

# Probability Review and Bayesian Models (COSC 425)



THE UNIVERSITY OF  
TENNESSEE  
KNOXVILLE



# Announcements

- Lab 4 posted today and due on October 27
- Final Projects:
  - **Deadline for team selection is October 19**
    - Email me your team with subject line: COSC 425 Team Project Members
    - One person from the team should email me by then and copy the other members
    - **5-point penalty on the final project if your team is not defined by October 19**
  - Finalization of dataset to be used by **November 3**.
    - 5-point penalty if the dataset is not selected by that date.

# Lab 4

Linear Regression and SVM

# Lab 4: Linear Regression and SVM

- In this lab, you will be practicing using linear regression, SVMs, and applying them to datasets.
  - Linear Regression dataset: Auto MPG from UCI ML Repository
  - SVM Dataset: Star Type Classification from NASA, available on Kaggle
  - Both datasets are on Canvas for download
- You will create a Jupyter notebook for each part that you will submit where you will put the code that you wrote for this lab.
- You will also submit a separate written report in PDF form in which you include the answers to the questions and/or plots required by the question.  
**Note:** For all plots, you should include axes labels and titles.



# Part 1: Linear Regression

- Read in the auto-mpg.csv dataset using pandas and create the y label vector that we will use throughout, which will be the MPG column.
- The features we will be using for regression are the columns cylinders, displacement, horsepower, weight, and acceleration.
  - You will ignore the model-year, origin, and car-name columns.
- Anywhere you are asked to create a train-test split on the data, you should use a fixed random state throughout (noted in your write-up) and a test size=0.2.

# Question 1.1

- Worth 60 points – 12 points for each feature
- For each of the five features (cylinders, displacement, horsepower, weight, and acceleration), do the following:
  - Create an X matrix where each row is the current feature of interest.
  - Do the train test split as indicated above.
  - Create a plot that shows the current feature on the x-axis and the MPG on the y-axis for the training data.
  - In the report, note whether you expect linear regression will perform well using that feature to predict the MPG value and why you believe it will perform well or not perform well.
  - Use sklearn's `LinearRegression()` to fit to the training data.
  - Create a prediction vector based on the training data, and calculate and print the mean-squared error and the  $R^2$  score using sklearn on the training set.
  - Create a prediction vector based on the testing data, and calculate and print the mean-squared error and the  $R^2$  score using sklearn on the testing set.
  - Print the coefficient and intercept parameters from the sklearn model for that feature.
  - Plot the line created from that linear regression along with points for the training data (as one color) and the testing data (as another color). Include a legend to denote which are training and which are testing. Note that the x-axis should be the feature you're predicting on and the y-axis should be MPG.
- In the report, note which of the features you believe is best for predicting MPG and why you selected that feature.

# Question 1.2

- Worth 10 points
- Combining all of the data, do the following:
  - Create an X matrix that includes all five features.
  - Do the train test split as indicated above.
  - Use sklearn's `LinearRegression()` to fit to the training data.
  - Create a prediction vector based on the training data and calculate and print the mean-squared error and the  $R^2$  score using sklearn.
  - Create a prediction vector based on the testing data and calculate and print the mean-squared error and the  $R^2$  score using sklearn.
- Does using all of the data improve performance over using each of the features individually?

# Part 2: Linear SVM

- Read in the Stars.csv dataset using pandas and create the y label vector, which will be the Type column.
- The features we will be using for prediction are the columns Temperature, L, R, and A M.
- Anywhere you are asked to create a train-test split on the data, you should use a fixed random state throughout (noted in your write-up) and a test size=0.2.
- Use a fixed random state for the LinearSVC classifier throughout.



# Question 2.1

- Worth 25 points
- Finding hyperparameters:
  - Create a validation set and subtraining set from the training set (a single validation set, rather than KFold) with 0.125 of the training set serving as the validation set.
  - Fit the LinearSVC classifier on the training set and evaluate on the validation set for all combinations of max iter=[1000, 10000, 100000, 1000000] and C=[0.01, 0.1, 1, 10, 100, 1000].
  - Print the validation accuracy for each combination and show a heatmap comparing of the validation accuracies across the combinations (include a colorbar).
  - Note the best parameter combination and discuss why those parameter values might be the best for this dataset (1-3 sentences).

## Question 2.2

- Worth 5 points.
- Use the maximum iterations and C values you found to perform best to inform how you create your `LinearSVC()` and fit on the whole training set.
- Compute and print the training and testing accuracy for the Stars dataset.

# Submission Checklist

- In this lab, you will submit on Canvas:
  - Jupyter notebook for part 1
  - Jupyter notebook for part 2
  - PDF of your report
- Deadline: October 28, 2022, 11:59 PM
- Late Penalty: 20 points off per day late

# Probability Review

# Rules of Probability

- A probability distribution  $P$  specifies the likelihood of event  $e$ , where  $P(e) \in [0,1]$
- Convention: random variables are capitalized and their instantiations are lower case:  $P(A = a, B = b, C = c)$  for random variables  $A, B, C$  and instantiations  $a, b, c$

