CSC2515 Lecture 8: Probabilistic Models

Roger Grosse

University of Toronto

Today's Agenda

- Bayesian parameter estimation: average predictions over all hypotheses, proportional to their posterior probability.
- Generative classification: learn to model the distributions of inputs belonging to each class
  - Naïve Bayes (discrete inputs)
  - Gaussian Discriminant Analysis (continuous inputs)

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.
- If you observe a single T in the test set, the log-likelihood is $-\infty$.

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.
- 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 $p(\mathcal{D} \mid \boldsymbol{\theta})$, same as in maximum likelihood
- 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.

Bayesian Parameter Estimation

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

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

- 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:

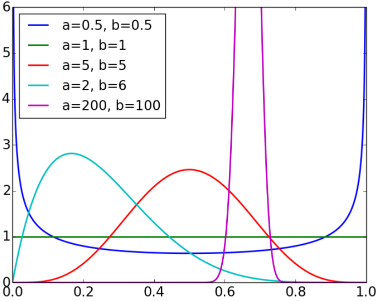
$$p(\theta; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\,\theta^{a-1}(1-\theta)^{b-1}.$$

  - 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 main thing 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:

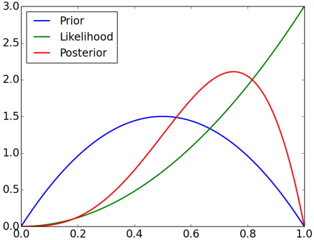
$$\mathbb{E}[\theta \mid \mathcal{D}] = \frac{N_H + a}{N_H + N_T + a + b}$$

- 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, and it's very useful.

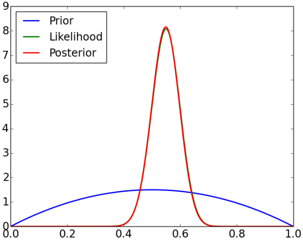
Bayesian Parameter Estimation

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.

Bayesian Parameter Estimation

- What do we actually do with the posterior?
- The posterior predictive distribution is the distribution over future observables given the past observations. We compute this by marginalizing out the parameter(s):

$$p(\mathcal{D}' \mid \mathcal{D}) = \int p(\boldsymbol{\theta} \mid \mathcal{D})\, p(\mathcal{D}' \mid \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}. \tag{1}$$

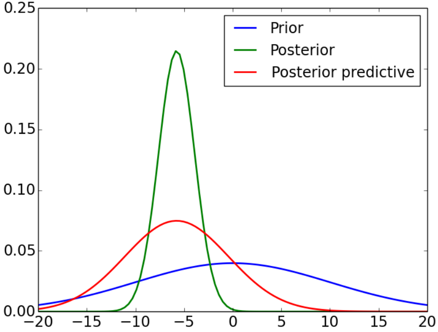
- For the coin flip example:

$$\begin{aligned} \theta_{\text{pred}} &= \Pr(x' = H \mid \mathcal{D}) \\ &= \int p(\theta \mid \mathcal{D}) \Pr(x' = H \mid \theta)\, \mathrm{d}\theta \\ &= \int \mathrm{Beta}(\theta; N_H + a, N_T + b) \cdot \theta\, \mathrm{d}\theta \\ &= \mathbb{E}_{\mathrm{Beta}(\theta; N_H + a, N_T + b)}[\theta] \\ &= \frac{N_H + a}{N_H + N_T + a + b}, \end{aligned} \tag{2}$$

Bayesian Parameter Estimation

Bayesian estimation of the mean temperature in Toronto

- Assume observations are i.i.d. Gaussian with known standard deviation σ and unknown mean μ
- Broad Gaussian prior over μ, centered at 0
- We can compute the posterior and posterior predictive distributions analytically (full derivation in notes)
- Why is the posterior predictive distribution more spread out than the posterior distribution?



Bayesian Parameter Estimation

Comparison of maximum likelihood and Bayesian parameter estimation

- Some advantages of the Bayesian approach
  - More robust to data sparsity
  - Incorporate prior knowledge
  - Smooth the predictions by averaging over plausible explanations
- Problem: maximum likelihood is an optimization problem, while Bayesian parameter estimation is an integration problem
  - This means maximum likelihood is much easier in practice, since we can just do gradient descent
  - Automatic differentiation packages make it really easy to compute gradients
  - There aren't any comparable black-box tools for Bayesian parameter estimation (although Stan can do quite a lot)

Maximum A-Posteriori Estimation

- 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

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

Maximum A-Posteriori Estimation

- Joint probability in the coin flip example:

