CMSC 478
Machine Learning

KMA Solaiman
ksolaima@umbc.edu

Midterm Review

Supervised Learning

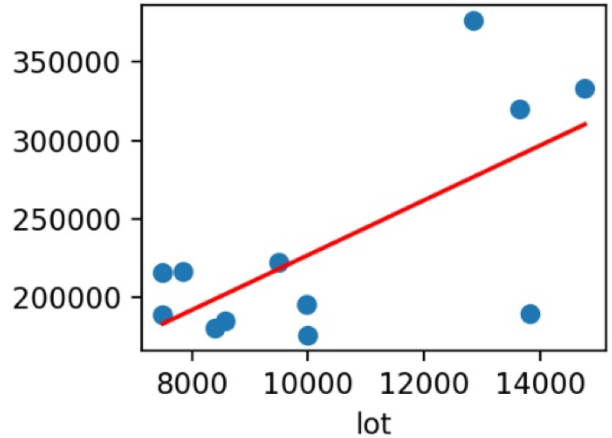
- A **hypothesis** or a prediction function is function $h : \mathcal{X} \to \mathcal{Y}$
  - $\mathcal{X}$ is an image, and $\mathcal{Y}$ contains "cat" or "not."
  - $\mathcal{X}$ is a text snippet, and $\mathcal{Y}$ contains "hate speech" or "not."
  - $\mathcal{X}$ is house data, and $\mathcal{Y}$ could be the price.
- A **training set** is a set of pairs $\{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})$ s.t. $x^{(i)} \in \mathcal{X}$ and $y^{(i)} \in \mathcal{Y}$ for $i = 1, \dots, n$.
- Given a training set our goal is to produce a *good* prediction function h
  - Defining "good" will take us a bit. It's a modeling question!
  - We will want to use h on *new* data not in the training set.

- If $\mathcal{Y}$ is continuous, then called a *regression problem*.
- If $\mathcal{Y}$ is discrete, then called a *classification problem*.

How do we represent h? (One popular choice)

$h(x) = \theta_0 + \theta_1 x_1$ is an *affine function*

Visual version of linear regression



Let $h_\theta(x) = \sum_{j=0}^{d} \theta_j x_j$ want to choose θ so that $h_\theta(x) \approx y$. One popular idea called **least squares**

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 .$$

Choose

$$\theta = \underset{\theta}{\operatorname{argmin}}\, J(\theta).$$

Solving the least squares optimization problem.

Gradient Descent

$$\theta^{(0)} = 0$$

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) \qquad \text{for } j = 0, \ldots, d.$$

Gradient Descent Computation

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) \text{ for } j = 0, \ldots, d.$$

Note that α is called the **learning rate** or **step size**.

Let's compute the derivatives. . .

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) &= \sum_{i=1}^{n} \frac{1}{2} \frac{\partial}{\partial \theta_j} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\ &= \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \frac{\partial}{\partial \theta_j} h_\theta(x^{(i)}) \end{aligned}$$

Gradient Descent Computation

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) \text{ for } j = 0, \ldots, d.$$

Note that α is called the **learning rate** or **step size**.

Let's compute the derivatives...

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta^{(t)}) &= \sum_{i=1}^{n} \frac{1}{2} \frac{\partial}{\partial \theta_j} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\ &= \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \frac{\partial}{\partial \theta_j} h_\theta(x^{(i)}) \end{aligned}$$

For our *particular* h_θ we have:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_d x_d \text{ so } \frac{\partial}{\partial \theta_j} h_\theta(x) = x_j$$

Gradient Descent Computation

Thus, our update rule for component j can be written:

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}.$$

Supervised Learning and Classification

- Linear Regression via a Probabilistic Interpretation
- Logistic Regression
- Optimization Method: Newton's Method

We'll learn the maximum likelihood method (a probabilistic interpretation) to generalize from linear regression to more sophisticated models.

Notation for Guassians **in our Problem**

Recall in our model,

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)} \text{ in which } \varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2) \text{ .... (11.1)}$$

or more compactly notation:

$$y^{(i)} \mid x^{(i)}; \theta \sim \mathcal{N}(\theta^T x, \sigma^2) \text{.............. (11.2)}$$

equivalently, **Probability distribution** over $y^{(i)}$, given $x^{(i)}$ and parameterized by θ

$$P\left(y^{(i)} \mid x^{(i)}; \theta\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(y^{(i)} - x^{(i)}\theta)^2}{2\sigma^2}\right\} \text{ ...... (11.3)}$$

- We **condition** on $x^{(i)}$.
- In contrast, θ **parameterizes** or "picks" a distribution.

