CSC 311: Introduction to Machine Learning

Lecture 7 - Probabilistic Models

Murat A. Erdogdu & Richard Zemel

University of Toronto

Recall: Maximum Likelihood (MLE)

- We have seen in the assignments that various ML algorithms can be derived using the Maximum Likelihood Estimation (MLE).
- Let's start with a simple example: estimating the parameter of a biased coin
  - You flip a coin $N = 100$ 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 density of the observed data, as a function of parameters θ.
- In our case, it's the probability of a *particular* sequence of H/T's.
- Under the Bernoulli model with i.i.d. observations:
  Let x_i be the # Hs in i-th flip (can be either 1 or 0)

$$p(x_i = 1|\theta) = \theta \;\text{ and }\; p(x_i = 0|\theta) = 1 - \theta$$

$$p(x_i|\theta) = \theta^{x_i}(1-\theta)^{1-x_i} \quad \text{where} \quad x_i \in \{0,1\}$$

  Likelihood is given as

$$\begin{aligned} L(\theta) =& p(x_1, ..., x_N|\theta) = \prod_{i=1}^{N} \theta^{x_i}(1-\theta)^{1-x_i} \\ =& \theta^{N_H}(1-\theta)^{N_T} \end{aligned}$$

  where $N_H = \sum_i x_i$ and $N_T = N - \sum_i x_i$
- We usually work with log-likelihoods:

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

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.

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

- Setting this to zero gives the maximum likelihood estimate:

$$\hat{\theta}_{\mathrm{ML}} = \frac{N_H}{N_H + N_T}.$$

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 (linear/logistic regression, decision trees etc)
- **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
- Key difference: is there a distributional assumption over inputs?

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:

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

- 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 $p(c)$

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

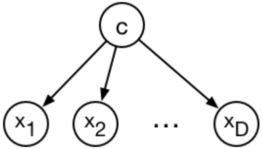
- Naïve assumption: **Naïve Bayes** assumes that the word features x_i are **conditionally independent** given the class c.
  - This means x_i and x_j are independent under the conditional distribution $p(\mathbf{x}|c)$.
  - Note: this doesn't mean they're independent.
  - Mathematically,

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

- 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**:



- 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.

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

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

Naïve Bayes: Learning

- 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:

$$\pi = \frac{\sum_i \mathbb{I}[c^{(i)} = 1]}{N} = \frac{\text{\# spams in dataset}}{\text{total \# samples}}$$

Naïve Bayes: Learning

- Each θ_{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:

$$\theta_{jc} = \frac{\sum_i \mathbb{1}[x_j^{(i)} = 1 \;\&\; c^{(i)} = c]}{\sum_i \mathbb{1}[c^{(i)} = c]} = \frac{\#\text{word } j \text{ appears in spams}}{\#\text{ spams in dataset}}$$

Naïve Bayes: Inference

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

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

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

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

- For input $\mathbf{x}$, 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.

MLE issue: Data Sparsity

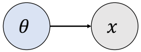
- Maximum likelihood has a pitfall: if you have too little data, it can overfit.
- E.g., what if you flip the coin twice and get H both times?

$$\theta_{\mathrm{ML}} = \frac{N_H}{N_H + N_T} = \frac{2}{2+0} = 1$$

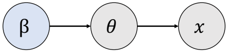
- Because it never observed T, it assigns this outcome probability 0. This problem is known as **data sparsity**.

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. β is the set of parameters in the prior distribution of θ.



