. to check your work

Embeddings

Key topics include:

- Word embeddings
- RAG
- Approx. Nearest Neighbour Search / Vector DBs

Word Embeddings

Goal: to learn a transformation W which can map any word in a vocabulary to a high-dimensional representation (i.e. W(word) → word embedding)

- **CBOW:** compute sum of vector embeddings of n words before and after target and try to predict the missing word
  - n is the “window size”
- **SkipGram:** inverse of the idea in CBOW – use representation of word to try to predict its context
  - Also uses a window size to create input/output training pairs
- Both techniques learn (one or more) large parameter matrices which can be used to perform embedding lookups
- **Issues:** large param space even for 10k words; poor perf. on rare words
- **Improvements:** word pairs / phrases, subsampling, selective updates

Retrieval Augmented Generation (RAG)

Problem: LLMs have limited context(s), but their generation performance can be substantially improved with higher quality context

- Higher quality == more relevant, more up-to-date, includes information not seen during training

Idea: Let’s improve (i.e. **augment**) the context we provide LLMs by retrieving high quality context from external data source(s)

- Allows **system** to leverage dataset larger than context length
- Can avoid re-training LLM by retrieving new information
- Extends LLM “memory”
- And more…

Approximate Nearest Neighbour Search / Vector DBs

- **Problem:** performing naive linear search for nearest neighbor embedding in a vector store is computationally expensive (O(nd))
- **Idea:** trading accuracy for speed with ANNS
- We covered
    - Vector Compression - reduce dimensionality of vectors
    - Locality Sensitive Hashing - Maximize hash collision of similar objects!
    - NSW - Navigable graph with greedy search
- Make sure you understand the tradeoffs implied by the parameters of these algorithms:
    - LSH
        - L and k controls the probability of finding a “close” point
    - NSW
        - w: number of greedy searches per query
        - f: number of neighbors per vertex

Embeddings – Self-Test Ideas

- Given some **small** vocabulary and **small** weight matrices W, W’, compute the forward pass of CBOW
- Walk yourself through the NSW for some **small** graph