Today

- So far in the course we have adopted a modular perspective, in which the model, loss function, optimizer, and regularizer are specified separately.
- Today we begin putting together a probabilistic interpretation of our model and loss, and introduce the concept of maximum likelihood estimation.
Probabilistic modeling of data
Let’s start with a simple biased coin example.

- You flip a coin $N = 100$ times and get outcomes $\{x_1, \ldots, x_N\}$ where $x_i \in \{0, 1\}$ and $x_i = 1$ is interpreted as heads $H$.
- Suppose you had $N_H = 55$ heads and $N_T = 45$ tails. We would like to think of a model of this phenomena.
- A good model should help us answer questions such as: What is the probability it will come up heads if we flip again?
- Let’s design a model for this scenario, fit the model. We can use the fit model to predict the next outcome.
The coin is possibly loaded. So, we can assume that one coin flip outcome $x$ is a Bernoulli random variable for some currently unknown parameter $\theta \in [0, 1]$.

$$p(x = 1|\theta) = \theta \text{ and } p(x = 0|\theta) = 1 - \theta$$
or more succinctly

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

It’s sensible to assume that $\{x_1, \ldots, x_N\}$ are independent and identically distributed (i.i.d.) Bernoullis.

Thus the joint probability of the outcome $\{x_1, \ldots, x_N\}$ is

$$p(x_1, \ldots, x_N|\theta) = \prod_{i=1}^{N} \theta^{x_i} (1 - \theta)^{1-x_i}$$
We call the probability mass (or density for continuous) of the observed data the likelihood function (as a function of the parameters $\theta$):

$$L(\theta) = \prod_{i=1}^{N} \theta^{x_i} (1 - \theta)^{1-x_i}$$

We usually work with log-likelihoods:

$$\ell(\theta) = \sum_{i=1}^{N} x_i \log \theta + (1 - x_i) \log(1 - \theta)$$

How can we choose $\theta$? Good values of $\theta$ should assign high probability to the observed data. This motivates the maximum likelihood criterion, that we should pick the parameters that maximize the likelihood:

$$\hat{\theta}_{ML} = \arg \max_{\theta \in [0,1]} \ell(\theta)$$
Remember how we found the optimal solution to linear regression by setting derivatives to zero? We can do that again for the coin example.

\[
\frac{d\ell}{d\theta} = \frac{d}{d\theta} \left( \sum_{i=1}^{N} x_i \log \theta + (1 - x_i) \log(1 - \theta) \right)
\]

\[
= \frac{d}{d\theta} \left( N_H \log \theta + N_T \log(1 - \theta) \right)
\]

\[
= \frac{N_H}{\theta} - \frac{N_T}{1 - \theta}
\]

where \(N_H = \sum_i x_i\) and \(N_T = N - \sum_i x_i\).

Setting this to zero gives the maximum likelihood estimate:

\[
\hat{\theta}_{\text{ML}} = \frac{N_H}{N_H + N_T}.
\]
Notice, in the coin example we are actually minimizing cross-entropies!

\[
\hat{\theta}_{ML} = \arg \max_{\theta \in [0,1]} \ell(\theta) = \arg \min_{\theta \in [0,1]} -\ell(\theta) = \arg \min_{\theta \in [0,1]} \sum_{i=1}^{N} -x_i \log \theta - (1 - x_i) \log(1 - \theta)
\]

This is an example of maximum likelihood estimation.
- define a model that assigns a probability (or has a probability density at) to a dataset
- maximize the likelihood (or minimize the neg. log-likelihood).

Many examples we’ve considered fall in this framework! Let’s consider classification again.
Strategies for classification
Spam classification

- If you are a large company that runs an email service, one of the important predictive problems you may have is the automated detection of spam email.
Discriminative Classifiers

- Given inputs $x$ and classes $y$ we can do classification in several ways. How?

