CS/ENGR M148 L18:
Final Review

Sandra Batista

Administrative News

Quiz 4 on PS4 today!

Only 15 minutes, T/F, multiple choice

Please bring laptop and hard copy of notes.

Final next Tuesday, 12/10/24, 3-6 pm PT!

180 minutes. Cumulative with more emphasis on material since midterm.

Please hard copy of notes. We will scan exams.

Final exam will be here, WGYoung Hall CS 50

Administrative News

This week in discussion section:

Neural Network Project Check-in. Applying NN to project data for classification or prediction. Reporting metrics. This can be what you include in final project report.

There will also be review for final exam.

Final project report guidelines posted, due 12/13/24.

Final Exam Practice Questions posted.

Join our slido for the week…

https://app.sli.do/event/nCV57u4mC7eUMit9euSBr2

What is unsupervised learning?

- Organize or describe data when no labels are available.
- A major area: clustering
- Organizing data into collections of points that are "close" is known as **clustering**.
- PCA is another example of unsupervised learning

[Shah 2020]

Dimensionality reduction

High-dimensional data is data with many features (such as thousands of variables) e.g. gene expression data, nutrition data

Dimensionality Reduction is the process of creating a lower-dimensional representation of a dataset.

1. Summarize the strongest patterns and relationships between the *variables* in the data (
2. Make computation on the data easier by reducing its size.

E.g.: Principal component analysis summarizes low-dimensional linear relationships in high-dimensional datasets

rter 2024]

Principal Component Analysis (PCA)

Principal component analysis is an algorithm that computes a series of "orthogonal" linear combinations that have the maximum possible variance relative to the origin

Variance measures of how well the variable's measurements can distinguish between the observations.

[Yu, Barter 2024]

Preprocessing for PCA

Standardization involves both **mean-centering** (subtracting the mean from each column) and **SD-scaling** (dividing each column by its standard deviation (SD)).

(SD-scaling is sometimes called normalizing.)

Common practice but not absolutely necessary.
Consider if

1. Means or 0 more important
2. If common scale needed or will it remove meaning of scale in data

[Yu, Barter 2024]

Singular value decomposition

$$X = UDV^T.$$

The left matrix, U, contains the ***left-singular vectors***

The matrix D is a **diagonal matrix** whose entries correspond to the magnitude or strength of the corresponding principal component directions. **Singular values are diagonal entries of D.**

V contains the **right-singular vectors** of the data matrix and each right-singular vector corresponds to a principal component.

Each column of V contains the coefficients of the corresponding principal component linear combination.

Variable Loadings

$$V = \begin{array}{r} \\ \text{(Sodium)} \\ \text{(Potassium)} \\ \text{(Calcium)} \\ \text{(Phosphorus)} \\ \text{(Magnesium)} \\ \text{(Total choline)} \end{array} \begin{array}{c} \begin{array}{cccccc} \text{(PC1)} & \text{(PC2)} & \text{(PC3)} & \text{(PC4)} & \text{(PC5)} & \text{(PC6)} \end{array} \\ \begin{bmatrix} 0.42 & 0.63 & 0.23 & -0.58 & 0.15 & -0.1 \\ 0.48 & -0.28 & -0.43 & -0.3 & -0.64 & 0.1 \\ 0.29 & -0.24 & 0.79 & 0.22 & -0.35 & -0.27 \\ 0.49 & -0.06 & 0.12 & 0.28 & 0.3 & 0.76 \\ 0.38 & -0.51 & -0.16 & -0.11 & 0.59 & -0.46 \\ 0.35 & 0.45 & -0.33 & 0.66 & -0.08 & -0.35 \end{bmatrix} \end{array}.$$

$$\begin{aligned} \text{PC1} = {} & 0.42\,\text{sodium} + 0.48\,\text{potassium} + 0.29\,\text{calcium} \\ & + 0.49\,\text{phosphorus} + 0.38\,\text{magnesium} \\ & + 0.35\,\text{total choline}. \end{aligned}$$

How to get PC transformed data?

[Yu, Barter 2024]

Variance Explained

The sum of the variance of the columns in the original dataset equals the sum of the squared singular values:

$$\sum_j Var(X_j) = \sum_j d_j^2,$$

$$\text{prop of variability explained by PC}_j = \frac{d_j^2}{\sum_i d_i^2}.$$

[Yu, Barter 2024]

Divisive clustering

1. Put all objects in one cluster
2. Repeat until all clusters are singletons {

- choose a cluster to split based on some criterion.
- replace the chosen cluster with sub-clusters.

}

[Shah 2020]

Agglomerative clustering

1. Use any computable cluster similarity measure $sim(C_i, C_j)$ e.g., Euclidean distance,
2. For n objects $v_1, \ldots, v_n$, assign each to a singleton cluster $C_i = \{v_i\}$
3. Repeat {

   - identify two most similar clusters C_j and C_k (could be ties-chose one pair)

   - delete C_j and C_k and add $(C_j \cup C_k)$ to the set of clusters.

   } until just one cluster.
4. Dendrograms diagram the sequence of cluster merges.

[Shah 2020]

K-means algorithm

1. Begin with a decision on the value of K = number of clusters.
2. Put any initial partition that classifies the data into K clusters. You may assign the training samples randomly or systematically.
3. Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.