We use bar (|) versus semicolon (;) notation above.

(Log) Likelihoods!

Intuition: among many distributions, pick the one that agrees with the data the most (is most "likely").

$$L(\theta) = p(y|X;\theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)};\theta) \qquad \text{iid assumption}$$

$$= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{(x^{(i)}\theta - y^{(i)})^2}{2\sigma^2} \right\}$$

For convenience, we use the *Log Likelihood* $\ell(\theta) = \log L(\theta)$.

$$\ell(\theta) = \sum_{i=1}^{n} \log \frac{1}{\sigma\sqrt{2\pi}} - \frac{(x^{(i)}\theta - y^{(i)})^2}{2\sigma^2}$$

$$= n \log \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x^{(i)}\theta - y^{(i)})^2 = C(\sigma, n) - \frac{1}{\sigma^2} J(\theta)$$

where $C(\sigma, n) = n \log \frac{1}{\sigma\sqrt{2\pi}}$.

(Log) Likelihoods!

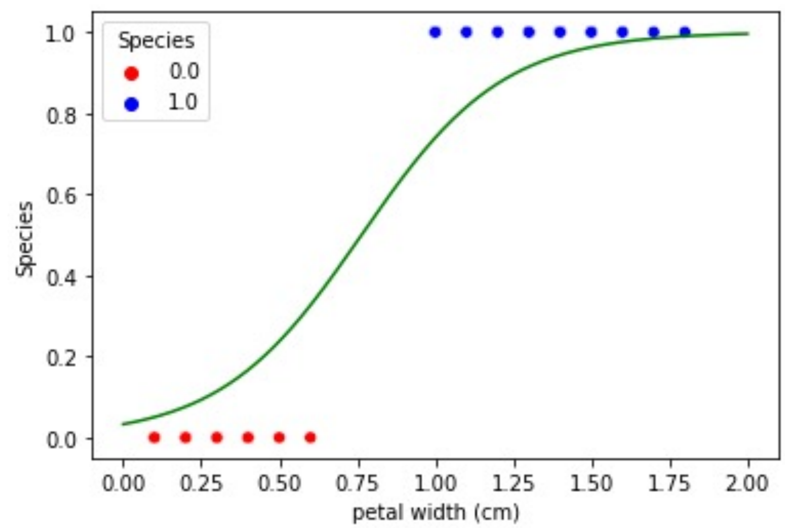
So we've shown that finding a θ to maximize $L(\theta)$ is the same as *maximizing*

$$\ell(\theta) = C(\sigma, n) - \frac{1}{\sigma^2} J(\theta)$$

Or minimizing, $J(\theta)$ directly (why?)

Takeaway: "Under the hood," solving least squares *is* solving a maximum likelihood problem for a particular probabilistic model.

This view shows a path to generalize to new situations!



Graph of Iris Dataset with logistic regression

Logistic Regression: Link Functions

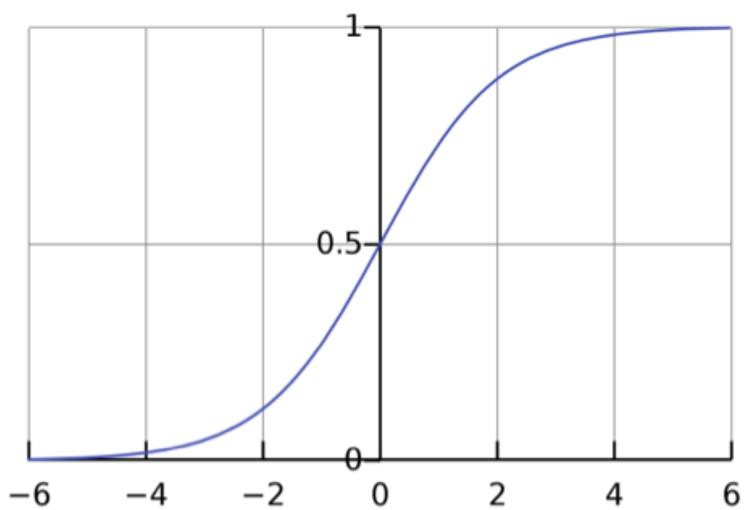
Given a training set $\{(x^{(i)}, y^{(i)}) \text{ for } i = 1, \ldots, n\}$ let $y^{(i)} \in \{0, 1\}$. Want $h_\theta(x) \in [0, 1]$. Let's pick a smooth function:

$$h_\theta(x) = g(\theta^T x)$$

Here, g is a link function. There are *many*. . . but we'll pick one!