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

- 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,

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

Maximum A-Posteriori Estimation

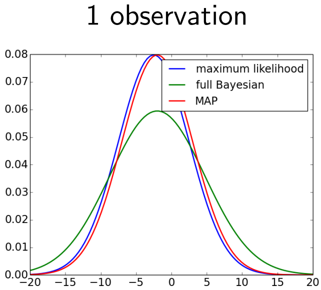
Comparison of estimates in the coin flip example:

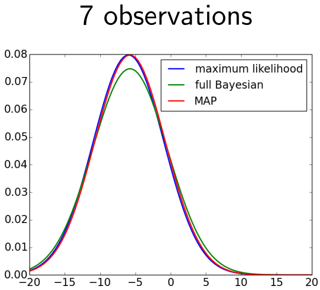
	Formula	$N_H = 2, N_T = 0$	$N_H = 55, N_T = 45$
$\hat{\theta}_{\mathrm{ML}}$	$\frac{N_H}{N_H+N_T}$	1	$\frac{55}{100} = 0.55$
θ_{pred}	$\frac{N_H+a}{N_H+N_T+a+b}$	$\frac{4}{6} \approx 0.67$	$\frac{57}{104} \approx 0.548$
$\hat{\theta}_{\mathrm{MAP}}$	$\frac{N_H+a-1}{N_H+N_T+a+b-2}$	$\frac{3}{4} = 0.75$	$\frac{56}{102} \approx 0.549$

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

Maximum A-Posteriori Estimation

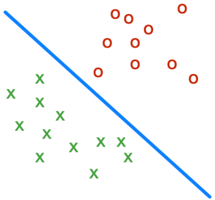
Comparison of predictions in the Toronto temperatures example



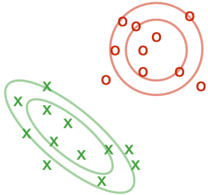


Generative Classifiers and Naïve Bayes

Two approaches to classification:



Discriminative



Generative

Generative vs. Discriminative

Two approaches to classification:

- Discriminative: directly learn to predict t as a function of $\mathbf{x}$.
  - Sometimes this means modeling $p(t \mid \mathbf{x})$ (e.g. logistic regression).
  - Sometimes this means learning a decision rule without a probabilistic interpretation (e.g. KNN, SVM).
- Generative: model the data distribution for each class separately, and make predictions using posterior inference.
  - Fit models of $p(t)$ and $p(\mathbf{x} \mid t)$.
  - Infer the posterior $p(t \mid \mathbf{x})$ using Bayes' Rule.

Bayes Classifier

- Bayes classifier: given features $\mathbf{x}$, we compute the posterior class probabilities using Bayes' Rule:

$$\overbrace{p(t \mid \mathbf{x})}^{\text{posterior}} = \frac{\overbrace{p(\mathbf{x} \mid t)}^{\substack{\text{class} \\ \text{likelihood}}} \; \overbrace{p(t)}^{\text{prior}}}{\underbrace{p(\mathbf{x})}_{\substack{\text{normalizing} \\ \text{constant}}}}$$

- Requires fitting $p(\mathbf{x} \mid t)$ and $p(t)$
- How can we compute $p(\mathbf{x})$ for binary classification?

$$p(\mathbf{x}) = p(\mathbf{x} \mid t = 0) \Pr(t = 0) + p(\mathbf{x} \mid t = 1) \Pr(t = 1)$$

- Note: sometimes it's more convenient to just compute the numerator and normalize.

Naïve Bayes

- **Example:** want to classify emails into spam ($t = 1$) or non-spam ($t = 0$) based on the words they contain.
  - Use bag-of-words features, i.e. a binary vector $\mathbf{x}$ where entry $x_j = 1$ if word j appeared in the email. (Assume a dictionary of D words.)
- Estimating the prior $p(t)$ is easy (e.g. maximum likelihood).
- **Problem:** $p(\mathbf{x} \mid t)$ is a joint distribution over D binary random variables, which requires 2^D entries to specify directly!
- 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, but you'll learn about them in detail in CSC2506.