- **Discriminative** classifiers try to either:
  - learn mappings directly from the space of inputs $\mathcal{X}$ to class labels $\{0, 1, 2, \ldots, K\}$
Generative Classifiers

How about this approach: build a model of “what data for a class looks like”

- **Generative** classifiers try to model $p(x, y)$. If we know $p(y)$ we can easily compute $p(x|y)$.
- Classification via Bayes rule (thus also called Bayes classifiers)

$$p(x|y)$$

<table>
<thead>
<tr>
<th>Probability of feature given label</th>
<th>Class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>postpone, board, meeting, Thanksgiving</td>
<td>Not spam</td>
</tr>
<tr>
<td>mining, Bitcoin, contact, opportunity</td>
<td>Spam</td>
</tr>
</tbody>
</table>
Generative vs Discriminative

Two approaches to classification:

- **Discriminative approach**: estimate parameters of decision boundary/class separator directly from labeled examples.
  - Model $p(t|x)$ directly (logistic regression models)
  - Learn mappings from inputs to classes (linear/logistic regression, decision trees etc)
  - Tries to solve: How do I separate the classes?

- **Generative approach**: model the distribution of inputs characteristic of the class (Bayes classifier).
  - Model $p(x|t)$
  - Apply Bayes Rule to derive $p(t|x)$.
  - Tries to solve: What does each class ”look” like?

- Key difference: is there a distributional assumption over inputs?
Naïve Bayes
A Generative Model: Bayes Classifier

- Aim to classify text into spam/not-spam (yes $c=1$; no $c=0$)
- Example: “You are one of the very few who have been selected as a winners for the free $1000$ Gift Card."
- Use bag-of-words features, get binary vector $\mathbf{x}$ for each email

Vocabulary:
- “a”: 1
- ...
- “car”: 0
- “card”: 1
- ...
- “win”: 0
- “winner”: 1
- “winter”: 0
- ...
- “you”: 1
Bayes Classifier

- Given features \( \mathbf{x} = [x_1, x_2, \cdots, x_D]^T \) we want to compute class probabilities using Bayes Rule:

\[
\begin{align*}
\Pr(\text{class given words}) &= \frac{\Pr(\mathbf{x}, \text{class})}{\Pr(\mathbf{x})} = \frac{\Pr(\mathbf{x} | \text{class}) \Pr(\text{class})}{\Pr(\mathbf{x})}
\end{align*}
\]

- More formally

\[
\text{posterior} = \frac{\text{Class likelihood} \times \text{prior}}{\text{Evidence}}
\]

- How can we compute \( \Pr(\mathbf{x}) \) for the two class case? (Do we need to?)

\[
\Pr(\mathbf{x}) = \Pr(\mathbf{x} | c = 0) \Pr(c = 0) + \Pr(\mathbf{x} | c = 1) \Pr(c = 1)
\]

- To compute \( \Pr(c | \mathbf{x}) \) we need: \( \Pr(\mathbf{x} | c) \) and \( \Pr(c) \)
Naïve Bayes

- Assume we have two classes: spam and non-spam. We have a dictionary of $D$ words, and binary features $\mathbf{x} = [x_1, \ldots, x_D]$ saying whether each word appears in the e-mail.

- If we define a joint distribution $p(c, x_1, \ldots, x_D)$, this gives enough information to determine $p(c)$ and $p(\mathbf{x}|c)$.

- Problem: specifying a joint distribution over $D + 1$ binary variables requires $2^{D+1} - 1$ entries. This is computationally prohibitive and would require an absurd amount of data to fit.