# Some Standard Rules of Probability

- Sum-to-one:  $\sum_e P(E = e) = 1$

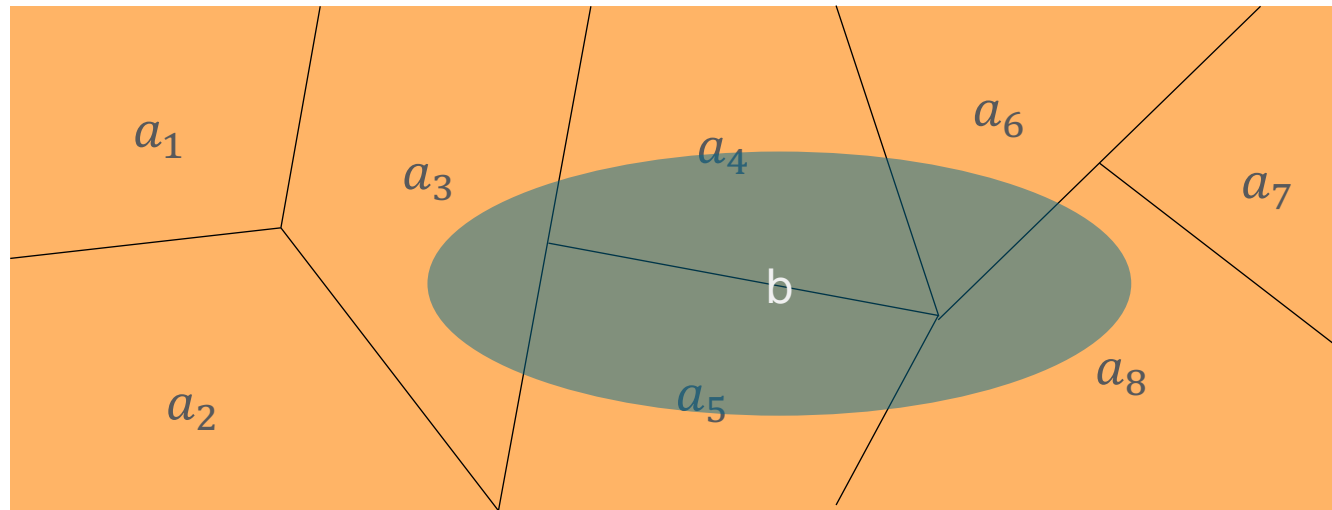


Sum across the entire box is 1



# Some Standard Rules of Probability

- Marginalization: You can sum out one random variable to remove it from the world:  $\sum_a P(A = a, B = b) = P(B = b)$



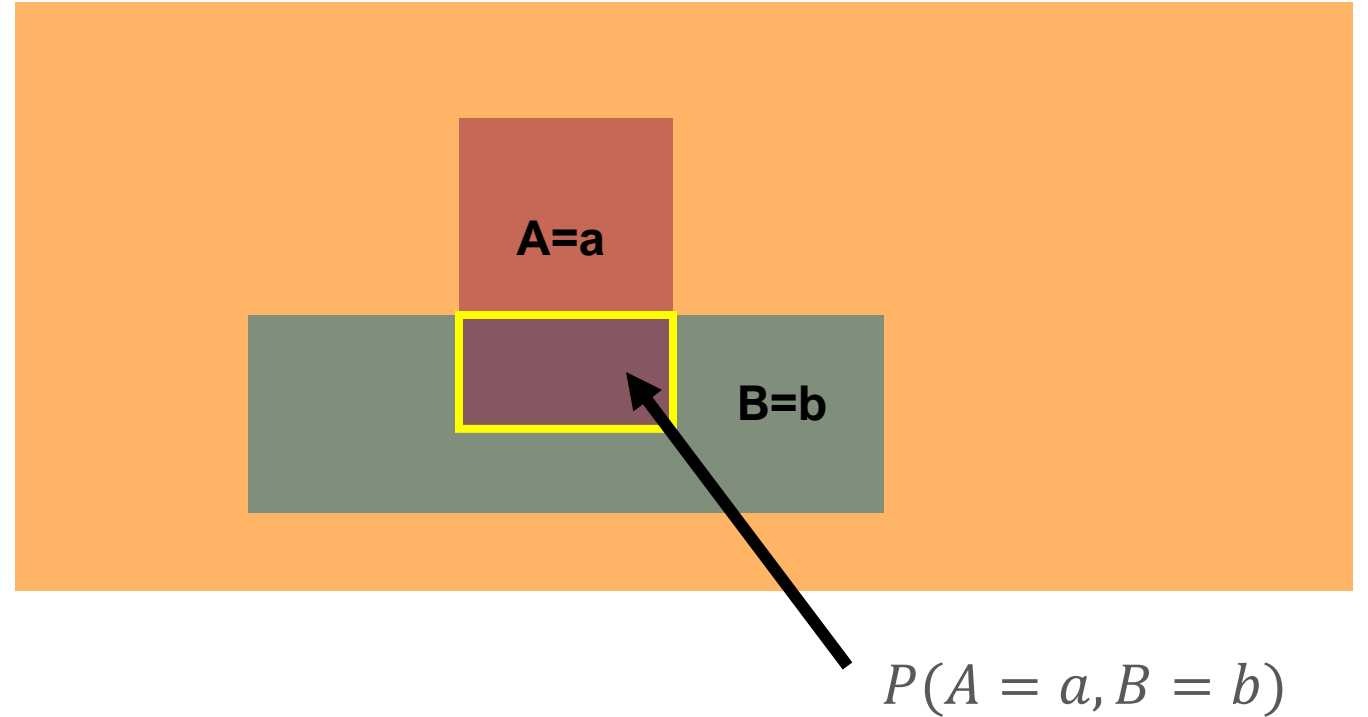
Sum across the entire box is 1

# Conditional Probability

- $P(A = a|B = b)$  is the probability that  $A = a$  when you know that  $B = b$