- To define a Bayesian model, we need to specify two distributions:
  - The **prior distribution** $p(\boldsymbol{\theta})$, which encodes our beliefs about the parameters *before* we observe the data
  - The **likelihood** $p(\mathcal{D} \,|\, \boldsymbol{\theta})$, same as in maximum likelihood

Bayesian Parameter Estimation

- When we **update** our beliefs based on the observations, we compute the **posterior distribution** using Bayes' Rule:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\boldsymbol{\theta}) p(\mathcal{D} \mid \boldsymbol{\theta})}{\int p(\boldsymbol{\theta}') p(\mathcal{D} \mid \boldsymbol{\theta}') \, \mathrm{d}\boldsymbol{\theta}'}.$$

- 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:

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

- 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**:

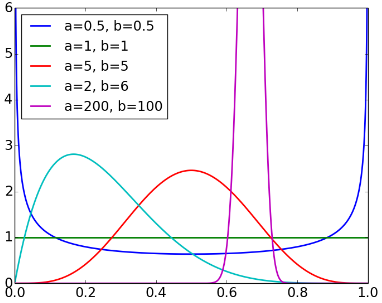
$$p(\theta; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\,\theta^{a-1}(1-\theta)^{b-1}.$$

  - 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 $\mathbb{E}[\theta] = a/(a+b)$ (easy to derive).
  - 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:

$$\begin{aligned} p(\boldsymbol{\theta} \mid \mathcal{D}) &\propto p(\boldsymbol{\theta}) p(\mathcal{D} \mid \boldsymbol{\theta}) \\ &\propto \left[\theta^{a-1}(1-\theta)^{b-1}\right]\left[\theta^{N_H}(1-\theta)^{N_T}\right] \\ &= \theta^{a-1+N_H}(1-\theta)^{b-1+N_T}. \end{aligned}$$

- This is just a beta distribution with parameters $N_H + a$ and $N_T + b$.
- The posterior expectation of θ is:

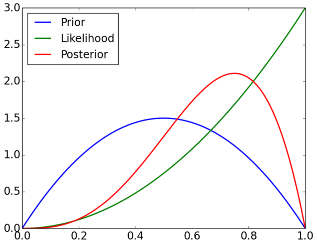
$$\mathbb{E}[\theta \mid \mathcal{D}] = \frac{N_H + a}{N_H + N_T + a + b}$$

- 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.

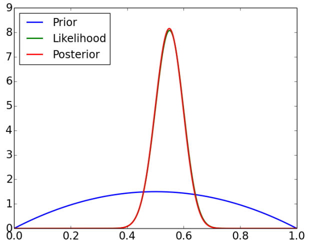
Bayesian Parameter Estimation

Bayesian inference for the coin flip example:

Small data setting
$N_H = 2,\ N_T = 0$



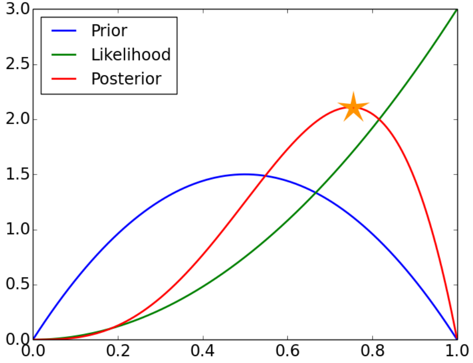
Large data setting
$N_H = 55,\ N_T = 45$



When you have enough observations, the **data overwhelm the prior**.

Maximum A-Posteriori Estimation

- **Maximum a-posteriori (MAP) estimation**: find the most likely parameter settings under the posterior



Maximum A-Posteriori Estimation

- This converts the Bayesian parameter estimation problem into a maximization problem

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta} \,|\, \mathcal{D}) \\
&= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta}, \mathcal{D}) \\
&= \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta}) \, p(\mathcal{D} \,|\, \boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \; \log p(\boldsymbol{\theta}) + \log p(\mathcal{D} \,|\, \boldsymbol{\theta})
\end{aligned}$$

- We already saw an example of this in the homework.

