

CSC 411: Introduction to Machine Learning

Lecture 13: Probabilistic Models I

Mengye Ren and Matthew MacKay

University of Toronto

- We'll shift directions now, and spend most of the next 4 weeks talking about probabilistic models.
- This lecture
 - Maximum likelihood estimation
 - Naïve Bayes

- Motivating example: estimating the parameter of a biased coin
 - You flip a coin 100 times. It lands heads $N_H = 55$ times and tails $N_T = 45$ times.
 - What is the probability it will come up heads if we flip again?
- Model: flips are independent Bernoulli random variables with parameter θ .
 - Assume the observations are **independent and identically distributed (i.i.d.)**

Maximum Likelihood

- The **likelihood function** is the probability of the observed data, as a function of θ .
- In our case, it's the probability of a *particular* sequence of H's and T's.
- Under the Bernoulli model with i.i.d. observations,

$$L(\theta) = p(\mathcal{D}) = \theta^{N_H} (1 - \theta)^{N_T}$$

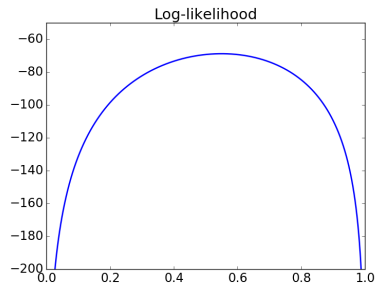
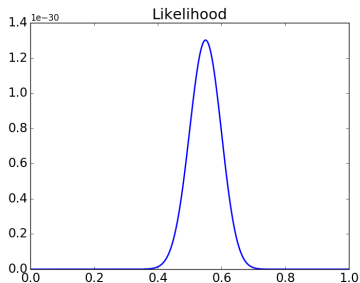
- This takes very small values. In this case,
 $L(0.5) = 0.5^{100} \approx 7.9 \times 10^{-31}$
- Therefore, we usually work with log-likelihoods:

$$\ell(\theta) = \log L(\theta) = N_H \log \theta + N_T \log(1 - \theta)$$

- Here, $\ell(0.5) = \log 0.5^{100} = 100 \log 0.5 = -69.31$

Maximum Likelihood

$$N_H = 55, N_T = 45$$



Maximum Likelihood

- Good values of θ should assign high probability to the observed data. This motivates the **maximum likelihood criterion**.
- Remember how we found the optimal solution to linear regression by setting derivatives to zero? We can do that again for the coin example.

$$\begin{aligned}\frac{d\ell}{d\theta} &= \frac{d}{d\theta} (N_H \log \theta + N_T \log(1 - \theta)) \\ &= \frac{N_H}{\theta} - \frac{N_T}{1 - \theta}\end{aligned}$$

- Setting this to zero gives the maximum likelihood estimate:

$$\hat{\theta}_{\text{ML}} = \frac{N_H}{N_H + N_T},$$

- This is equivalent to minimizing cross-entropy. Let $t_i = 1$ for heads and $t_i = 0$ for tails.

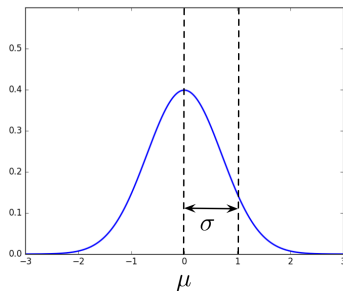
$$\begin{aligned}\mathcal{L}_{CE} &= - \sum_i t_i \log \theta - (1 - t_i) \log(1 - \theta) \\ &= -N_H \log \theta - N_T \log(1 - \theta) \\ &= -\ell(\theta)\end{aligned}$$

Maximum Likelihood

- Recall the **Gaussian**, or **normal**, distribution:

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

- 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.



Maximum Likelihood

- Suppose we want to model the distribution of temperatures in Toronto in March, and we've recorded the following observations:

-2.5 -9.9 -12.1 -8.9 -6.0 -4.8 2.4

- Assume they're drawn from a Gaussian distribution with known standard deviation $\sigma = 5$, and we want to find the mean μ .
- Log-likelihood function:

$$\begin{aligned}\ell(\mu) &= \log \prod_{i=1}^N \left[\frac{1}{\sqrt{2\pi} \cdot \sigma} \exp \left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2} \right) \right] \\ &= \sum_{i=1}^N \log \left[\frac{1}{\sqrt{2\pi} \cdot \sigma} \exp \left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2} \right) \right] \\ &= \sum_{i=1}^N \underbrace{-\frac{1}{2} \log 2\pi - \log \sigma}_{\text{constant}} - \frac{(x^{(i)} - \mu)^2}{2\sigma^2}\end{aligned}$$