Repeat the above three steps until convergence is achieved.

[Shah 2020]

Within Sum of Squares (WSS)

Within-Cluster Sum of Squares:

$$WSS = \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{j=1}^{p} (x_{i,j} - c_{k,j})^2$$

- **WSS** quantifies cluster tightness
- WSS decreases with number of clusters
- WSS increases with number of data points

[Yu, Barter 2024]

Silhouette score

For a single data point:

Let a_i be average distance from point i to other points in same cluster

Let b_i be average distance from point i to points in nearest cluster

Silhouette score:

$$\text{silhouette score}_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- **Silhouette** quantifies how tightly each data point is clustered and how distinctly each data point is clustered
- Average silhouette scores for each point to get clustering silhouette score

[Yu, Barter 2024]

Rand Index

How to compare different sets of clusters?

For each pair of data points, consider if the clusters put the Same pair of points in the same cluster or different clusters.

Rand Index = # pairs in agreement/ total # of pairs

[Yu, Barter 2024]

Sample code for PS3

You can review PS3 by reviewing the following lecture notebooks:

L11 on PCA:

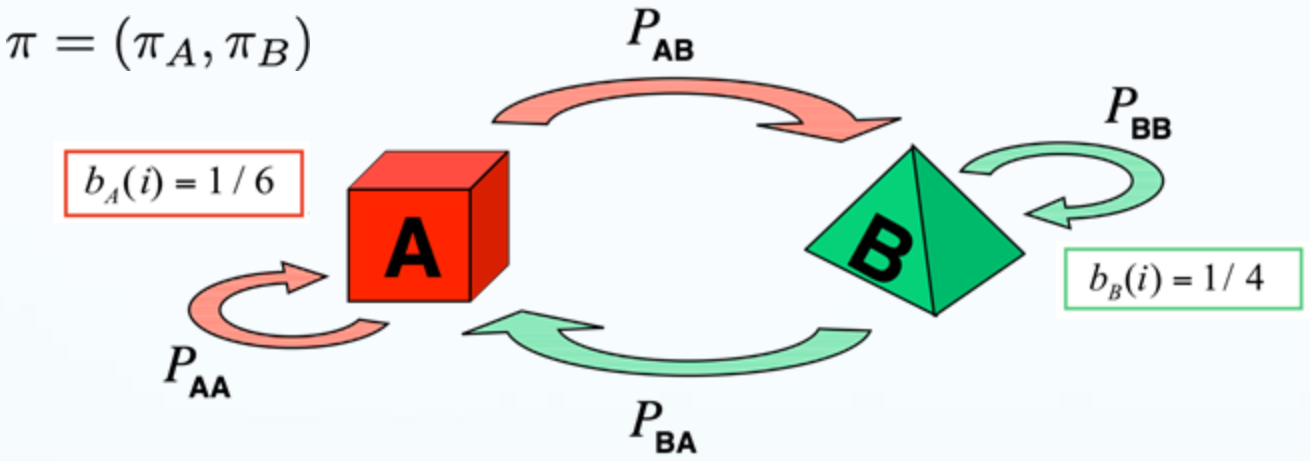
https://colab.research.google.com/drive/1UwcuL31OrlRVKFIRPyDcAM1tdG73j8ko?usp=sharing

L12 on Clustering:

https://colab.research.google.com/drive/1LFqbyV7mr6jZup16zVfB1Fgh9TwxU-7f?usp=sharing

[Yu, Barter 2024]

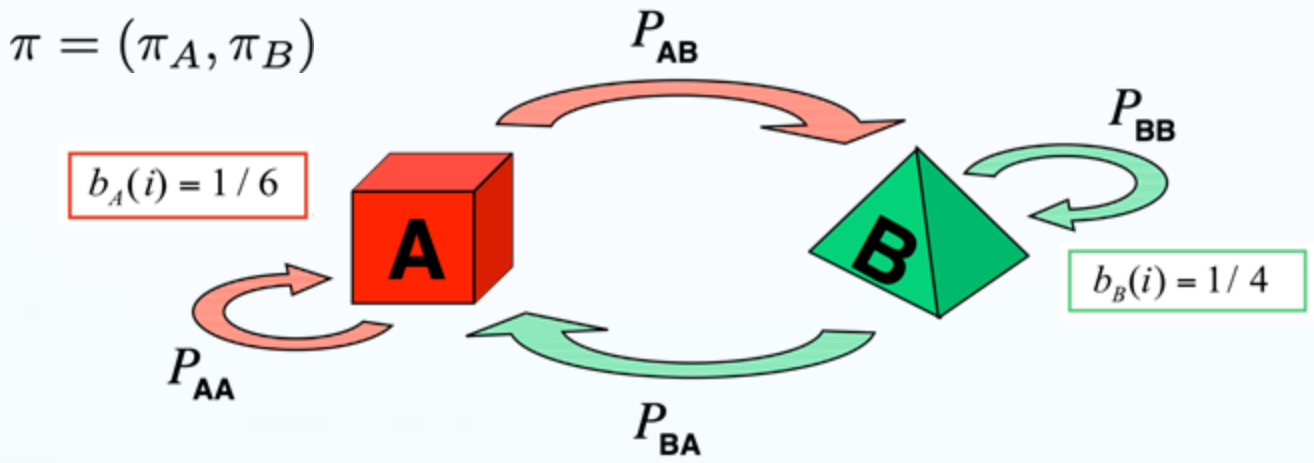
A hidden Markov model



$$\Pr(Y_0 = 1, Y_1 = 4, Y_2 = 3, Y_3 = 6, Y_4 = 6, Y_5 = 4) = ?$$

Solved with the forward algorithm.