Maximum A-Posteriori Estimation

- Joint probability in the coin flip example:

$$\begin{aligned}\log p(\theta, \mathcal{D}) &= \log p(\theta) + \log p(\mathcal{D} \mid \theta) \\ &= \text{Const} + (a-1)\log\theta + (b-1)\log(1-\theta) + N_H \log\theta + N_T \log(1-\theta) \\ &= \text{Const} + (N_H + a - 1)\log\theta + (N_T + b - 1)\log(1-\theta)\end{aligned}$$

- Maximize by finding a critical point

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

- Solving for θ,

$$\hat{\theta}_{\text{MAP}} = \frac{N_H + a - 1}{N_H + N_T + a + b - 2}$$

Maximum A-Posteriori Estimation

Comparison of estimates in the coin flip example:

	Formula	$N_H = 2, N_T = 0$	$N_H = 55, N_T = 45$
$\hat{\theta}_{\text{ML}}$	$\frac{N_H}{N_H+N_T}$	1	$\frac{55}{100} = 0.55$
$\mathbb{E}[\theta \mid \mathcal{D}]$	$\frac{N_H+a}{N_H+N_T+a+b}$	$\frac{4}{6} \approx 0.67$	$\frac{57}{104} \approx 0.548$
$\hat{\theta}_{\text{MAP}}$	$\frac{N_H+a-1}{N_H+N_T+a+b-2}$	$\frac{3}{4} = 0.75$	$\frac{56}{102} \approx 0.549$

$\hat{\theta}_{\text{MAP}}$ assigns nonzero probabilities as long as $a, b > 1$.

Gaussian Discriminant Analysis

- Generative models - data generating distribution $p(\mathbf{x}|t = k)$
- Instead of trying to separate classes, try to model what each class "looks like".
- Recall that $p(\mathbf{x}|t = k)$ may be very complex

$$p(x_1, \cdots, x_d, y) = p(x_1|x_2, \cdots, x_d, y) \cdots p(x_{d-1}|x_d, y)p(x_d, y)$$

- Naive bayes used a conditional independence assumption. What else could we do? Choose a simple distribution.
- Next, we will discuss fitting Gaussian distributions to our data.

Bayes Classifier

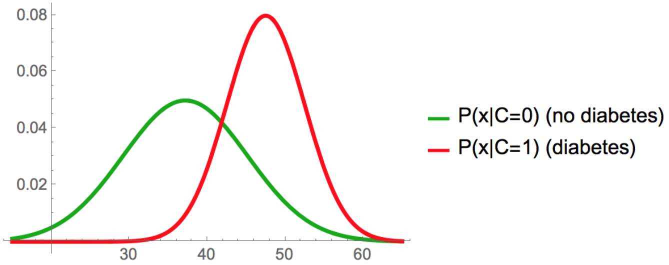
- Let's take a step back...
- Bayes Classifier

$$
\begin{aligned}
h(\mathbf{x}) = \arg\max_k p(t = k|\mathbf{x}) &= \arg\max \frac{p(\mathbf{x}|t = k)p(t = k)}{p(\mathbf{x})} \\
&= \arg\max_k p(\mathbf{x}|t = k)p(t = k)
\end{aligned}
$$

- Talked about Discrete $\mathbf{x}$, what if $\mathbf{x}$ is continuous?

Classification: Diabetes Example

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



- How can we model $p(x|t = k)$? Multivariate Gaussian

Multivariate Data

- Multiple measurements (sensors)
- D inputs/features/attributes
- N instances/observations/examples

$$\mathbf{X} = \begin{bmatrix} [\mathbf{x}^{(1)}]^\top \\ [\mathbf{x}^{(2)}]^\top \\ \vdots \\ [\mathbf{x}^{(N)}]^\top \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_D^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_D^{(N)} \end{bmatrix}$$

Multivariate Parameters

- Mean

$$\mathbb{E}[\mathbf{x}^{(i)}] = \boldsymbol{\mu} = [\mu_1, \cdots, \mu_d]^T \in \mathbb{R}^D$$

- Covariance

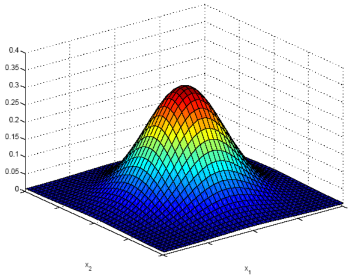
$$\boldsymbol{\Sigma} = Cov(\mathbf{x}^{(i)}) = \mathbb{E}[(\mathbf{x}^{(i)} - \boldsymbol{\mu})(\mathbf{x}^{(i)} - \boldsymbol{\mu})^\top] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{D1}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

- For Gaussians - all you need to know to represent (not true in general).