$$P(A = a|B = b) = \frac{P(A = a, B = b)}{P(B = b)}$$

$$P(B = b|A = a) = \frac{P(A = a, B = b)}{P(A = a)}$$



# Chain Rule

$$P(A = a|B = b) = \frac{P(A = a, B = b)}{P(B = b)}$$

$$P(B = b|A = a) = \frac{P(A = a, B = b)}{P(A = a)}$$

**Chain Rule:**

$$P(A = a, B = b) = P(A = a)P(B = b|A = a) = P(B = b)P(A = a|B = b)$$

# Bayes Rule

$$P(A = a|B = b) = \frac{P(A = a)P(B = b|A = a)}{P(B = b)}$$

# Theorem: Bayes Optimal Classifier

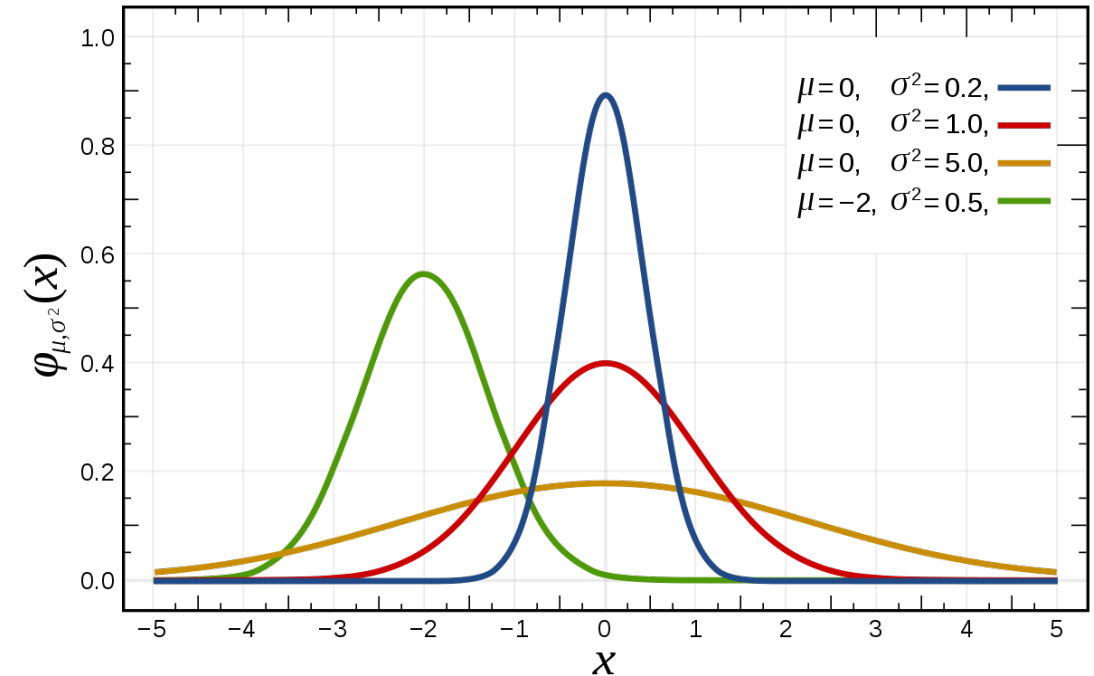
- Bayesian optimal classifier:

$$f^{(BO)}(\hat{x}) = \arg \max_{\hat{y} \in Y} \mathcal{D}(\hat{x}, \hat{y})$$

- Theorem: Bayes Optimal Classifier: The Bayes Optimal Classifier  $f^{(BO)}$  achieves minimal zero/one error of any deterministic classifier.
- Remember: We don't know what  $\mathcal{D}$  is!
- **However:** Maybe we can make a classifier by trying to estimate what  $\mathcal{D}$  actually is

# Constructing a Probability Distribution

- Select a family of parametric distributions
  - For example, the Gaussian distribution is parametric
  - Parameters are mean and covariance
- For learning, the job is to figure out which parameters are best in terms of the training data





# Constructing a Probability Distribution

- You need to make the assumption that the training data we have access to is drawn independently from  $\mathcal{D}$ 
  - If you draw examples  $(x_1, y_1) \sim \mathcal{D}$  and  $(x_2, y_2) \sim \mathcal{D}$  and so on, the  $n$ th draw  $(x_n, y_n)$  is drawn from  $\mathcal{D}$  and does NOT depend on the previous  $n-1$  samples.
- Important Note: This assumption is usually false!
  - It is also usually sufficiently close to being true to be useful
- This assumption along with the assumption that training data is drawn from the same distribution  $\mathcal{D}$  leads to the ***i.i.d. assumption***: independently and identically distributed

# Statistical Estimation

- Suppose you need to model a coin that is maybe biased
  - You can think about this as a label in a binary classification problem
- Suppose you observe HHTH
- You can assume all flips came from the same coin and each flip was independent (thus, i.i.d.)
- You can also assume that the coin has a fixed probability  $\beta$  of coming up heads (and thus  $1 - \beta$  of coming up tails)
  - So, the parameter of the model is just  $\beta$

# Pop Quiz

cs425

# Question 1

- Given that the sequence of coinflips gave you HHTH (and you have no other information), what would you guess  $\beta$  (the probability of heads) is?
  - Note:  $\beta$  is between 0 and 1

# Question 2

- Given that a sequence of coinflips gave you HHTTT (and you have no other information), what would you guess  $\beta$  (the probability of heads) is?

