

# CS 188 Discussion 11:

## Naive Bayes, Perceptrons

Kenny Wang ([kwkw@berkeley.edu](mailto:kwkw@berkeley.edu))

Wed Nov 8, 2023

Slides based on Sashrika + Joy

# Administrivia

- Project 4 deadline extended Mon, Nov 6 => Fri, Nov 10
- Homework is due on Tuesdays
- We have office hours pretty much all day every weekday (12-7), come to Soda 341B! (my hours are 1-3 PM on Mondays)
- Discussion slides are on Ed

# Today's Topics

- Machine Learning!
- Naive Bayes
- Perceptrons

# Musheen Lurning!!1

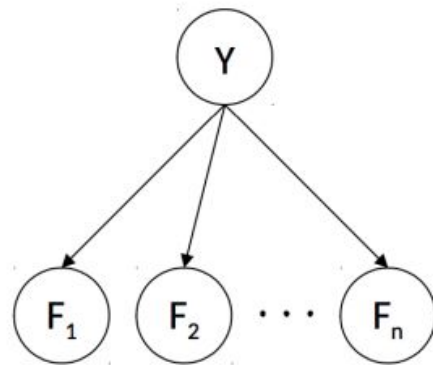
- **Core Idea:** We give machines access to data and they learn for themselves!
- Data is often split into training, validation, and test sets
  - **Training set:** Used to fit the model
  - **Validation set:** Used to tune **hyperparameters** (learning rate, model structure, etc)
  - **Test set:** Used to test the entire model
- Some types of machine learning problems
  - **Classification problems:** try to classify data into discrete classes
  - **Regression problems:** try to estimate some numerical value from data
  - **Clustering problems:** try to group similar data into clusters
- Types of learning
  - **Supervised learning:** training data has labels, e.g. classification
  - **Unsupervised learning:** training data has no labels, e.g. clustering

# Naive Bayes

- **Goal:** create a model that can predict a label  $Y$  given features, where we assume all features are independently affected by the label
- Ex: Spam filter
  - $Y$  is in {Spam, Ham}
  - $F_i$  in {0, 1} is whether word  $i$  appears in the email.
- Label email based on the higher of these two probabilities:

$$P(Y = \text{ham} | F_1 = f_1, \dots, F_n = f_n) \quad P(Y = \text{spam} | F_1 = f_1, \dots, F_n = f_n)$$

- Generalized: 
$$\begin{aligned} \text{prediction}(f_1, \dots, f_n) &= \underset{y}{\operatorname{argmax}} P(Y = y | F_1 = f_1, \dots, F_n = f_n) \\ &= \underset{y}{\operatorname{argmax}} P(Y = y, F_1 = f_1, \dots, F_n = f_n) \\ &= \underset{y}{\operatorname{argmax}} P(Y = y) \prod_{i=1}^n P(F_i = f_i | Y = y) \end{aligned}$$

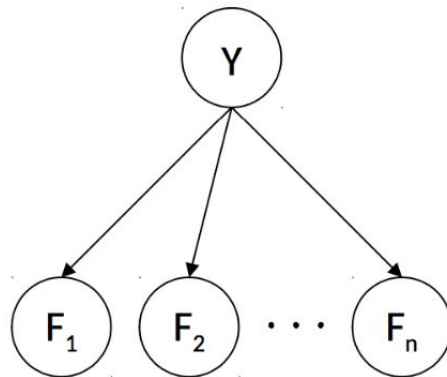


# Maximum Likelihood Estimation

- How to estimate CPTs, since we don't actually know them?
  - Parameter estimation with MLE
- Find the probabilities (CPT values)  $\theta = \mathbf{P}(\cdot)$  such that we maximize the likelihood of observing our observations,  $P(\text{observations} \mid \theta)$
- Answer is actually fairly intuitive. f/e, given data  $(F, Y)$ ,

$$\begin{aligned} P(Y=y) &= \text{MLE}(\theta \mid (F, Y)) \\ &= \frac{\text{\# examples with } Y=y}{\text{total \# examples}} \end{aligned}$$

$$\begin{aligned} P(F_i=f \mid Y=y) &= \text{MLE}(\theta \mid (F, Y)) \\ &= \frac{\text{\# examples with } (F_i=f, Y=y)}{\text{\# examples with } Y=y} \end{aligned}$$



# Laplace Smoothing

## Worksheet Q1

- Chance of **overfitting** (model doesn't generalize well post-training) with our parameter estimation
- **Laplace Smoothing**: pretend you saw each of the  $|X|$  possible outcomes  $k$  extra times

$$P_{LAP,k}(x) = \frac{\text{count}(x) + k}{N + k|X|}$$

$$P_{LAP,k}(x|y) = \frac{\text{count}(x,y) + k}{\text{count}(y) + k|X|}$$

- $k$  is a **hyperparameter**, meaning you can choose what to set it to
  - Smaller  $k$  means your probability estimates follow the training data more closely
  - Larger  $k$  means your probability estimates are more uniform

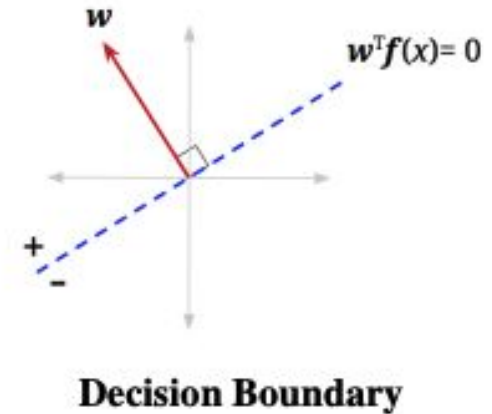
# Perceptrons

- Naive Bayes
- **Perceptrons**



# Binary Perceptrons

- Idea: Linearly separate data into two classes using a decision boundary (defined by a set of weights)
- If the data is linearly separable, the algorithm will perfectly classify the data!
- To find boundaries that don't have to cross the origin, incorporate a “bias” feature that always has value 1



# Perceptron Algorithm

## Worksheet Q2

1. Initialize all weights to 0:  $\mathbf{w} = \mathbf{0}$
2. For each training sample, with features  $\mathbf{f}(x)$  and true class label  $y^* \in \{-1, +1\}$ , do:
  - (a) Classify the sample using the current weights, let  $y$  be the class predicted by your current  $\mathbf{w}$ :

$$y = \text{classify}(x) = \begin{cases} +1 & \text{if } \text{activation}_w(x) = \mathbf{w}^T \mathbf{f}(x) > 0 \\ -1 & \text{if } \text{activation}_w(x) = \mathbf{w}^T \mathbf{f}(x) < 0 \end{cases}$$

- (b) Compare the predicted label  $y$  to the true label  $y^*$ :
    - If  $y = y^*$ , do nothing
    - Otherwise, if  $y \neq y^*$ , then update your weights:  $\mathbf{w} \leftarrow \mathbf{w} + y^* \mathbf{f}(x)$
3. If you went through **every** training sample without having to update your weights (all samples predicted correctly), then terminate. Else, repeat step 2.

**Rest of the Worksheet**

# Thank you for attending!

Attendance link:

- <https://tinyurl.com/cs188fa23>

Discussion No: 11

Remember my name is Kenny

My email: [kwkw@berkeley.edu](mailto:kwkw@berkeley.edu)