- We’d like to impose structure on the distribution such that:
  - it can be compactly represented
  - learning and inference are both tractable
Naïve Bayes

- Naïve assumption: Naïve Bayes assumes that the word features $x_i$ are conditionally independent given the class $c$.
  - This means $x_i$ and $x_j$ are independent under the conditional distribution $p(x|c)$.
  - Note: this doesn’t mean they’re independent.
  - Mathematically,

  $$p(c, x_1, \ldots, x_D) = p(c)p(x_1|c) \cdots p(x_D|c).$$

- Compact representation of the joint distribution
  - Prior probability of class: $p(c = 1) = \pi$ (e.g. spam email)
  - Conditional probability of word feature given class:
    $$p(x_j = 1|c) = \theta_{jc}$$ (e.g. word “price” appearing in spam)
  - $2D + 1$ parameters total (before $2^{D+1} - 1$)
Bayes Nets

- We can represent this model using an directed graphical model, or Bayesian network:

![Bayesian Network Diagram]

- This graph structure means the joint distribution factorizes as a product of conditional distributions for each variable given its parent(s).

- Intuitively, you can think of the edges as reflecting a causal structure. But mathematically, this doesn’t hold without additional assumptions.
Naïve Bayes: Learning

- The parameters can be learned efficiently because the log-likelihood decomposes into independent terms for each feature.

\[
\ell(\theta) = \sum_{i=1}^{N} \log p(c^{(i)}, x^{(i)}) = \sum_{i=1}^{N} \log \left\{ p(x^{(i)}|c^{(i)})p(c^{(i)}) \right\}
\]

\[
= \sum_{i=1}^{N} \log \left\{ p(c^{(i)}) \prod_{j=1}^{D} p(x^{(i)}_j | c^{(i)}) \right\}
\]

\[
= \sum_{i=1}^{N} \left[ \log p(c^{(i)}) + \sum_{j=1}^{D} \log p(x^{(i)}_j | c^{(i)}) \right]
\]

\[
= \sum_{i=1}^{N} \log p(c^{(i)}) + \sum_{j=1}^{D} \sum_{i=1}^{N} \log p(x^{(i)}_j | c^{(i)})
\]

- Each of these log-likelihood terms depends on different sets of parameters, so they can be optimized independently.
We can handle these terms separately. For the prior we maximize:
\[ \sum_{i=1}^{N} \log p(c^{(i)}) \]

This is a minor variant of our coin flip example. Let \( p(c^{(i)} = 1) = \pi \).
Note \( p(c^{(i)}) = \pi^{c^{(i)}} (1 - \pi)^{1-c^{(i)}} \).

Log-likelihood:
\[
\sum_{i=1}^{N} \log p(c^{(i)}) = \sum_{i=1}^{N} c^{(i)} \log \pi + \sum_{i=1}^{N} (1 - c^{(i)}) \log (1 - \pi)
\]

Obtain MLEs by setting derivatives to zero:
\[
\hat{\pi} = \frac{\sum_{i} \mathbb{I}[c^{(i)} = 1]}{N} = \frac{\# \text{ spams in dataset}}{\text{total \# samples}}
\]
Naïve Bayes: Learning

- Each $\theta_{jc}$'s can be treated separately: maximize $\sum_{i=1}^{N} \log p(x_j^{(i)} | c^{(i)})$
- This is (again) a minor variant of our coin flip example.
  
  Let $\theta_{jc} = p(x_j^{(i)} = 1 | c)$. Note $p(x_j^{(i)} | c) = \theta_{jc} x_j^{(i)} (1 - \theta_{jc})^{1-x_j^{(i)}}$.
- Log-likelihood:
  
  $$\sum_{i=1}^{N} \log p(x_j^{(i)} | c^{(i)}) = \sum_{i=1}^{N} c^{(i)} \left\{ x_j^{(i)} \log \theta_{j1} + (1 - x_j^{(i)}) \log(1 - \theta_{j1}) \right\}$$
  
  $$+ \sum_{i=1}^{N} (1 - c^{(i)}) \left\{ x_j^{(i)} \log \theta_{j0} + (1 - x_j^{(i)}) \log(1 - \theta_{j0}) \right\}$$

- Obtain MLEs by setting derivatives to zero:
  
  $$\hat{\theta}_{jc} = \frac{\sum_{i} \mathbb{I}[x_j^{(i)} = 1 \& c^{(i)} = c]}{\sum_{i} \mathbb{I}[c^{(i)} = c]} \quad \text{for } c = 1 \quad \# \text{word } j \text{ appears in spams}$$
  
  $$\text{\# spams in dataset}$$
Naïve Bayes: Inference

- We predict the category by performing inference in the model.