# Statistical Estimation

- HHTH is our data, and we need to find  $\beta$  that maximizes the probability of the observed data under that parameter (maximum likelihood estimation):

$$p_{\beta}(\mathcal{D}) = p_{\beta}(HHTH)$$

Definition of  $\mathcal{D}$

$$= p_{\beta}(H)p_{\beta}(H)p_{\beta}(T)p_{\beta}(H)$$

Data is independent

$$= \beta\beta(1 - \beta)\beta$$

$$= \beta^3(1 - \beta)$$

$$= \beta^3 - \beta^4$$



# Maximizing Likelihood Estimation

- We want the parameter  $\beta$  that maximizes the probability of the data, we take the derivative of  $\beta^3 - \beta^4$  with respect to  $\beta$ , set it to 0 and solve for  $\beta$

$$\frac{\partial}{\partial \beta} [\beta^3 - \beta^4] = 3\beta^2 - 4\beta^3 = 0$$

$$4\beta^3 = 3\beta^2$$

$$4\beta = 3$$

$$\beta = \frac{3}{4}$$

# Maximizing Likelihood Evaluation

- If you have H-many heads and T-many tails, then the probability of the data sequence is  $\beta^H (1 - \beta)^T$
- You can take the derivative of this with respect to  $\beta$  but the products make it difficult
- Instead of maximizing likelihood, we can instead maximize ***log likelihood*** or ***log probability***
- Log likelihood here is:

$$H \log \beta + T \log(1 - \beta)$$

# Maximizing Log Likelihood

- Log likelihood is:

$$H \log \beta + T \log(1 - \beta)$$

- Differentiating with respect to  $\beta$ :

$$\frac{H}{\beta} - \frac{T}{1 - \beta} = 0 \Rightarrow \frac{H}{\beta} = \frac{T}{1 - \beta} \Rightarrow H(1 - \beta) = T\beta \Rightarrow H - H\beta = T\beta$$

$$\Rightarrow \beta = \frac{H}{T + H}$$

# Multi-Class Classification

- Instead of a coin, we're now rolling a K-sided die with probabilities of landing on each side of  $\theta_1, \theta_2, \dots, \theta_K$ :

$$\sum_{k=1}^K \theta_k = 1$$

- Given a particular dataset that has  $x_1$  rolls of 1,  $x_2$  rolls of 2, etc., the probability that this data occurs is:

$$\prod_k \theta_k^{x_k}$$

This has a log probability of:

$$\sum_k x_k \log \theta_k$$

# Multi-Class Classification

- To come up with the parameters that will fit the data, we define this as a constrained optimization problem:

$$\begin{aligned} \min_{\theta} & - \sum_k x_k \log \theta_k \\ \text{subj. to} & \sum_k \theta_k - 1 = 0 \end{aligned}$$

- To map it to a standard minimization problem, we introduce a Lagrange variable corresponding to the constraints and to use it to move the constraint into the objective:

$$\max_{\lambda} \min_{\theta} - \sum_k x_k \log \theta_k - \lambda \left( \sum_k \theta_k - 1 \right)$$

# Lagrangian: What's happening here?

$$\max_{\lambda} \min_{\theta} - \sum_k x_k \log \theta_k - \lambda \left( \sum_k \theta_k - 1 \right)$$

- Think of  $\lambda$  as an adversary
  - $\lambda$  is trying to maximize this function while you're trying to minimize it
- If you pick  $\theta_k$  so that the constraint is satisfied, then  $\lambda$  doesn't matter.
- If it's even slightly unsatisfied, then  $\lambda$  can tend towards  $+\infty$  to blow up the objective



# Naïve Bayes Models

# Naïve Bayes Models

- Consider the binary classification problem
- We want a parameterized probability distribution that can describe the training data
- Suppose our task is to predict whether a movie review is positive or negative based on what words (features) appear in that review.
- The probability of a single data point is:

$$P_{\theta}((y, x)) = P_{\theta}(y, x_1, x_2, \dots, x_D)$$

# Simplify using chain rule

$$\begin{aligned} P_{\theta}(y, x_1, x_2, \dots, x_D) \\ &= P_{\theta}(y)P_{\theta}(x_1|y)P_{\theta}(x_2|y, x_1)P_{\theta}(x_3|y, x_1, x_2) \cdots P_{\theta}(x_D|y, x_1, \dots, x_{D-1}) \\ &= P_{\theta}(y) \prod_d P_{\theta}(x_d|y, x_1, \dots, x_{d-1}) \end{aligned}$$

- This equality is exact for any probability distribution
- But it might be hard to craft a probability distribution for this!
- It might even be difficult to accurately estimate

# So what do we do?

- We make assumptions!
- Classic assumption: the *naïve Bayes* assumption
  - Features are independent, conditioned on the label
  - What does this mean?
  - Once we know the label, the probability of any feature appears is independent of the other features
  - Example: Once we know a movie review is positive, then the probability that “excellent” appears is independent of whether “amazing” appeared

# Naïve Bayes Assumption

$$P(x_d|y, x_{d'}) = p(x_d|y) \quad , \forall d \neq d'$$

This means that we can simplify this:

$$P_{\theta}(y) \prod_d P_{\theta}(x_d|y, x_1, \dots, x_{d-1})$$

To this:

$$P_{\theta}((y, x)) = P_{\theta}(y) \prod_d P_{\theta}(x_d|y)$$

# Naïve Bayes

- Now we're ready to start parameterizing  $P$ .
- Suppose that our labels are binary, and our features are also binary.
- We can model each label as a biased coin with a probability of heads (e.g., positive review) given by  $\theta_0$ .
- For each label, you can imagine having one biased coin for each feature.
  - With  $D$ -many features, you have  $1+2D$  total coins.
  - One for the label  $\theta_0$
  - And one for each label/feature combination:  $\theta_{d,+1}$  and  $\theta_{d,-1}$

# Naïve Bayes

$$\begin{aligned} P_{\theta}((y, x)) &= P_{\theta}(y) \prod_d P_{\theta}(x_d | y) \\ &= \theta_0^{[y=+1]} (1 - \theta_0)^{[y=-1]} \prod_d \theta_{d,(y)}^{[x_d=1]} (1 - \theta_{d,(y)})^{[x_d=0]} \end{aligned}$$