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

SIGMOID



How do we interpret $h_\theta(x)$?

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Logistic Regression: Link Functions

Let's write the Likelihood function. Recall:

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

Then,

$$
\begin{aligned}
L(\theta) &= P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{n} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \quad \text{exponents encode ``if-then''}
\end{aligned}
$$

Taking logs to compute the log likelihood $\ell(\theta)$ we have:

$$
\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))
$$

Now to solve it. . .

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

We maximize for θ but we already saw how to do this! Just compute derivative, run (S)GD and you're done with it!

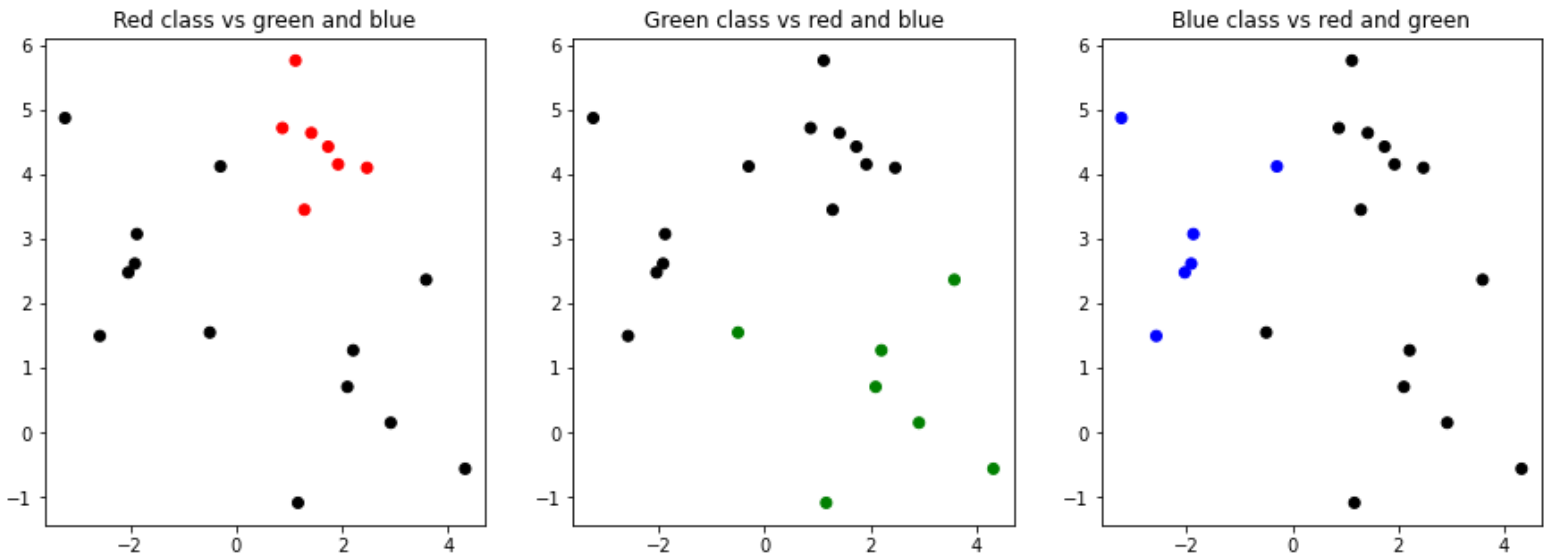
Takeaway: This is *another* example of the max likelihood method: we setup the likelihood, take logs, and compute derivatives.

Optimization Method Summary

Method	Compute per Step	Number of Steps to convergence
SGD	$\theta(d)$	$\approx \epsilon^{-2}$
Minibatch SGD		
GD	$\theta(nd)$	$\approx \epsilon^{-1}$
Newton	$\Omega(nd^2)$	$\approx \log(1/\epsilon)$

- In classical stats, d is small (< 100), n is often small, and *exact parameters matter*
- In modern ML, d is huge (billions, trillions), n is huge (trillions), and parameters used *only* for prediction
  - These are approximate number of computing steps
  - Convergence happens when loss settles to within an error range around the final value.
  - Newton would be very fast, where SGD needs a lot of step, but individual steps are fast, makes up for it
- As a result, (minibatch) SGD is the *workhorse* of ML.