- Apply Bayes’ Rule:

\[ p(c | x) = \frac{p(c)p(x | c)}{\sum_{c'} p(c')p(x | c')} = \frac{p(c) \prod_{j=1}^{D} p(x_j | c)}{\sum_{c'} p(c') \prod_{j=1}^{D} p(x_j | c')} \]

- We need not compute the denominator if we’re simply trying to determine the most likely \( c \).

- Shorthand notation:

\[ p(c | x) \propto p(c) \prod_{j=1}^{D} p(x_j | c) \]

- For input \( x \), predict by comparing the values of \( p(c) \prod_{j=1}^{D} p(x_j | c) \) for different \( c \) (e.g. choose the largest).
Naïve Bayes

- Naïve Bayes is an amazingly cheap learning algorithm!

- **Training time:** estimate parameters using maximum likelihood
  - Compute co-occurrence counts of each feature with the labels.
  - Requires only one pass through the data!

- **Test time:** apply Bayes’ Rule
  - Cheap because of the model structure. (For more general models, Bayesian inference can be very expensive and/or complicated.)

- We covered the Bernoulli case for simplicity. But our analysis easily extends to other probability distributions.

- Unfortunately, it’s usually less accurate in practice compared to discriminative models due to its “naïve” independence assumption.
Bayesian Parameter Estimation
MLE issue: Data Sparsity

- Maximum likelihood has a pitfall: if you have too little data, it can overfit.

- E.g., what if you flip the coin twice and get H both times?

\[ \theta_{ML} = \frac{N_H}{N_H + N_T} = \frac{2}{2 + 0} = 1 \]

- Because it never observed T, it assigns this outcome probability 0. This problem is known as data sparsity.
Bayesian Parameter Estimation

- In maximum likelihood, the observations are treated as random variables, but the parameters are not.

- The **Bayesian** approach treats the parameters as random variables as well. $\beta$ is the set of parameters in the prior distribution of $\theta$.

- To define a Bayesian model, we need to specify two distributions:
  - The prior distribution $p(\theta)$, which encodes our beliefs about the parameters *before* we observe the data
  - The likelihood $p(D \mid \theta)$, same as in maximum likelihood
When we update our beliefs based on the observations, we compute the posterior distribution using Bayes’ Rule:

$$p(\theta | D) = \frac{p(\theta)p(D | \theta)}{\int p(\theta')p(D | \theta') d\theta'}.$$ 

We rarely ever compute the denominator explicitly. In general, it is computationally intractable.
Bayesian Parameter Estimation

- Let’s revisit the coin example. We already know the likelihood:

\[ L(\theta) = p(D|\theta) = \theta^N_H (1 - \theta)^N_T \]

- It remains to specify the prior \( p(\theta) \).
  - We can choose an **uninformative prior**, which assumes as little as possible. A reasonable choice is the uniform prior.
  - But our experience tells us 0.5 is more likely than 0.99. One particularly useful prior that lets us specify this is the **beta distribution**:

\[
p(\theta; a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)} \theta^{a-1}(1 - \theta)^{b-1}.
\]

- This notation for proportionality lets us ignore the normalization constant:

\[
p(\theta; a, b) \propto \theta^{a-1}(1 - \theta)^{b-1}.
\]
Bayesian Parameter Estimation

- Beta distribution for various values of $a$, $b$:

- Some observations:
  - The expectation $\mathbb{E}[\theta] = a/(a + b)$
  - The distribution gets more peaked when $a$ and $b$ are large.
  - The uniform distribution is the special case where $a = b = 1$.