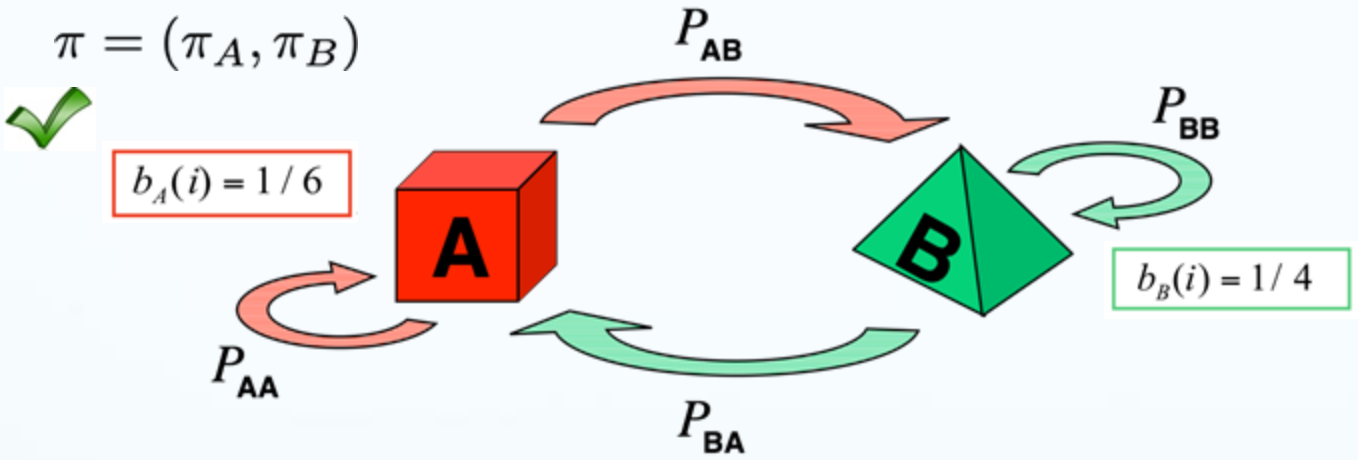
A hidden Markov model



What is the most likely sequence of states of the Markov chain to have resulted in 1,4,3,6,6,4?

Solved with the Viterbi algorithm.

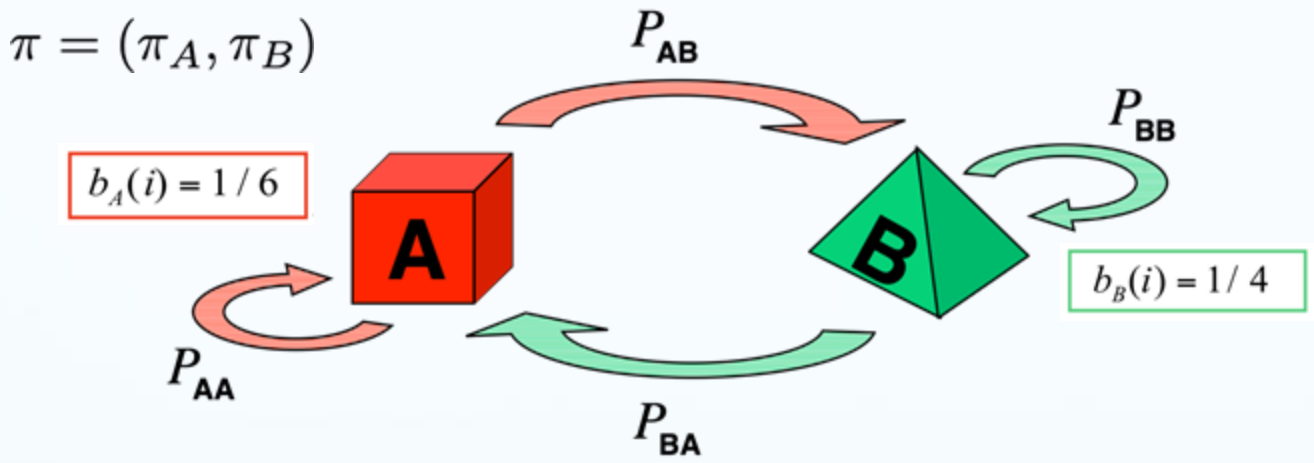
A hidden Markov model



What is the probability that X_3 = B if Y_0=1,Y_1=4,Y_2=3,Y_3=6,Y_4=6,Y_5=4 ?

Solved with the forward-backward algorithm.