1 vs All


Red class vs green and blue
Green class vs red and blue
Blue class vs red and green
6
5
4
3
2
1
0
−1
−2
0
2
4

Multiclass

Suppose we want to choose among k discrete values, e.g., $\{\text{'Cat'}, \text{'Dog'}, \text{'Car'}, \text{'Bus'}\}$ so $k = 4$.

We encode with **one-hot** vectors i.e. $y \in \{0,1\}^k$ and $\sum_{j=1}^{k} y_j = 1$.

$$\underset{\text{'Cat'}}{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}} \quad \underset{\text{'Dog'}}{\begin{pmatrix}0\\1\\0\\0\end{pmatrix}} \quad \underset{\text{'Car'}}{\begin{pmatrix}0\\0\\1\\0\end{pmatrix}} \quad \underset{\text{'Bus'}}{\begin{pmatrix}0\\0\\0\\1\end{pmatrix}}$$

A prediction here is actually a *distribution* over the k classes. This leads to the SoftMax function described below (derivation in the notes!). That is our hypothesis is a vector of k values:

$$P(y = j | x; \bar{\theta}) = \frac{\exp(\theta_j^T x)}{\sum_{i=1}^{k} \exp(\theta_i^T x)}.$$

Here each θ_j has the *same dimension* as x, i.e., $x, \theta_j \in R^{d+1}$ for $j = 1, \ldots, k$.

How do you train multiclass?

Fixing x and θ, our output is a vector $\hat{p} \in \mathbb{R}_+^k$ s.t. $\sum_{j=1}^k \hat{p}_j = 1$.

$$\hat{p}_j = P(y = j | x; \theta) = \frac{\exp(\theta_j^T x)}{\sum_{i=1}^k \exp(\theta_i^T x)}.$$

Formally, we maximize the probability of the given class!
We can view as CrossEntropy:

$$\text{CrossEntropy}(p, \hat{p}) = -\sum_j p(x = j) \log \hat{p}(x = j).$$

Here, p is the label, which is a one-hot vector.Thus, if the label is i, this formula reduces to:

$$-\log \hat{p}(x = i) = -\log \frac{\exp(\theta_i^T x)}{\sum_{j=1}^k \exp(\theta_j^T x)}.$$

We minimize this–and you've seen the movie, it works the same as the others!

Summary for binary classification/ logistic regression

- Calculate $h_\theta(x) = g(\theta^T x)$
- Get $P(y \mid X; \theta)$ using $h_\theta(x)$, that's likelihood
- Calculate log likelihood from there
- Maximize log likelihood from there –
  use SGD to maximize for θ
    - Start with a guess for θ
    - Keep updating with the rule until convergence

Discriminative Approach

Predicted Output

inference $h_\theta(x)$ is the **output**.

learn $\max_\theta \log p(y \mid x; \theta)$ by maximum likelihood.

algorithm: SGD $\theta^{(t+1)} = \theta^{(t)} + \alpha \left(y^{(i)} - h_{\theta^{(t)}}(x^{(i)}) \right) x^{(i)}$.

Other Forms of Bayes Rule

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

$$P(A|B) = \frac{P(B \mid A)P(A)}{P(B \mid A)P(A) + P(B \mid \sim A)P(\sim A)}$$

$$P(A|B \wedge X) = \frac{P(B \mid A \wedge X)P(A \wedge X)}{P(B \wedge X)}$$

Discriminative vs Generative Models

Discriminative Models	**Generative Models**
Directly learn the function mapping $h: X \rightarrow y$ **or,** Calculate likelihood $P(y\|X)$	Calculate $P(y\|X)$ **from** $P(X\|y)$ **and** $P(y)$ **But Joint Distribution** $P(X, y) = P(X\|y)\, P(y)$
1. Assume some functional form for $P(y\|X)$ 2. Estimate parameters of $P(y\|X)$ directly from training data	1. Assume some functional form for $P(y), P(X\|y)$ 2. Estimate parameters of $P(X\|y), P(y)$ directly from training data 3. Use Bayes rule to calculate $P(y\|X)$

How many parameters must we estimate?

Suppose $X = \langle X_1, \ldots X_n \rangle$
where X_i and Y are boolean RV's

$P(Y|X_1 \ldots X_n)$

Gender	HrsWorked	P(rich | G,HW)	P(poor | G,HW)
F	<40.5	.09	.91
F	>40.5	.21	.79
M	<40.5	.23	.77
M	>40.5	.38	.62

To estimate $P(Y| X_1, X_2, \ldots X_n)$

2^n

If we have 30 X_i's instead of 2?