- Solving for  $\theta_0$  is identical to solving for the biased coin from before
  - It's just the relative frequency of positive labels in your data
  - $\theta_0$  doesn't depend on  $x$  at all

# Solving for Distribution Parameters

$$\theta_0 = \frac{1}{N} \sum_n [y_n = +1]$$

$$\theta_{d,+1} = \frac{\sum_n [y_n = +1 \wedge x_{n,d} = 1]}{\sum_n [y_n = +1]}$$

Count of all instances where feature  $x_{n,d} = 1$  and  $y_n = 1$

Count of all instances where  $y_n = 1$

$$\theta_{d,-1} = \frac{\sum_n [y_n = -1 \wedge x_{n,d} = 1]}{\sum_n [y_n = -1]}$$



# Solving for Distribution Parameters

$$\theta_0 = \frac{1}{N} \sum_n [y_n = +1]$$

$$\theta_{d,+1} = \frac{\sum_n [y_n = +1 \wedge x_{n,d} = 1]}{\sum_n [y_n = +1]}$$

$$\theta_{d,-1} = \frac{\sum_n [y_n = -1 \wedge x_{n,d} = 1]}{\sum_n [y_n = -1]}$$

Count of all instances where feature  $x_{n,d} = 1$  and  $y_n = -1$

Count of all instances where  $y_n = -1$

# What about non-binary features?

- With binary features, we chose the Bernoulli distribution, which is a distribution over independent coin flips
- For non-binary features, we need to choose a different model for  $p(x_d|y)$ 
  - The die example from before is a discrete distribution
  - If the data is continuous, you might choose a Gaussian (normal) distribution
- The choice of distribution is a form of inductive bias!
  - You can use the choice of distribution to inject knowledge of the problem into the learning algorithm

# Bernoulli Distribution

- Models binary outcomes like coinflips
- Parameterized by a single scalar value  $\theta \in [0,1]$ , which represents the probability of heads:  $Ber(\theta)$
- The likelihood function is:

$$Ber(x|\theta) = \theta^x (1 - \theta)^{1-x}$$

# Discrete Distribution

- Generalization of Bernoulli to more than two possible outcomes (like rolls of the die)
- Parameterized by a set of scalar values  $\theta \in \mathbb{R}^k$ , which represents the probabilities of each side of the die coming up:  $\text{Disc}(\theta)$ 
  - $\theta_k$  is the probability that the side of the die comes up on  $k$
  - $\sum_k \theta_k = 1$
- Likelihood function:

$$\text{Disc}(x|\theta) = \prod_k \theta_k^{1[x=k]}$$

# Binomial Distribution

- Just like the Bernoulli distribution, except for multiple flips of the coin instead of a single flip
- Likelihood is:

$$\binom{Ber(k | n, \theta) = n}{k \theta^k (1 - \theta)^{n-k}}$$

Where  $n$  is the number of flips and  $k$  is the number of heads

# Multinomial Distribution

- Extends the Discrete distribution to multiple rolls
- Likelihood is:

$$Mult(x|n, \theta) = \frac{n!}{\prod_k x_k!} \prod_k \theta_k^{x_k}$$

Where  $n$  is the number of rolls and  $x_k$  is the number of times the die came up on side  $k$

# Two Common Continuous Distributions

- Uniform distribution:  $Uni(a, b)$ , which is uniform over the closed range  $[a, b]$ 
  - Density function is  $Uni(x|a, b) = \frac{1}{b-a} 1[x \in [a, b]]$
- Gaussian distribution: parameterized by mean  $\mu$  and variance  $\sigma^2$ 
  - Density function is  $Nor(x | \mu, \sigma^2) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left[-\frac{1}{2\sigma^2} (x - \mu)^2\right]$

# Predicting with Naïve Bayes

- Let's assume we have the naïve Bayes model with the Bernoulli features
- What does its decision boundary look like?
- For probabilistic models, the decision boundary is the set of inputs for which the likelihood of  $y=+1$  is precisely 50%
  - The set of inputs  $x$  for which:  $\frac{P(y = +1|x)}{P(y = -1|x)} = 1$
  - This simplifies to:  $\frac{P(y=+1,x)}{P(y=-1,x)} = 1$
- Instead of computing this, it's easier to compute the log-likelihood ratio...



# Log-Likelihood Ratio

$$\begin{aligned} LLR &= \log \left[ \theta_0 \prod_d \theta_{d,+1}^{[x_d=1]} (1 - \theta_{d,+1})^{[x_d=0]} \right] \\ &\quad - \log \left[ (1 - \theta_0) \prod_d \theta_{d,-1}^{[x_d=1]} (1 - \theta_{d,-1})^{[x_d=0]} \right] \\ &= \log \theta_0 - \log(1 - \theta_0) + \sum_d [x_d = 1] (\log \theta_{d,+1} - \log \theta_{d,-1}) \\ &\quad + \sum_d [x_d = 0] (\log(1 - \theta_{d,+1}) - \log(1 - \theta_{d,-1})) \end{aligned}$$

$\log(ab)$   
 $= \log(a)$   
 $+ \log(b)$

# Log-Likelihood Ratio

$$\begin{aligned} & \log \theta_0 - \log(1 - \theta_0) + \sum_d [x_d = 1](\log \theta_{d,+1} - \log \theta_{d,-1}) \\ & + \sum_d [x_d = 0](\log(1 - \theta_{d,+1}) - \log(1 - \theta_{d,-1})) \\ & = \sum_d x_d \log \frac{\theta_{d,+1}}{\theta_{d,-1}} + \sum_d (1 - x_d) \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} + \log \frac{\theta_0}{1 - \theta_0} \\ & = \sum_d x_d \left[ \log \frac{\theta_{d,+1}}{\theta_{d,-1}} - \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} \right] + \sum_d \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} + \log \frac{\theta_0}{1 - \theta_0} \end{aligned}$$