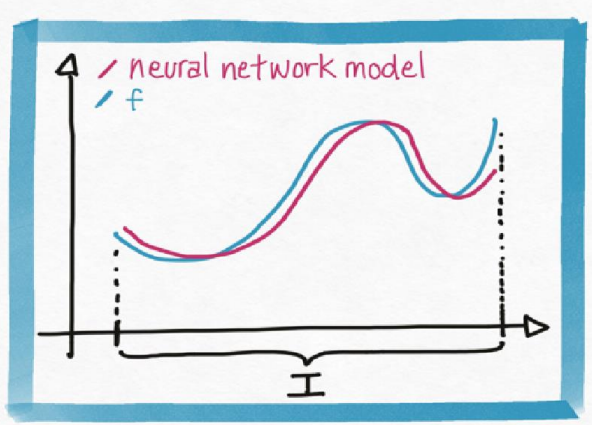
A hidden Markov model



*Given multiple sequences of numbers (observations of **Y**), estimate parameters for the model*

This is expectation-maximization algorithm

Neural Networks as Universal Approximators



Theorem:

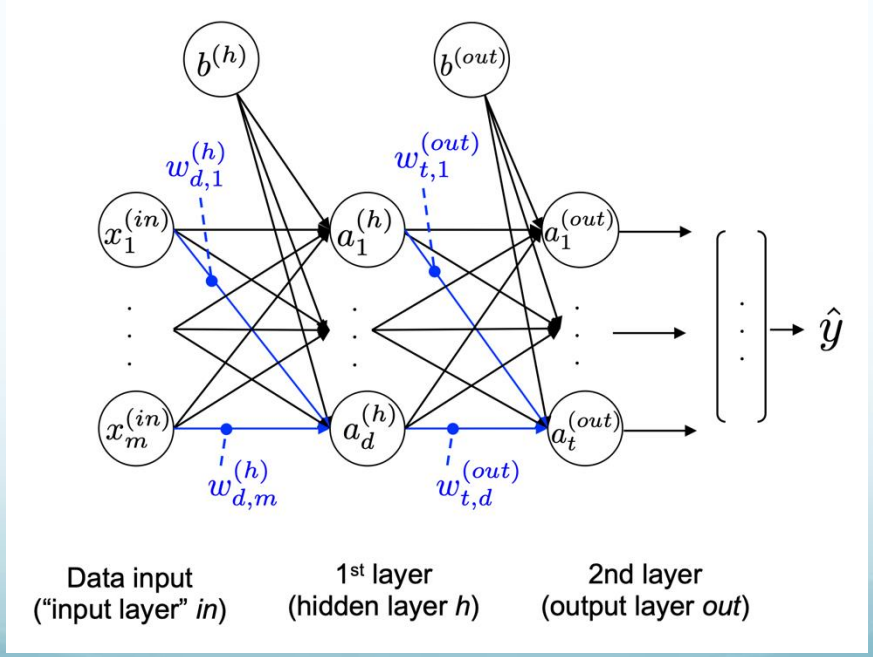
For any continuous function f defined on a bounded domain, we can find a neural network that approximates f with an arbitrary degree of accuracy.

One hidden layer is enough to represent an approximation of any function to an arbitrary degree of accuracy.

A neural network can approximate non-linear functions either for regression or classification.

Multilayer Perceptron

- *A neural network is a combination of neurons such as logistic regression (or other types) units.*
- *A **multilayer perceptron (MLP)** is a fully connected network of neurons.*
- *An MLP is a multilayer **feedforward** NN because each layer is input to the next.*
- *A network with more than one hidden layer is a **deep NN.***



[Raschka et al 2022]

Activation function

$$h = f(W^{\top} X + b)$$

The activation function should:

- Provide non-linearity
- Ensure gradients remain large through hidden unit

Examples:

- sigmoid
- ReLU
- identity

Forward Algorithm

Require: Network depth, l
Require: $\boldsymbol{W}^{(i)}, i \in \{1, \ldots, l\}$, the weight matrices of the model
Require: $\boldsymbol{b}^{(i)}, i \in \{1, \ldots, l\}$, the bias parameters of the model
Require: $\boldsymbol{x}$, the input to process
Require: $\boldsymbol{y}$, the target output

$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
for $k = 1, \ldots, l$ **do**
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
end for
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda \Omega(\theta)$

[Goodfellow et al 2016]

Loss Functions

As with other models, we have choice of loss functions, such as

1. *MSE*
2. *Negative Log Likelihood (Binary Cross Entropy) or its generalization for multi-class classification*

We'll look at MSE for MNIT

Backprop Algorithm

1. *Compute the gradient of loss function with respect to output layer*
2. *For each layer*
   - *i. Convert gradient on layer's output into a gradient with respect to the net input before activation*
   - *ii. Compute gradients on weights and biases by using gradient of the net input with respect to weights*
   - *iii. Propagate the gradients backwards in NN (i.e. save to reuse for gradients in lower layers)*

Gradients give us how much output layer or weight should change To reduce loss.

[Goodfellow et al 2016]

Gradient Descent

1. *Initialize small weights*
2. *For each sample*
   - *i. Apply the forward algorithm*
   - *ii. Apply backprop*
   - *iii. Use the gradients of the weights and biases to update the weights and biases*

*Repeat step 2 for a number of **epochs,** complete pass through Training data*

[Goodfellow et al 2016]

Updating the weights

For each weight

$$\Delta w = \frac{\partial L}{\partial w}$$

$$w^{new} = w^{old} - \eta \Delta w$$

What is regularization?