Naïve Bayes

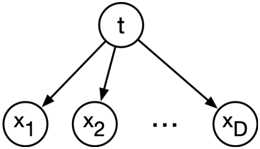
- Naïve Bayes makes the assumption that the word features x_j are conditionally independent given the class t.
  - This means x_i and x_j are independent under the conditional distribution $p(\mathbf{x} \mid t)$.
  - Note: this doesn't mean they're independent. (E.g., "Viagra" and "cheap" are correlated insofar as they both depend on t.)
  - Mathematically, this means the distribution factorizes:

$$p(t, x_1, \ldots, x_D) = p(t)\, p(x_1 \mid t) \cdots p(x_D \mid t).$$

- Compact representation of the joint distribution
  - Prior probability of class: $\Pr(t = 1) = \phi$
  - Conditional probability of word feature given class: $\Pr(x_j = 1 \mid t) = \theta_{jt}$
  - $2D + 1$ parameters total

Bayes Nets (Optional)

- 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, we can't infer causality without additional assumptions.
- You'll learn a lot about graphical models in CSC2506.

Naïve Bayes: Learning

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

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

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

Naïve Bayes: Learning

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

$$\sum_{i=1}^{N} \log p(x_j^{(i)} \mid t^{(i)}) = \sum_{i=1}^{N} t^{(i)} x_j^{(i)} \log \theta_{11} + \sum_{i=1}^{N} t^{(i)} (1 - x_j^{(i)}) \log(1 - \theta_{11})$$
$$+ \sum_{i=1}^{N} (1 - t^{(i)}) x_j^{(i)} \log \theta_{10} + \sum_{i=1}^{N} (1 - t^{(i)})(1 - x_j^{(i)}) \log(1 - \theta_{10})$$

- Obtain maximum likelihood estimates by setting derivatives to zero:

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

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

Naïve Bayes: Inference

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

$$p(t \,|\, \mathbf{x}) = \frac{p(t)\, p(\mathbf{x} \,|\, t)}{\sum_{t'} p(t')\, p(\mathbf{x} \,|\, t')}$$
$$= \frac{p(t) \prod_{j=1}^{D} p(x_j \,|\, t)}{\sum_{t'} p(t') \prod_{j=1}^{D} p(x_j \,|\, t')}$$

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

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

Naïve Bayes: Decisions

- Once we compute $p(t \mid \mathbf{x})$, what do we do with it?
- Sometimes we want to make a single prediction or decision y. This is a decision theory problem, just like when we analyzed the bias/variance/Bayes-error decomposition.
  - Define a loss function $\mathcal{L}(y, t)$ and choose $y_\star = \arg\min_y \mathbb{E}[\mathcal{L}(y, t) \mid \mathbf{x}]$.
- Examples
  - Squared error loss: choose $y_\star = \mathbb{E}[t \mid \mathbf{x}]$
  - 0-1 loss: choose the most likely category
  - Cross-entropy loss: return the probability $y = \Pr(t = 1 \mid \mathbf{x})$
  - Asymmetric loss (e.g. false positives are much worse than false negatives for spam filtering): apply a threshold other than 0.5.
    - Warning: this is theoretically tidy, but doesn't really work unless you're careful to obtain calibrated posterior probabilities.
    - "Calibrated" means all the times you predict (say) $\Pr(t = k \mid \mathbf{x}) = 0.9$ should be correct 90% on average.
    - Naïve Bayes is generally not calibrated due to the "naïve" conditional independence assumption.

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.
  - The problem is the "naïve" independence assumption.
  - We're covering it primarily as a stepping stone towards latent variable models.

Gaussian Discriminant Analysis

Motivation

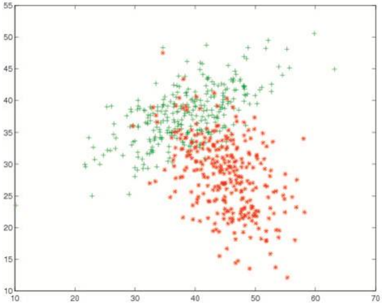
- Generative models — model $p(t)$ and $p(\mathbf{x} \mid t)$
- Recall that $p(\mathbf{x} \mid t = k)$ may be very complex

$$p(x_1, \cdots, x_D \mid t) = p(x_1 \mid x_2, \cdots, x_D, t) \cdots p(x_{D-1} \mid x_D, t) p(x_D \mid t)$$

- Naïve Bayes used a conditional independence assumption to make everything tractable.
- For continuous inputs, we can instead make it tractable by using a simple distribution: multivariate Gaussians.