$2^{30} \sim 1$ Billion

Can we reduce params using Bayes Rule?

Suppose $X = \langle X_1, \ldots X_n \rangle$
where X_i and Y are boolean RV's

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

How many parameters to define $P(X_1, \ldots X_n \mid Y)$?

$P(X|Y=1)$ ----- $2^n - 1$
$P(X|Y=0)$ ----- $2^n - 1$

How many parameters to define $P(Y)$?

Can we reduce params using Bayes Rule?

Suppose X =<X_1,… X_n>
where X_i and Y are boolean RV's

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

how many params for $P(X_1 \cdots X_n|Y)$ $(2^n-1) \cdot 2$

how many for $P(Y) = 1$

Naïve Bayes in a Nutshell

Bayes rule:

$$P(Y = y_k | X_1 \ldots X_n) = \frac{P(Y = y_k) P(X_1 \ldots X_n | Y = y_k)}{\sum_j P(Y = y_j) P(X_1 \ldots X_n | Y = y_j)}$$

Assuming conditional independence among X_i' s:

$$P(Y = y_k | X_1 \ldots X_n) = \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}$$

So, to pick most probable Y for $X^{new} = < X_1, \ldots, X_n >$

$$Y^{new} \leftarrow \arg\max_{y_k} \; P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)$$

Principles for Estimating Probabilities

Principle 1 (maximum likelihood):

- choose parameters θ that maximize $P(\text{data} \mid \theta)$

Principle 2 (maximum a posteriori prob.):

- choose parameters θ that maximize

$$P(\theta \mid \text{data}) = \frac{P(\text{data} \mid \theta)\, P(\theta)}{P(\text{data})}$$

Maximum Likelihood Estimation

X=1 X=0

$P(X=1) = \theta$ $P(X=0) = (1-\theta)$

Data D: $= \{1 \; 0 \; 0 \; 1\} \; 1$

$P(D|\theta) = \theta \cdot (1-\theta) \cdot (1-\theta) \cdot \theta \cdot \theta = \theta^{\alpha_1}(1-\theta)^{\alpha_0}$

Flips produce data D with α_1 heads, α_0 tails

- flips are independent, identically distributed 1's and 0's (Bernoulli)
- α_1 and α_0 are counts that sum these outcomes (Binomial)

$$P(D|\theta) = P(\alpha_1, \alpha_0|\theta) = \theta^{\alpha_1}(1-\theta)^{\alpha_0}$$

Maximum Likelihood Estimate for Θ

$$\hat{\theta} = \arg\max_{\theta} \ln P(\mathcal{D} \mid \theta)$$
$$= \arg\max_{\theta} \ln \theta^{\alpha_H} (1-\theta)^{\alpha_T}$$

- Set derivative to zero: $\frac{d}{d\theta} \ln P(\mathcal{D} \mid \theta) = 0$

[C. Guestrin]

$$\hat{\theta} = \arg\max_{\theta} \ \ln P(D|\theta)$$

$$= \arg\max_{\theta} \ \ln \left[\theta^{\alpha_1}(1-\theta)^{\alpha_0}\right]$$

- Set derivative to zero: $\frac{d}{d\theta} \ln P(\mathcal{D} \mid \theta) = 0$

hint: $\frac{\partial \ln \theta}{\partial \theta} = \frac{1}{\theta}$

$$\frac{\partial}{\partial \theta} \alpha_1 \ln \theta + \alpha_0 \ln(1-\theta)$$

$$\alpha_1 \frac{1}{\theta} + \alpha_0 \frac{\partial \ln(1-\theta)}{\partial \theta}$$

$$\underbrace{\frac{\partial \ln(1-\theta)}{\partial(1-\theta)}}_{\frac{1}{1-\theta}} \cdot \underbrace{\frac{\partial(1-\theta)}{\partial \theta}}_{-1}$$

$$0 = \alpha_1 \frac{1}{\theta} - \frac{\alpha_0}{1-\theta}$$

$$\theta = \frac{\alpha_1}{\alpha_1 + \alpha_0}$$

Summary:
Maximum Likelihood Estimate

X=1 X=0

P(X=1) = θ
P(X=0) = 1-θ
(Bernoulli)