# Log-Likelihood Ratio

$$\begin{aligned} & \sum_d x_d \log \frac{\theta_{d,+1}}{\theta_{d,-1}} + \sum_d (1 - x_d) \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} + \log \frac{\theta_0}{1 - \theta_0} \\ &= \sum_d x_d \left[ \log \frac{\theta_{d,+1}}{\theta_{d,-1}} - \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} \right] + \sum_d \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} + \log \frac{\theta_0}{1 - \theta_0} \end{aligned}$$

Group  $x_d$  terms together

# Why did we do all of that math?

$$\sum_d x_d \left[ \log \frac{\theta_{d,+1}}{\theta_{d,-1}} - \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} \right] + \sum_d \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} + \log \frac{\theta_0}{1 - \theta_0}$$
$$= x \cdot w + b$$

$$w_d = \log \frac{\theta_{d,+1}(1 - \theta_{d,-1})}{\theta_{d,-1}(1 - \theta_{d,+1})}, \quad b = \sum_d \log \frac{1 - \theta_{d,+1}}{1 - \theta_{d,-1}} + \log \frac{\theta_0}{1 - \theta_0}$$

**The Naïve Bayes Model is a linear model!**

# Generative Stories

# Generative Stories

- We can tell a generative story about how we believe our training data came into existence
  - This is definitely a fictional story, but a convenient one, nonetheless
- Consider a multi-class classification problem with continuous features modeled by independent Gaussians
- Since the label can take values  $1, \dots, K$ , you can use a discrete distribution to model it

# Generative Stories

- For each example  $n = 1, \dots, N$ :
  - Choose a label  $y_n \sim \text{Disc}(\theta)$
  - For each feature  $d = 1, \dots, D$ :
    - Choose a feature value  $x_{n,d} \sim \text{Nor}(\mu_{y_n,d}, \sigma_{y_n,d}^2)$
- This generative story can be translated into a likelihood function by replacing the “for each” with products:

$$P(D) = \prod_n \underbrace{\theta_{y_n}}_{\text{Choose Label}} \prod_d \underbrace{\frac{1}{\sqrt{2\pi\sigma_{y_n,d}^2}} \exp\left[-\frac{1}{2\sigma_{y_n,d}^2} (x_{n,d} - \mu_{y_n,d})^2\right]}_{\text{Choose feature value}}$$

For each example

For each feature

# Generative Stories

$$P(D) = \prod_n \theta_{y_n} \prod_d \frac{1}{\sqrt{2\pi\sigma_{y_n,d}^2}} \exp\left[-\frac{1}{2\sigma_{y_n,d}^2} (x_{n,d} - \mu_{y_n,d})^2\right]$$

- Take logs to get to log-likelihood:

$$\log P(D) = \sum_n \left[ \log \theta_{y_n} + \sum_d -\frac{1}{2} \log(\sigma_{y_n,d}^2) - \frac{1}{2\sigma_{y_n,d}^2} (x_{n,d} - \mu_{y_n,d})^2 \right] + \text{const}$$



# Generative Stories

- To optimize for  $\theta$ , we add the “sums to one” constraint and this ends up leading to where  $\theta_k$  is proportional to the number of examples with label  $k$
- To optimize for the  $\mu$ s, you have to take the derivative:

$$\frac{\partial \log P(D)}{\partial \mu_{k,i}} = \sum_{n:y_n=k} \frac{1}{\sigma_{k,d}^2} (x_{n,i} - \mu_{k,i})$$

- If we set this equal to 0 and solve:

$$\mu_{k,i} = \frac{\sum_{(n:y_n=k)} x_{n,i}}{\sum_{(n:n=k)} 1}$$

- This is the sample mean of the  $i$ th feature of the data points that fall in class  $k$

# Generative Stories

- You can do similar math to solve for  $\sigma_{k,i}^2$  :

$$\sigma_{k,i}^2 = \frac{\sum_{n:y_k=k} (x_{n,i} - \mu_{k,i})^2}{\sum_{n:y_k=k} 1}$$

- This is just the sample variance of feature  $i$  for class  $k$

# Conditional Models

- Previously, our task was formulated as attempting to model the joint distribution of  $(x,y)$  pairs
- At prediction time, all we actually care about is  $P(y|x)$
- Can we just model this directly?

# Regression

- Suppose we believe that the relationship between a real value  $y$  and the vector  $x$  should be linear
- You should expect that this holds for some parameters  $(w, b)$ :
$$y = w \cdot x + b$$
- You can think of deviations from  $y = w \cdot x + b$  as noise

# Noise

- To form a probabilistic model, we must assume some distribution over noise
- A convenient choice is zero-mean Gaussian noise
- Now we have a generative story like this:

For each example  $n = 1, \dots, N$ :

    Compute  $t_n = w \cdot x_n + b$

    Choose noise  $e_n \sim \text{Nor}(0, \sigma^2)$

    Return  $y_n = t_n + e_n$

- Here,  $t_n$  is the target (the noiseless variable we don't get to observe)
- The value that we actually get to observe is  $y_n$

# Noise

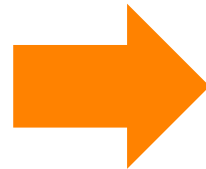
- Gaussian distribution has the additive property:  
 $a \sim \text{Nor}(\mu, \sigma^2)$  and  $b = a + c$ , then  $b \sim \text{Nor}(\mu + c, \sigma^2)$
- This lets us re-write the generative story:

For each example  $n = 1, \dots, N$ :

Compute  $t_n = w \cdot x_n + b$

Choose noise  $e_n \sim \text{Nor}(0, \sigma^2)$

Return  $y_n = t_n + e_n$



For each example  $n = 1, \dots, N$ :

Choose  $y_n \sim \text{Nor}(w \cdot x_n + b, \sigma^2)$

# Log Likelihood

- $y_n \sim \text{Nor}(w \cdot x_n + b, \sigma^2)$
- The log likelihood of a dataset from this generative story is:

$$\log P(D) = \sum_n \left[ -\frac{1}{2} \log \sigma^2 - \frac{1}{2\sigma^2} (w \cdot x_n + b - y_n)^2 \right]$$

- Removing constants, we get:

$$\log P(D) = -\frac{1}{2\sigma^2} \sum_n (w \cdot x_n + b - y_n)^2 + \text{const}$$

# Log Likelihood

Optimizing for this  
is just linear  
regression!

- $y_n \sim \text{Nor}(w \cdot x_n + b, \sigma^2)$
- The log likelihood of a dataset from this generative story is:

$$\log P(D) = \sum_n \left[ -\frac{1}{2} \log \sigma^2 - \frac{1}{2\sigma^2} (w \cdot x_n + b - y_n)^2 \right]$$

- Removing constants, we get:

$$\log P(D) = -\frac{1}{2\sigma^2} \sum_n (w \cdot x_n + b - y_n)^2 + \text{const}$$



# What about binary classification?

- Using Gaussian noise doesn't make sense.
- Bernoulli makes more sense, but now we need a parameter for the Bernoulli distribution (the probability of “heads”)
  - Your model has to produce a value between 0 and 1 for this
- You could produce a real-valued target and then transform to a value between 0 and 1

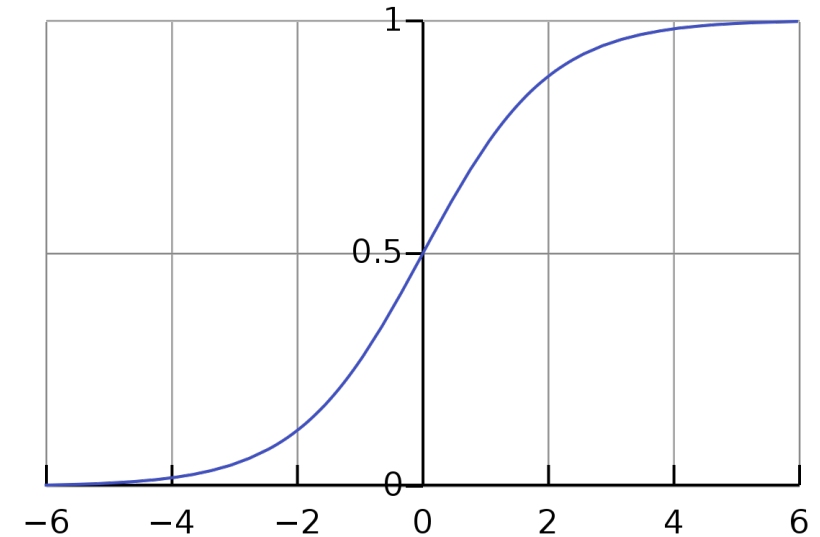
# What about binary classification?

- The logistic function takes a real value and maps it between 0 and 1:

$$\sigma(z) = \frac{\exp z}{1 + \exp z}$$

- It also has a few nice properties:

- $\sigma(-z) = 1 - \sigma(z)$
- $\frac{\partial \sigma}{\partial z} = \sigma(z)(1 - \sigma(z))$



# Generative Story for Binary Classification

For each example  $n = 1, \dots, N$ :

    Compute  $t_n = \sigma(w \cdot x_n + b)$

    Compute  $z_n \sim \text{Ber}(t_n)$

    Return  $y_n = 2z_n - 1$

The log likelihood for this model is:

$$\log P(D) = \sum_n [y_n = +1] \log \sigma(w \cdot x_n + b) + [y_n = -1] \log \sigma(-w \cdot x_n + b)$$