- The beta distribution is used for is as a prior for the Bernoulli distribution.
Bayesian Parameter Estimation

- Computing the posterior distribution:

\[
p(\theta | D) \propto p(\theta)p(D | \theta)
\]

\[
\propto \left[ \theta^{a-1}(1 - \theta)^{b-1} \right] \left[ \theta^{N_H}(1 - \theta)^{N_T} \right]
\]

\[
= \theta^{a-1+N_H}(1 - \theta)^{b-1+N_T}.
\]

- This is just a beta distribution with parameters \( N_H + a \) and \( N_T + b \).
- The posterior expectation of \( \theta \) is:

\[
E[\theta | D] = \frac{N_H + a}{N_H + N_T + a + b}
\]

- The parameters \( a \) and \( b \) of the prior can be thought of as pseudo-counts.
  - The reason this works is that the prior and likelihood have the same functional form. This phenomenon is known as conjugacy (conjugate priors), and it’s very useful.
Bayesian Parameter Estimation

Bayesian inference for the coin flip example:

**Small data setting**

\[ N_H = 2, \quad N_T = 0 \]

**Large data setting**

\[ N_H = 55, \quad N_T = 45 \]

When you have enough observations, the data overwhelm the prior.
Maximum a-posteriori (MAP) estimation: find the most likely parameter settings under the posterior
This converts the Bayesian parameter estimation problem into a maximization problem

\[
\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} \ p(\theta \mid \mathcal{D}) \\
= \arg \max_{\theta} \ p(\theta, \mathcal{D}) \\
= \arg \max_{\theta} \ p(\theta) \ p(\mathcal{D} \mid \theta) \\
= \arg \max_{\theta} \ \log p(\theta) + \log p(\mathcal{D} \mid \theta)
\]
Joint probability in the coin flip example:

\[
\log p(\theta, D) = \log p(\theta) + \log p(D | \theta) \\
= \text{Const} + (a - 1) \log \theta + (b - 1) \log(1 - \theta) + N_H \log \theta + N_T \log(1 - \theta) \\
= \text{Const} + (N_H + a - 1) \log \theta + (N_T + b - 1) \log(1 - \theta)
\]

Maximize by finding a critical point

\[
0 = \frac{d}{d\theta} \log p(\theta, D) = \frac{N_H + a - 1}{\theta} - \frac{N_T + b - 1}{1 - \theta}
\]

Solving for \( \theta \),

\[
\hat{\theta}_{\text{MAP}} = \frac{N_H + a - 1}{N_H + N_T + a + b - 2}
\]
Comparison of estimates in the coin flip example:

<table>
<thead>
<tr>
<th>Formula</th>
<th>( N_H = 2, N_T = 0 )</th>
<th>( N_H = 55, N_T = 45 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{\theta}_{ML} )</td>
<td>( \frac{N_H}{N_H + N_T} )</td>
<td>( 1 )</td>
</tr>
<tr>
<td>( \mathbb{E}[\theta</td>
<td>D] )</td>
<td>( \frac{N_H + a}{N_H + N_T + a + b} )</td>
</tr>
<tr>
<td>( \hat{\theta}_{MAP} )</td>
<td>( \frac{N_H + a - 1}{N_H + N_T + a + b - 2} )</td>
<td>( \frac{3}{4} = 0.75 )</td>
</tr>
</tbody>
</table>

\( \hat{\theta}_{MAP} \) assigns nonzero probabilities as long as \( a, b > 1 \).
Multivariate Gaussian Distribution
Classification: Diabetes Example

- Observation per patient: White blood cell count & glucose value.

\[ p(\mathbf{x} \mid t = k) \] for each class is shaped like an ellipse

\[ \implies \text{we model each class as a multivariate Gaussian} \]
Recall the Gaussian, or normal, distribution:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi \sigma}} \exp \left( -\frac{(x - \mu)^2}{2\sigma^2} \right)$$

Parameterized by mean $\mu$ and variance $\sigma^2$.