Multivariate Gaussian Distribution

- $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, a Gaussian (or normal) distribution defined as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$



- 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.

Bivariate Normal

$$\mathbf{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{\Sigma} = 0.5 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{\Sigma} = 2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

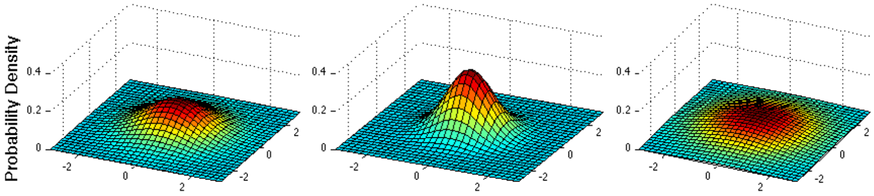


Figure: Probability density function

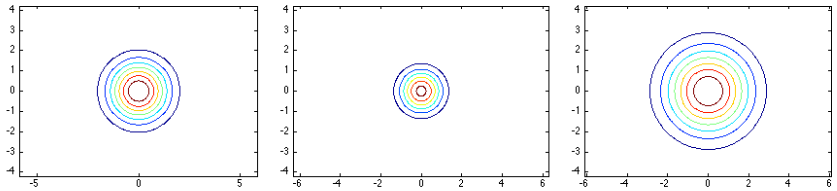


Figure: Contour plot of the pdf

Bivariate Normal

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$$

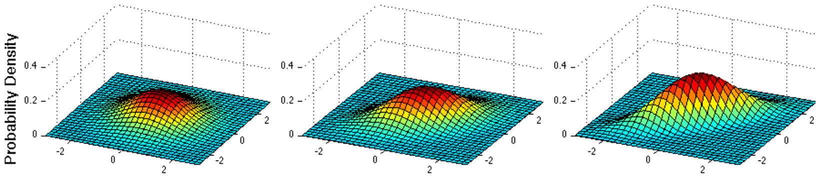


Figure: Probability density function

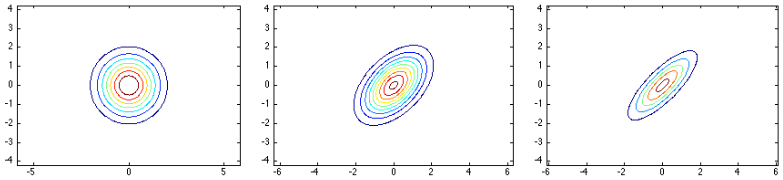


Figure: Contour plot of the pdf

Maximum Likelihood

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

  (-2.5,-7.5) (-9.9,-14.9) (-12.1,-17.5) (-8.9,-13.9) (-6.0,-11.1)
- Assume they're drawn from a Gaussian distribution with mean $\boldsymbol{\mu}$, and covariance $\boldsymbol{\Sigma}$. We want to estimate these using data.
- Log-likelihood function:

$$\begin{aligned}
\ell(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \log \prod_{i=1}^{N} \left[\frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x}^{(i)} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}) \right\} \right] \\
&= \sum_{i=1}^{N} \log \left[\frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x}^{(i)} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu}) \right\} \right] \\
&= \sum_{i=1}^{N} \underbrace{-\log (2\pi)^{d/2}}_{\text{constant}} - \log |\boldsymbol{\Sigma}|^{1/2} - \frac{1}{2} (\mathbf{x}^{(i)} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}^{(i)} - \boldsymbol{\mu})
\end{aligned}$$

Maximum Likelihood

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

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

- Solving we get $\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^{(i)}$. In general, “hat” means estimator
- This is just the sample mean of the observed values, or the **empirical mean**.