- Maximize the log-likelihood by setting the derivative to zero:

$$\begin{aligned} 0 &= \frac{d\ell}{d\mu} = -\frac{1}{2\sigma^2} \sum_{i=1}^N \frac{d}{d\mu} (x^{(i)} - \mu)^2 \\ &= \frac{1}{\sigma^2} \sum_{i=1}^N x^{(i)} - \mu \end{aligned}$$

- Solving we get $\mu = \frac{1}{N} \sum_{i=1}^N x^{(i)}$
- This is just the mean of the observed values, or the **empirical mean**.

Maximum Likelihood

- In general, we don't know the true standard deviation σ , but we can solve for it as well.
- Set the *partial* derivatives to zero, just like in linear regression.

$$0 = \frac{\partial \ell}{\partial \mu} = -\frac{1}{\sigma^2} \sum_{i=1}^N x^{(i)} - \mu$$

$$\begin{aligned} 0 = \frac{\partial \ell}{\partial \sigma} &= \frac{\partial}{\partial \sigma} \left[\sum_{i=1}^N -\frac{1}{2} \log 2\pi - \log \sigma - \frac{1}{2\sigma^2} (x^{(i)} - \mu)^2 \right] \\ &= \sum_{i=1}^N -\frac{1}{2} \frac{\partial}{\partial \sigma} \log 2\pi - \frac{\partial}{\partial \sigma} \log \sigma - \frac{\partial}{\partial \sigma} \frac{1}{2\sigma} (x^{(i)} - \mu)^2 \\ &= \sum_{i=1}^N 0 - \frac{1}{\sigma} + \frac{1}{\sigma^3} (x^{(i)} - \mu)^2 \\ &= -\frac{N}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^N (x^{(i)} - \mu)^2 \end{aligned}$$

$$\hat{\mu}_{\text{ML}} = \frac{1}{N} \sum_{i=1}^N x^{(i)}$$

$$\hat{\sigma}_{\text{ML}} = \sqrt{\frac{1}{N} \sum_{i=1}^N (x^{(i)} - \mu)^2}$$

- Sometimes there is no closed-form solution. E.g., consider the gamma distribution, whose PDF is

$$p(x) = \frac{b^a}{\Gamma(a)} x^{a-1} e^{-bx},$$

where Γ is the gamma function, a generalization of the factorial function to continuous values.

- There is no closed-form solution, but we can still optimize the log-likelihood using gradient ascent.

Maximum Likelihood

- So far, maximum likelihood has told us to use empirical counts or statistics:
 - **Bernoulli:** $\theta = \frac{N_H}{N_H + N_T}$
 - **Gaussian:** $\mu = \frac{1}{N} \sum x^{(i)}$, $\sigma^2 = \frac{1}{N} \sum (x^{(i)} - \mu)^2$
- This doesn't always happen; the class of probability distributions that have this property is **exponential families**.

$$f_X(x|\theta) = h(x) \exp(\eta(\theta)^\top T(x) - A(\theta))$$

- θ : distribution parameters
- $\eta(\theta)$: natural parameter
- $T(x)$: sufficient statistic

We've been doing maximum likelihood estimation all along!

- Squared error loss (e.g. linear regression)

$$p(t|y) = \mathcal{N}(t; y, \sigma^2)$$

$$-\log p(t|y) = \frac{1}{2\sigma^2}(y - t)^2 + \text{const}$$

- Cross-entropy loss (e.g. logistic regression)

$$p(t = 1|y) = \text{Bernoulli}(t; y)$$

$$-\log p(t|y) = -t \log y - (1 - t) \log(1 - y)$$

Generative vs Discriminative

Two approaches to classification:

- **Discriminative approach:** estimate parameters of decision boundary/class separator directly from labeled examples.
 - Tries to solve: How do I separate the classes?
 - learn $p(y|\mathbf{x})$ directly (logistic regression models)
 - learn mappings from inputs to classes (least-squares, decision trees)
- **Generative approach:** model the distribution of inputs characteristic of the class (Bayes classifier).
 - Tries to solve: What does each class "look" like?
 - Build a model of $p(\mathbf{x}|y)$
 - Apply Bayes Rule