Regularization *is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.*

Regularization of NN

1. Norm penalties
2. Early Stopping
3. Data Augmentation
4. Ensembles such as Dropout

Norm Penalties

We used to optimize:

$$W^{(i+1)} = W^{(i)} - \lambda \frac{\partial L}{\partial W}$$

Change to:

$$L_R(W; X, y) = L(W; X, y) + \frac{1}{2}\alpha \parallel W \parallel_2^2$$

$$W^{(i+1)} = W^{(i)} - \lambda \frac{\partial L}{\partial W} - \lambda\alpha\, W^{(i)}$$

Weights decay in proportion to size

Biases not penalized

L_2 regularization:

- Weights decay

$$\Omega(W) = \frac{1}{2} \parallel W \parallel_2^2$$

L_1 regularization:

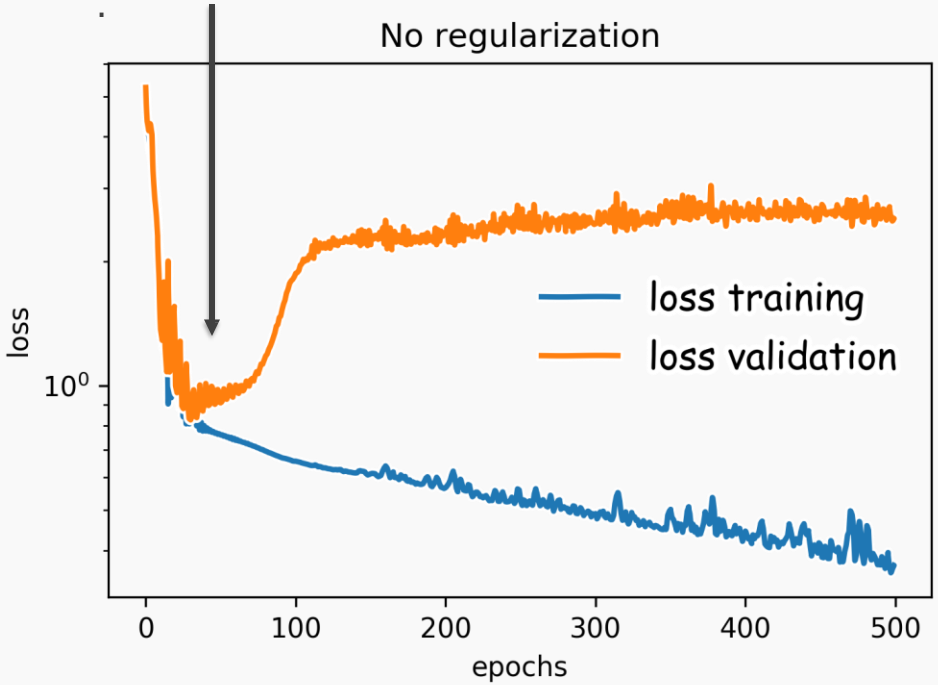
- encourages sparsity

$$\Omega(W) = \frac{1}{2} \parallel W \parallel_1$$

[CS M148 Winter 2024]

Early Stopping

Early stopping: terminate while validation set performance is better. Stop training when validation error reaches a minimum
Hinton calls this a “beautiful free lunch”



Patience is defined as the number of epochs to wait before early stop if no progress on the validation set.

The patience is often set somewhere between 10 and 100 (10 or 20 is more common), but it really depends on the dataset and network.

[CS M148 Winter 2024]

Aurelien Geron. 2019. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd. ed.). O'Reilly Media, Inc

Data Augmentation

Data Augmentation is adding fake data to your training data set.

What are some means for data augmentation?

1. *Transforming data*
2. *Adding noise to data*
3. *Bootstrap*
4. *Gaussian Mixture Models (Here use as a generative model)*

Random Forests

Random Forest (RF) algorithm is an **ensemble** algorithm that combines many decision trees:

- Training each tree using a different random bootstrap sample of the training data.

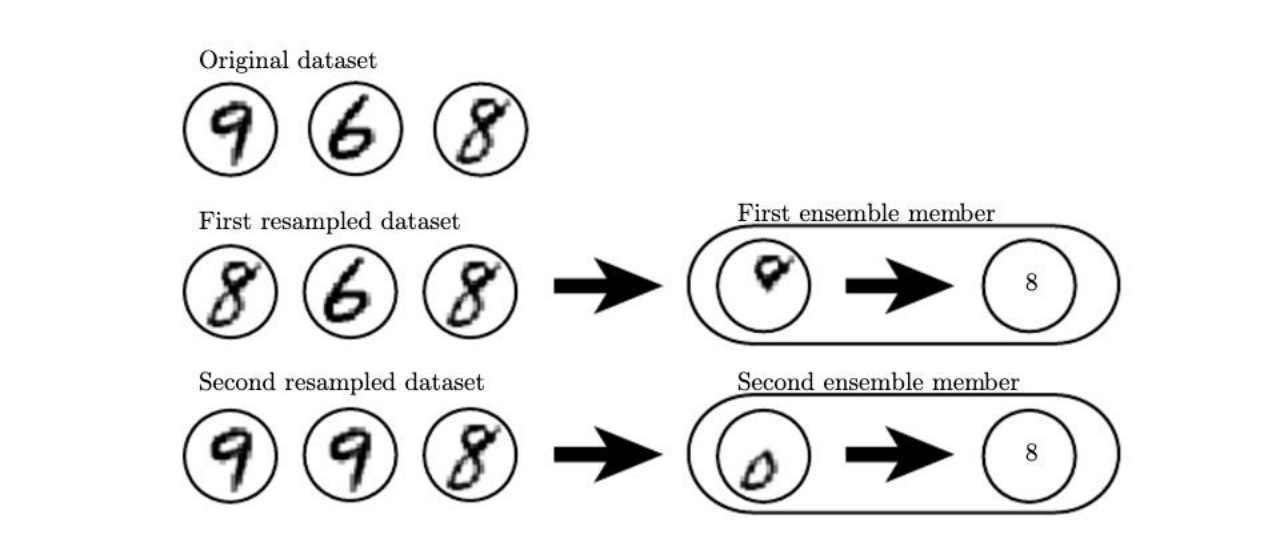
- Considering a different random subset of the predictor variables for each node split