Maximum Likelihood

- Similar calculation for the covariance matrix Σ yields:
- Set the *partial* derivatives to zero, just like before

$$0 = \frac{\partial \ell}{\partial \mathbf{\Sigma}} \implies \hat{\mathbf{\Sigma}} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^\top$$

- This is called the empirical covariance and comes up quite often (i.e. PCA next lecture)
- Derivation in multivariate case is tedious. No need to worry about it. But it is good practice to derive this in one dimension. See appendix.

Gaussian Discriminant Analysis (Gaussian Bayes Classifier)

- Gaussian Discriminant Analysis in its general form assumes that $p(\mathbf{x}|t)$ is distributed according to a multivariate normal (Gaussian) distribution
- Multivariate Gaussian distribution:

$$p(\mathbf{x}|t=k) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]$$

  where $|\boldsymbol{\Sigma}_k|$ denotes the determinant of the matrix, and D is dimension of $\mathbf{x}$
- Each class k has a mean vector $\boldsymbol{\mu}_k$ and a covariance matrix $\boldsymbol{\Sigma}_k$
- $\boldsymbol{\Sigma}_k$ has $\mathcal{O}(D^2)$ parameters - could be hard to estimate

Gaussian Discriminant Analysis (Gaussian Bayes Classifier)

- GDA (GBC) decision boundary is based on class posterior.
- Make decisions by comparing class probabilities:

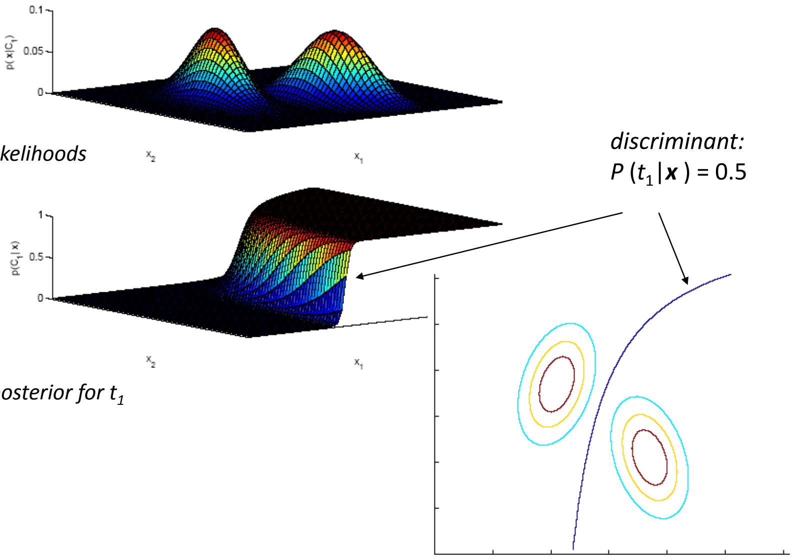
$$
\begin{aligned}
\log p(t_k|\mathbf{x}) &= \log p(\mathbf{x}|t_k) + \log p(t_k) - \log p(\mathbf{x}) \\
&= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k^{-1}| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \\
&\quad + \log p(t_k) - \log p(\mathbf{x})
\end{aligned}
$$

- Decision boundary ($\log p(t_k|\mathbf{x}) = \log p(t_l|\mathbf{x})$):

$$
\begin{aligned}
(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) &= (\mathbf{x}-\boldsymbol{\mu}_\ell)^T\boldsymbol{\Sigma}_\ell^{-1}(\mathbf{x}-\boldsymbol{\mu}_\ell) + C_{k,l} \\
\mathbf{x}^T\boldsymbol{\Sigma}_k^{-1}\mathbf{x} - 2\boldsymbol{\mu}_k^T\boldsymbol{\Sigma}_k^{-1}\mathbf{x} &= \mathbf{x}^T\boldsymbol{\Sigma}_\ell^{-1}\mathbf{x} - 2\boldsymbol{\mu}_\ell^T\boldsymbol{\Sigma}_\ell^{-1}\mathbf{x} + C_{k,l}
\end{aligned}
$$

- Quadratic function in $\mathbf{x}$ $\Longrightarrow$ quadratic decision boundary
- What is $C_{k,l}$? What if $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_\ell$?