# Log Likelihood for Binary Classification

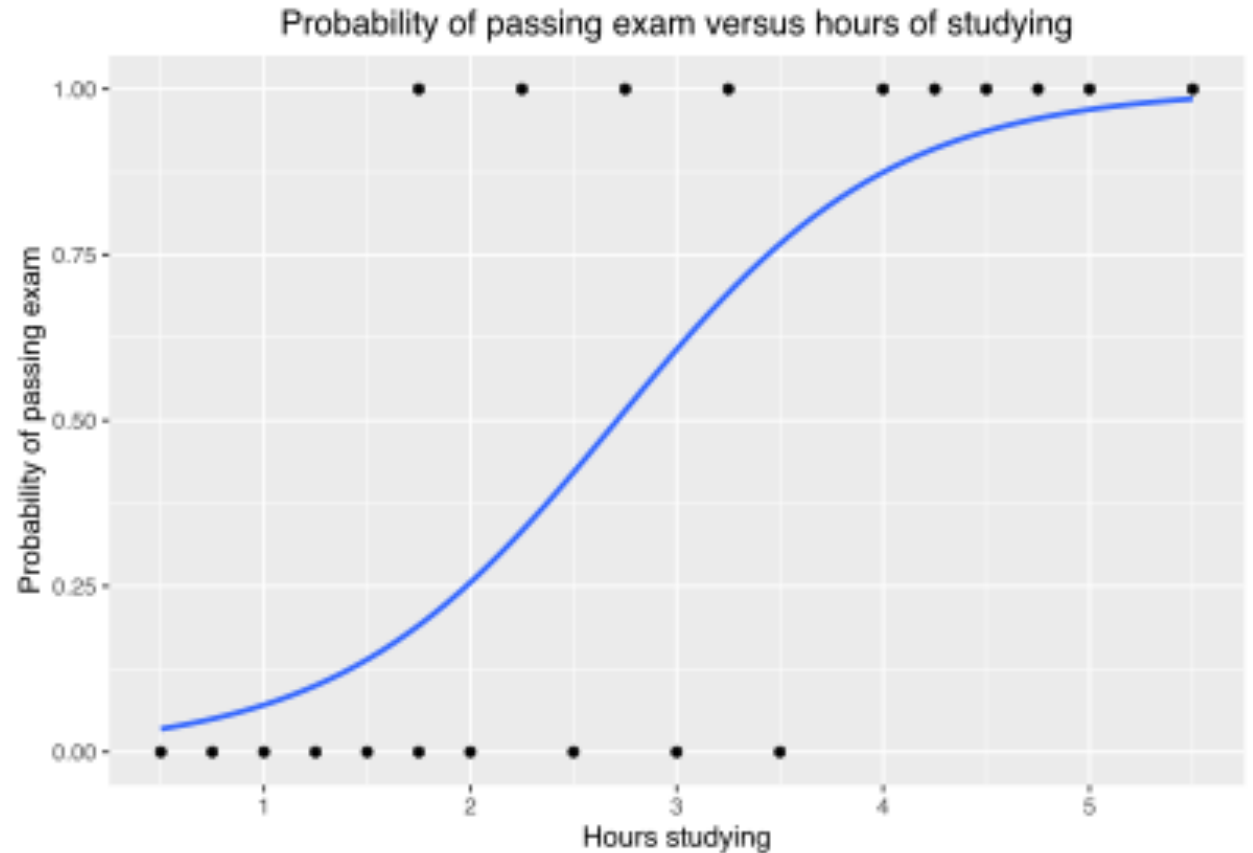
The log likelihood for this model is:

$$\begin{aligned}\log P(D) &= \sum_n [y_n = +1] \log \sigma(w \cdot x_n + b) + [y_n = -1] \log \sigma(-w \cdot x_n + b) \\ &= \sum_n \log \sigma(y_n(w \cdot x_n + b)) \\ &= - \sum_n \log[1 + \exp(-y_n(w \cdot x_n + b))] \\ &= - \sum_n \ell^{(\log)}(y_n, w \cdot x_n + b)\end{aligned}$$

This is logistic regression

# Logistic Regression

- Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.
- Since the outcome is a probability, the dependent variable is bounded between 0 and 1.



# Takeaways for Naïve Bayes

- For Naïve Bayes, we're making the assumption of conditional independence across features so that

$$P_{\theta}((y, x)) = P_{\theta}(y) \prod_d P_{\theta}(x_d | y)$$

- There are different distributions for modeling what's happening with  $P_{\theta}$ 
  - Gaussian for continuous
  - Bernoulli for binary
  - Discrete distribution for categorical
- When we're training for any of these models, we're looking for the parameters of the distribution to maximize the loglikelihood of that probability

# Naïve Bayes

SKLearn

# Announcements

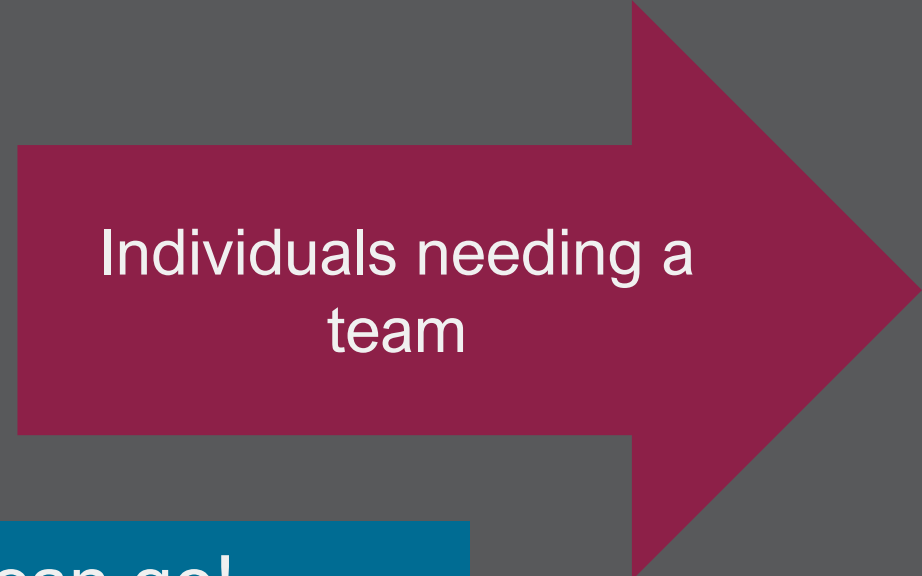
- Lab 4 posted today and due on October 27
- Final Projects:
  - **Deadline for team selection is October 19**
    - Email me your team with subject line: COSC 425 Team Project Members
    - One person from the team should email me by then and copy the other members
    - **5-point penalty on the final project if your team is not defined by October 19**
  - Finalization of dataset to be used by **November 3**.
    - 5-point penalty if the dataset is not selected by that date.



# Team Project Matching



Teams of 2 needing a  
member



Individuals needing a  
team



Teams that are full can go!