The Central Limit Theorem says that sums of lots of independent random variables are approximately Gaussian.

In machine learning, we use Gaussians a lot because they make the calculations easy.
Multivariate Mean and Covariance

- Mean

\[ \mu = \mathbb{E}[x] = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_d \end{pmatrix} \]

- Covariance

\[ \Sigma = \text{Cov}(x) = \mathbb{E}[(x - \mu)(x - \mu)^\top] = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{D1} & \sigma_{D2} & \cdots & \sigma_D^2 \end{pmatrix} \]

- The statistics (\(\mu\) and \(\Sigma\)) uniquely define a multivariate Gaussian (or multivariate Normal) distribution, denoted \(\mathcal{N}(\mu, \Sigma)\) or \(\mathcal{N}(x; \mu, \Sigma)\).
  - This is not true for distributions in general!
Multivariate Gaussian Distribution

- PDF of the multivariate Gaussian distribution:

\[ 
\mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right] 
\]

- Compare to the univariate case \((d = 1, \Sigma = \sigma^2):\)

\[ 
\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) 
\]
Gaussian Intuition: (Multivariate) Shift + Scale

- Recall that in the univariate case, all normal distributions are shaped like the standard normal distribution.
- The densities are related to the standard normal by a shift ($\mu$), a scale (or stretch, or dilation) $\sigma$, and a normalization factor.
The same intuition applies in the multivariate case.

We can think of the multivariate Gaussian as a shifted and “scaled” version of the standard multivariate normal distribution.

- The standard multivariate normal has $\mu = 0$ and $\Sigma = I$

Multivariate analog of the shift is simple: it’s a vector $\mu$

But what about the scale?

- In the univariate case, the scale factor was the square root of the variance: $\sigma = \sqrt{\sigma^2}$
- But in the multivariate case, the covariance $\Sigma$ is a matrix! Does $\Sigma^{\frac{1}{2}}$ exist, and can we scale by it?
We call a matrix “positive definite” if it scales the space in orthogonal directions. The univariate analog is positive scalar $\alpha > 0$. Consider, e.g., how these two matrices transform the orthogonal vectors:

Consider matrix:

$$
\begin{pmatrix}
2 & 0 \\
0 & 0.5 \\
\end{pmatrix}
\begin{pmatrix}
1 & 0.5 \\
0.5 & 1 \\
\end{pmatrix}
$$

Consider action on:

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

Notice: both matrices are symmetric!
Multivariate Scaling (Formal) (details optional)

We summarize some definitions/results from linear algebra (without proof). Knowing them is optional, but they may help with intuition (esp. for PCA).

- **Definition.** Symmetric matrix $A$ is **positive semidefinite** if $x^\top A x \geq 0$ for all non-zero $x$. It is **positive definite** if $x^\top A x > 0$ for all non-zero $x$.
  - Any positive definite matrix is positive semidefinite.
  - Positive definite matrices have positive eigenvalues, and positive semidefinite matrices have non-negative eigenvalues.
  - For any matrix $X$, $X^\top X$ and $XX^\top$ are positive semidefinite.

- **Theorem (Unique Positive Square Root).** Let $A$ be a positive semidefinite real matrix. Then there is a unique positive semidefinite matrix $B$ such that $A = B^\top B = BB$. We call $A^{\frac{1}{2}} \triangleq B$ the **positive square root** of $A$.

- **Theorem (Spectral Theorem).** The following are equivalent for $A \in \mathbb{R}^{d \times d}$:
  1. $A$ is symmetric.
  2. $\mathbb{R}^D$ has an orthonormal basis consisting of the eigenvectors of $A$.
  3. There exists orthogonal matrix $Q$ and diagonal matrix $\Lambda$ such that $A = Q \Lambda Q^T$. This is called the **spectral decomposition** of $A$.
    - The columns of $Q$ are (unit) eigenvectors of $A$. 