[Yu, Barter 2024]

Bootstrap Aggregating

Bootstrap aggregating or **bagging**: Train several different models separately on different bootstrapped samples and then have average the results

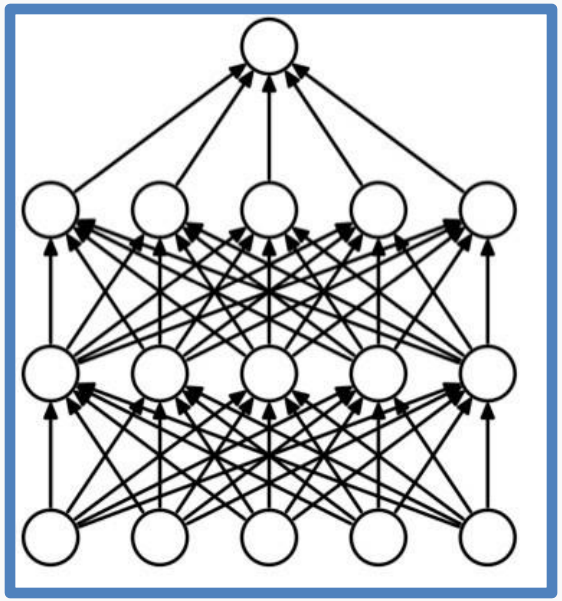
This is called **model averaging.**



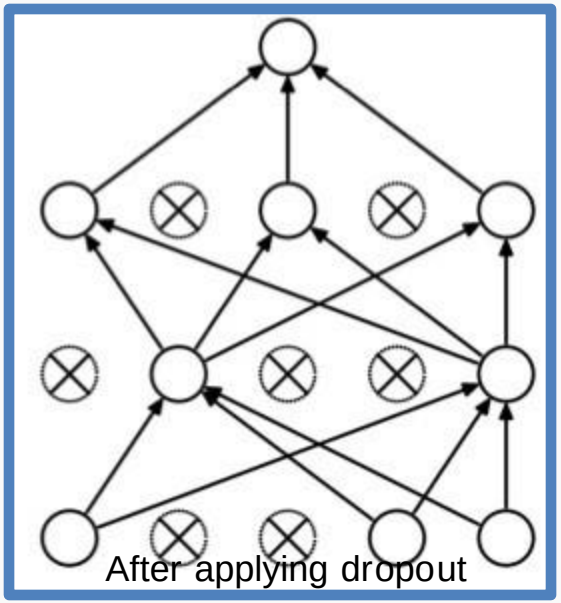
[Goodfellow et al 2016, Figure 7.5]

Dropout

- Proposed by Hinton (2012) and Srivastava et al (2014)
- Randomly set some neurons and their connections to zero (i.e. “dropped”)
- Prevent overfitting by reducing co-adaptation of neurons
- Like training many random sub-networks