Decision Boundary



Learning

- Learn the parameters for each class using maximum likelihood
- Assume the prior is Bernoulli (we have two classes)

$$p(t|\phi) = \phi^t (1-\phi)^{1-t}.$$

- You can compute the MLE in closed form (good exercise!)

$$\begin{aligned}
\phi &= \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}[t^{(n)} = 1] \\
\boldsymbol{\mu}_k &= \frac{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = k] \cdot \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = k]} \\
\boldsymbol{\Sigma}_k &= \frac{1}{\sum_{n=1}^{N} \mathbb{1}[t^{(n)} = k]} \sum_{n=1}^{N} \mathbb{1}[t^{(n)} = k](\mathbf{x}^{(n)} - \mu_{t^{(n)}})(\mathbf{x}^{(n)} - \mu_{t^{(n)}})^T
\end{aligned}$$

Simplifying the Model

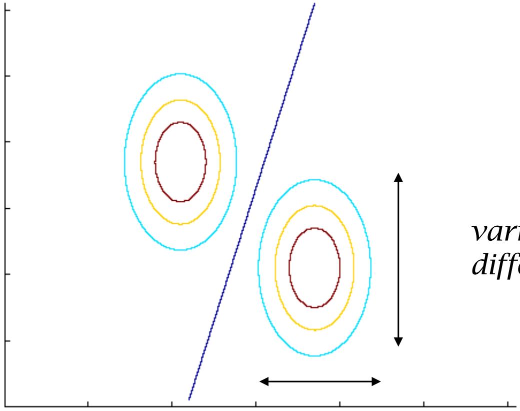
What if $\mathbf{x}$ is high-dimensional?

- For Gaussian Bayes Classifier, if input $\mathbf{x}$ is high-dimensional, then covariance matrix has many parameters $O(D^2)$
- Save some parameters by using a shared covariance for the classes, i.e. $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_l$.
- Any other idea you can think of? (next lecture)
- MLE in this case:

$$\Sigma = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}^{(n)} - \mu_{t^{(n)}})(\mathbf{x}^{(n)} - \mu_{t^{(n)}})^T$$

- Linear decision boundary (verify this mathematically!).

Decision Boundary: Shared Variances (between Classes)



variances may be different

Gaussian Discriminative Analysis vs Logistic Regression

- Binary classification: If you examine $p(t = 1|\mathbf{x})$ under GDA and assume $\Sigma_0 = \Sigma_1 = \Sigma$, you will find that it looks like this:

$$p(t|\mathbf{x}, \phi, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

  where $\mathbf{w}$ is an appropriate function of $(\phi, \mu_0, \mu_1, \Sigma)$, $\phi = p(t = 1)$. You derived this in hw2.
- GDA is similar to logistic regression (LR), parameter estimates are computed differently.
- When should we prefer GDA to LR, and vice versa?

Gaussian Discriminative Analysis vs Logistic Regression

- GDA is a generative model, LR is a discreminative model.
- GDA makes stronger modeling assumption: assumes class-conditional data is multivariate Gaussian
- If this is true, GDA is asymptotically efficient
- But LR is more robust, less sensitive to incorrect modeling assumptions (what loss is it optimizing?)
- Many class-conditional distributions lead to logistic classifier. (You saw an example in hw2)
- When these distributions are non-Gaussian (true almost always), LR usually beats GDA

Generative models - Recap

- GDA has quadratic, LR has linear decision boundary
- With shared covariance, GDA is similar to logistic regression.
- Generative models:
  - Flexible models, easy to add/remove class.
  - Handle missing data naturally
  - More "natural" way to think about things, but usually doesn't work as well.
- Tries to solve a hard problem in order to solve a easy problem.

Appendix: MLE for univariate Gaussian

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

$$
\begin{aligned}
\hat{\mu}_{\mathrm{ML}} &= \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^{(i)} \\
\hat{\sigma}_{\mathrm{ML}} &= \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}^{(i)} - \mu)^2}
\end{aligned}
$$