Bayes Classifier

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

Bayes Classifier

- Given features $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$ we want to compute class probabilities using Bayes Rule:

$$\underbrace{p(c|\mathbf{x})}_{\text{Pr. class given words}} = \frac{p(\mathbf{x}, c)}{p(\mathbf{x})} = \frac{\overbrace{p(\mathbf{x}|c)}^{\text{Pr. words given class}} p(c)}{p(\mathbf{x})}$$

- More formally

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

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

$$p(\mathbf{x}) = p(\mathbf{x}|c = 0)p(c = 0) + p(\mathbf{x}|c = 1)p(c = 1)$$

- To compute $p(c|\mathbf{x})$ we need: $p(\mathbf{x}|c)$ and $p(c)$

- Assume we have two classes: spam and non-spam. We have a dictionary of D words, and binary features $\mathbf{x} = [x_1, \dots, x_D]$ saying whether each word appears in the e-mail.
- If we define a joint distribution $p(c, x_1, \dots, x_D)$, 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} entries. This is computationally prohibitive and would require an absurd amount of data to fit.

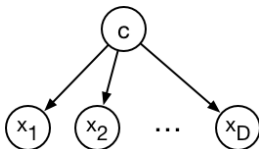
- We'd like to impose **structure** on the distribution such that:
 - it can be **compactly** represented
 - **learning** and **inference** are both tractable
- **Probabilistic graphical models** are a powerful and wide-ranging class of techniques for doing this. We'll just scratch the surface here.

- **Naïve Bayes** makes the assumption 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 $p(\mathbf{x}|c)$.
 - Note: this doesn't mean they're independent. (E.g., "gift" and "card" are correlated insofar as they both depend on c .)
 - Mathematically,

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

- Compact representation of the joint distribution
 - Prior probability of class: $p(c = 1) = \theta_C$
 - Conditional probability of word feature given class: $p(x_j = 1|c) = \theta_{jc}$
 - $2D + 1$ parameters total

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



- 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.

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \sum_{i=1}^N \log p(c^{(i)}, \mathbf{x}^{(i)}) \\ &= \sum_{i=1}^N \log p(c^{(i)}) \prod_{j=1}^D p(x_j^{(i)} | c^{(i)}) \\ &= \sum_{i=1}^N \left[\log p(c^{(i)}) + \sum_{j=1}^D \log p(x_j^{(i)} | c^{(i)}) \right] \\ &= \underbrace{\sum_{i=1}^N \log p(c^{(i)})}_{\text{Bernoulli log-likelihood of labels}} + \sum_{j=1}^D \underbrace{\sum_{i=1}^N \log p(x_j^{(i)} | c^{(i)})}_{\text{Bernoulli log-likelihood for feature } x_j}\end{aligned}$$

- Each of these log-likelihood terms depends on different sets of parameters, so they can be optimized independently.

Naïve Bayes: Learning

- Want to maximize $\sum_{i=1}^N \log p(x_j^{(i)} | c^{(i)})$
- This is a minor variant of our coin flip example. Let $\theta_{ab} = p(x_j = a | c = b)$. Note $\theta_{1b} = 1 - \theta_{0b}$.
- Log-likelihood:

$$\begin{aligned} \sum_{i=1}^N \log p(x_j^{(i)} | c^{(i)}) &= \sum_{i=1}^N c^{(i)} x_j^{(i)} \log \theta_{11} + \sum_{i=1}^N c^{(i)} (1 - x_j^{(i)}) \log(1 - \theta_{11}) \\ &\quad + \sum_{i=1}^N (1 - c^{(i)}) x_j^{(i)} \log \theta_{10} + \sum_{i=1}^N (1 - c^{(i)}) (1 - x_j^{(i)}) \log(1 - \theta_{10}) \end{aligned}$$

- Obtain maximum likelihood estimates by setting derivatives to zero:

$$\theta_{11} = \frac{N_{11}}{N_{11} + N_{01}} \quad \theta_{10} = \frac{N_{10}}{N_{10} + N_{00}}$$

where N_{ab} is the counts for $x_j = a$ and $c = b$.

Naïve Bayes: Inference

- We predict the category by performing **inference** in the model.
- Apply **Bayes' Rule**:

$$\begin{aligned} p(c | \mathbf{x}) &= \frac{p(c)p(\mathbf{x} | c)}{\sum_{c'} p(c')p(\mathbf{x} | c')} \\ &= \frac{p(c) \prod_{j=1}^D p(x_j | c)}{\sum_{c'} p(c') \prod_{j=1}^D p(x_j | c')} \end{aligned}$$

- We need not compute the denominator if we're simply trying to determine the mostly likely c .
- Shorthand notation:

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

- 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.