Standard Neural Network

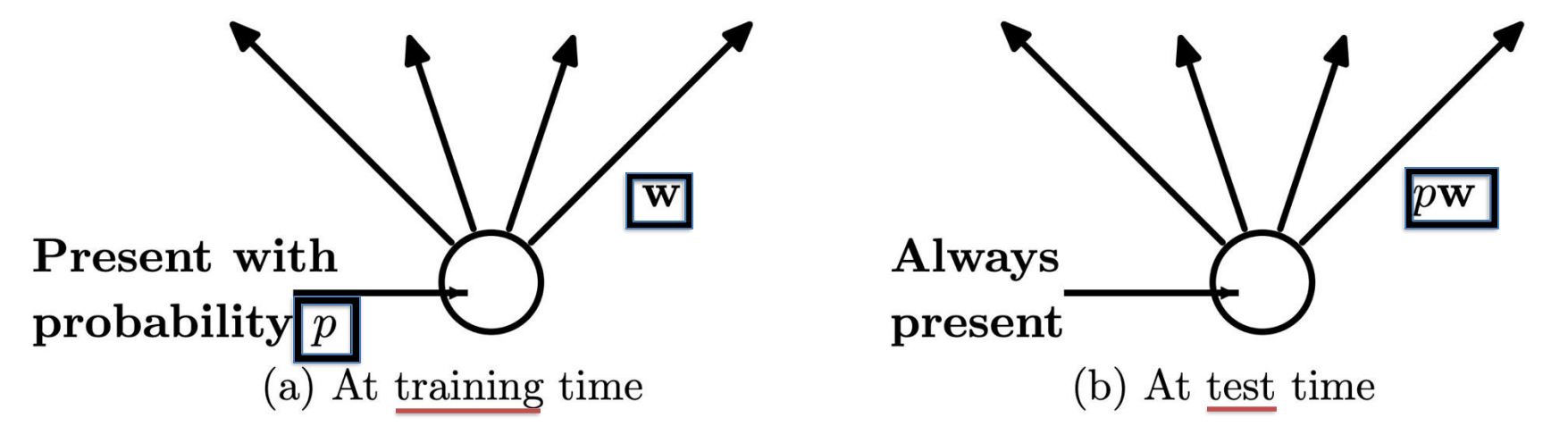


After applying dropout

https://arxiv.org/pdf/1207.0580.pdf

Dropout: Prediction

- We can think of dropout as training many of sub-networks
- At **test time**, we can "aggregate" over these sub-networks by **reducing connection weights in proportion to dropout probability, *p***



NOTE: Dropouts can be used for **neural network inference** by dropping during predictions and predicting multiple times to get a distribution

Dropout vs Bagging

- Both are training ensembles of models
- In bagging each model is independent
- In bagging each model is trained on entire training set
- In dropout, the models share parameters!!
- In dropout, some set of models are sampled and trained on a single sample.
- In dropout, the parameter sharing across sampled models helps set parameters and make problem tractable.

- Otherwise both use bootstrapping and averaging of models.

[Goodfellow et al 2016]

Can we Trust Our Models?

When we train a model, we implicitly trust that the model make sensible predictions. Such assessment is usually done by looking at held-out accuracy or some other aggregate measure.

However, such metrics can be very misleading.

- *There can be data leakage*
- *The results may not generalize*
- *The results may not tell us what features are important to consider for further study*

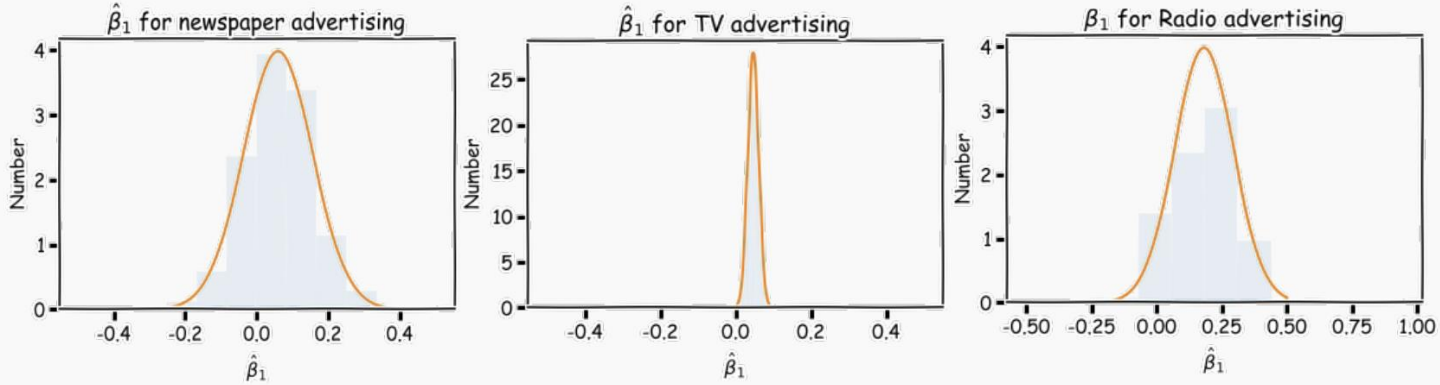
Understanding the model's predictions can be an additional useful tool when deciding if a model is trustworthy or not.

[CS M148 Winter 2024]

Feature importance

Now we know how to generate these distributions we are ready to answer ***two important questions:***

A. Which predictors are most important?
B. And which of them really affect the outcome?



$\mu_{\beta_1} = 0.03$
σ_{β_1}=0.13

$\mu_{\beta_1} = 0.033$
σ_{β_1}=0.01

$\mu_{\beta_1} = 0.23$
σ_{β_1}=0.25

0

Comparing coefficients

The coefficients of different predictive features (predictor variables) are not comparable unless

- They're on the same scale

- Coefficients have been standardized to create **t-values:**

$$t_j = \frac{b_j}{SD(b_j)}.$$

[Yu, Barter 2024]

Hypothesis testing

1. **State Hypothesis:**

Null hypothesis: There is no relation between *X* and *Y*

The alternative: There is some relation between *X* and *Y*

2. **Choose test statistics**

t-test

3. **Do permutation testing**

4. **Reject or not reject the hypothesis:**

We compute ***p-value ,*** the probability of observing any value equal to $|t|$ or larger, from random data.

p-value < p-value-threshold we reject the null.

Shapley values

The Banks simplified model uses only 3 features: the income, the location and the age.

How can we understand how much each feature value contributed to the final prediction?

Shapley values *are from game theory:*
In a group of players with different skill sets that worked together to reach a payout, what is the fairest way to split the payout among them?

In machine learning: the game is the prediction task, the "payout" is model output and the "players" are the features.