- Each flip yields boolean value for X

$$X \sim \text{Bernoulli: } P(X) = \theta^X (1-\theta)^{(1-X)}$$

- Data set D of independent, identically distributed (iid) flips produces α_1 ones, α_0 zeros (Binomial)

$$P(D|\theta) = P(\alpha_1, \alpha_0|\theta) = \theta^{\alpha_1}(1-\theta)^{\alpha_0}$$

$$\hat{\theta}^{MLE} = \text{argmax}_\theta \, P(D|\theta) = \frac{\alpha_1}{\alpha_1+\alpha_0}$$

Beta prior distribution – P(θ)

- $$P(\theta) = \frac{\theta^{\beta_H - 1}(1-\theta)^{\beta_T - 1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$$
- Likelihood function: $P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$
- Posterior: $P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta) P(\theta)$

Beta prior distribution – P(θ)

- $$P(\theta) = \frac{\theta^{\beta_H - 1}(1-\theta)^{\beta_T - 1}}{B(\beta_H, \beta_T)} \sim Beta(\beta_H, \beta_T)$$
- Likelihood function: $P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$
- Posterior: $P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta) P(\theta)$

$$\propto \theta^{\alpha_H + \beta_H - 1}(1-\theta)^{\alpha_T + \beta_T - 1}$$

$$\hat{\theta}^{MAP} = \frac{(\alpha_H + \beta_H - 1)}{(\alpha_H + \beta_H - 1) + (\alpha_T + \beta_T - 1)}$$

Eg. 2 Dice roll problem (6 outcomes instead of 2)

Likelihood is ~ Multinomial($\theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$)

$$P(\mathcal{D} \mid \theta) = \theta_1^{\alpha_1} \theta_2^{\alpha_2} \ldots \theta_k^{\alpha_k}$$

If prior is Dirichlet distribution,

$$P(\theta) = \frac{\theta_1^{\beta_1 - 1}\ \theta_2^{\beta_2 - 1} \ldots \theta_k^{\beta_k - 1}}{B(\beta_1, \ldots, \beta_k)} \sim \mathrm{Dirichlet}(\beta_1, \ldots, \beta_k)$$

Then posterior is Dirichlet distribution

$$P(\theta|D) \sim \mathrm{Dirichlet}(\beta_1 + \alpha_1, \ldots, \beta_k + \alpha_k)$$

and MAP estimate is therefore

$$\hat{\theta}_i^{MAP} = \frac{\alpha_i + \beta_i - 1}{\sum_{j=1}^{k}(\alpha_j + \beta_j - 1)}$$

Estimating Parameters: Y, X_i discrete-valued

Maximum likelihood estimates:

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|}$$

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_j | Y = y_k) = \frac{\#D\{X_i = x_j \wedge Y = y_k\}}{\#D\{Y = y_k\}}$$

MAP estimates (Beta, Dirichlet priors):

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\} + (\beta_k - 1)}{|D| + \sum_m (\beta_m - 1)}$$

Only difference: "imaginary" examples

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_j | Y = y_k) = \frac{\#D\{X_i = x_j \wedge Y = y_k\} + (\beta_k - 1)}{\#D\{Y = y_k\} + \sum_m (\beta_m - 1)}$$

What if we have continuous X_i ?

Eg., image classification: X_i is i[th] pixel

Gaussian Naïve Bayes (GNB): assume

$$P(X_i = x \mid Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \; e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

Sometimes assume σ_{ik}

- is independent of Y (i.e., σ_i),
- or independent of X_i (i.e., σ_k)
- or both (i.e., σ)

Gaussian Naïve Bayes Algorithm – continuous X_i (but still discrete Y)

- Train Naïve Bayes (examples)

  for each value y_k

  estimate* $\pi_k \equiv P(Y = y_k)$

  for each attribute X_i estimate

  class conditional mean μ_{ik} , variance σ_{ik}

- Classify (X^{new})

$$Y^{new} \leftarrow \arg\max_{y_k} \; P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)$$

$$Y^{new} \leftarrow \arg\max_{y_k} \; \pi_k \prod_i Normal(X_i^{new}, \mu_{ik}, \sigma_{ik})$$

* probabilities must sum to 1, so need estimate only n-1 parameters...

- Go through the sample midterm questions, specifically for bias-variance, regularization, kernel, SVM, and conditional probabilities
- Go through the homework, know how to estimate parameters in different manners
- Read the lecture notes for bias-variance, regularization, and kernel (on top of the review slides).
- Read the SVM slides, not included in this discussion

Best of Luck!