Review Exercise

Thank you to **Naomi Nguyen**

1. Compare the advantages and disadvantages of batch gradient descent, stochastic gradient descent, and mini-batch gradient descent in this context.

2. Explain the importance of the learning rate in gradient descent, and analyze the consequences of choosing a learning rate that is too large or too small.

Review Exercise

Thank you to **Bryan Yang**.

HMM with hidden states H (indicating coding DNA) and L (indicating non-coding DNA)

Observe strings of bases: A, C, G, T

Initial state is equally likely to be H or L.

Transition probabilities H->L = 50% L->H = 40

Probabilities of four DNA bases in each state:
State H: P(A) = 0.2 P(C) = 0.3 P(G) = 0.3 P(T) = 0.2
State L: P(A) = 0.3 P(C) = 0.2 P(G) = 0.2 P(T) = 0.3

Draw this HMM

Review Exercise

Thank you to **Bryan Yang**.

HMM with hidden states H (indicating coding DNA) and L (indicating non-coding DNA)

Observe strings of bases: A, C, G, T

Initial state is equally likely to be H or L.

Transition probabilities H->L = 50% L->H = 40

Probabilities of four DNA bases in each state:
State H: P(A) = 0.2 P(C) = 0.3 P(G) = 0.3 P(T) = 0.2
State L: P(A) = 0.3 P(C) = 0.2 P(G) = 0.2 P(T) = 0.3

Review Exercise

Thank you to **Bryan Yang**.

Now consider the sequence S = GGCAC, which was generated via a path through this HMM. What might have been the most probable path to obtain S?

Viterbi: Example

π

	ε	G	G^{x}	C	A	C
B	1	0	0	0	0	0
H	0					
L	0					

$$v_k(i) = e_k(x_i)\max_r(v_r(i-1)a_{rk})$$

Review Exercise

Thank you to **Bryan Yang**.

Now consider the sequence S = GGCAC, which was generated via a path through this HMM.

For further practice:

How would you apply the forward algorithm to figure out the probability of the sequence?

How would you apply the forward-backward algorithm to figure out that the first C occurred while in the state L?

The Forward Algorithm

- *Initialization ($i = 0$)*

$$f_0(0) = 1, \quad f_k(0) = 0 \text{ for } k > 0$$

- *Recursion ($i = 1, \ldots, L$): For each state k*

$$f_k(i) = e_k(x_i) \sum_r f_r(i-1) a_{rk}$$

- *Termination:*

$$Pr(x) = \sum_k f_k(L) a_{k0}$$

The Backward Algorithm

- *Initialization (i = L)*

$$b_k(L) = a_{k0} \text{ for all } k$$

- *Recursion (i = L-1, . . . , 1): For each state k*

$$b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$$

Backward Algorithm and Posterior Decoding

- *How likely is it that my observation comes from a certain state?*

  $P(x_i \text{ is emitted by state } k \mid \text{whole observation})$

- *Like the Forward matrix, one can compute a Backward matrix*
- *Multiply Forward and Backward entries*

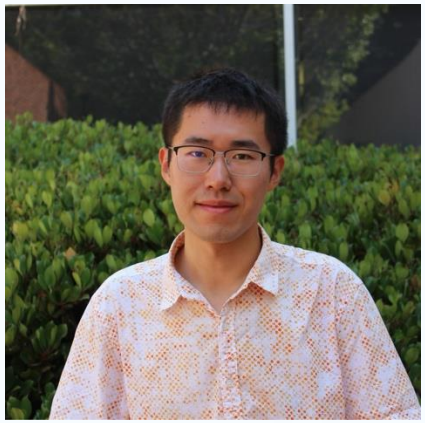
$$P(\pi_i = k \mid x) = \frac{f_k(i) \cdot b_k(i)}{P(x)}$$

  - *P(x) is the total probability computed by, e.g., forward algorithm*

Thank you!

We would not have been able to have this course this quarter without our amazing course staff:

TAs



Fang

Lucas

Yihe

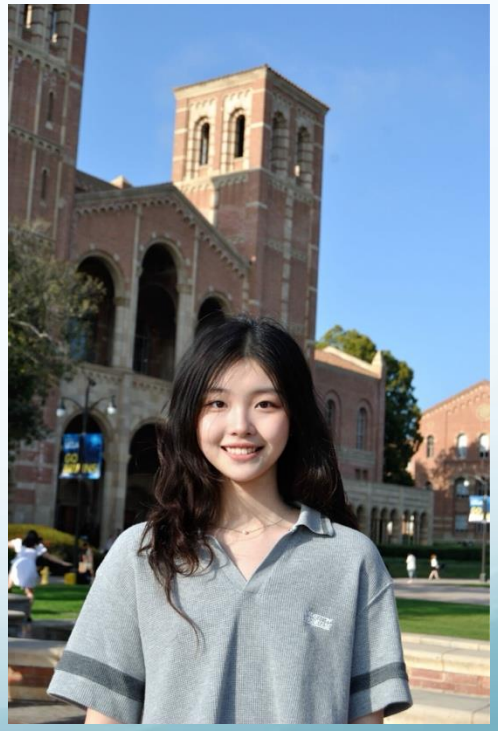
LAs

Andy

Clyde

Princeton

Sylvia
Tiffany
Victoria



Special thanks to our anonymous graders also!!

Thank You

Thank you for all your hard work all quarter!