Classification: Diabetes Example

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



- How can we model $p(\mathbf{x} \mid t = k)$? Multivariate Gaussian

Multivariate Parameters

- Mean

$$\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}] = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_D \end{pmatrix}$$

- Covariance

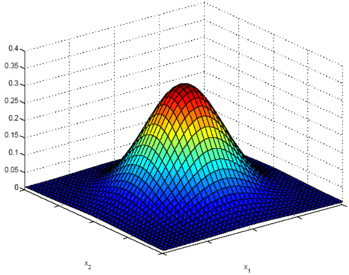
$$\boldsymbol{\Sigma} = \mathrm{Cov}(\mathbf{x}) = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})^\top (\mathbf{x} - \boldsymbol{\mu})] = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1D} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{D1}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

- These statistics uniquely define a multivariate Gaussian distribution. (This is not true for distributions in general!)

Multivariate Gaussian Distribution

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

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



- Mahalanobis distance $(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$ measures the distance from $\mathbf{x}$ to $\boldsymbol{\mu}$ in a space stretched according to $\boldsymbol{\Sigma}$.

Bivariate Gaussian

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

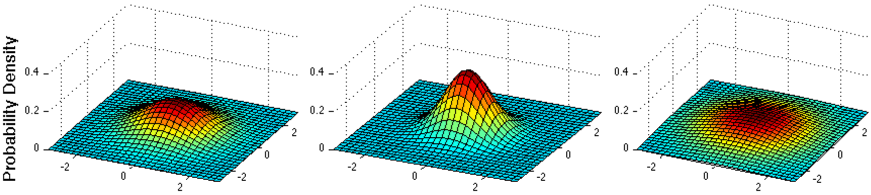


Figure: Probability density function

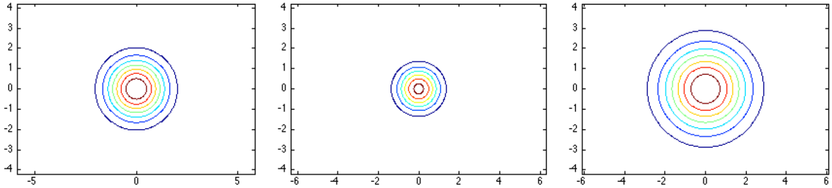


Figure: Contour plot of the pdf

Bivariate Gaussian

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

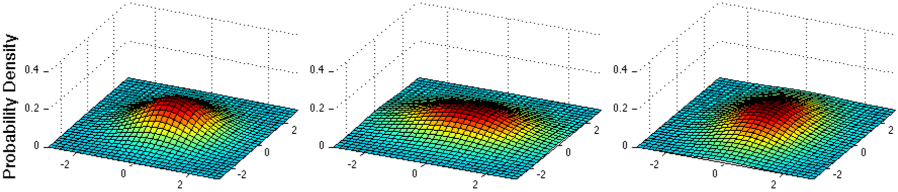


Figure: Probability density function

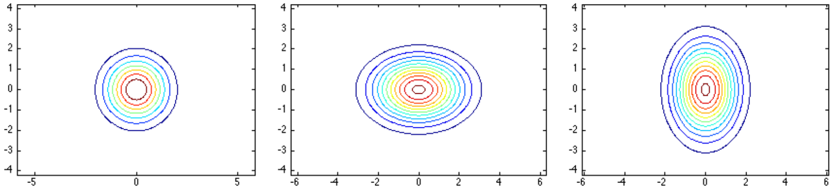


Figure: Contour plot of the pdf

Bivariate Gaussian

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

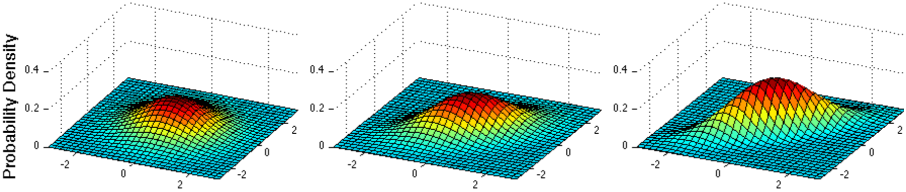


Figure: Probability density function

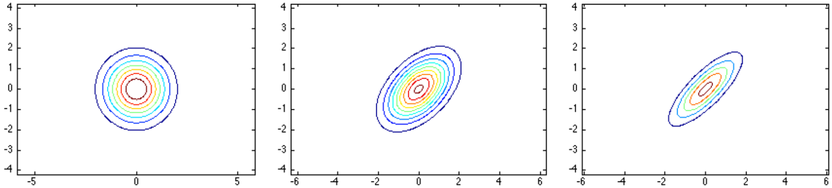


Figure: Contour plot of the pdf

Bivariate Gaussian

$Cov(x_1, x_2) = 0$ $\quad$ $Cov(x_1, x_2) > 0$ $\quad$ $Cov(x_1, x_2) < 0$

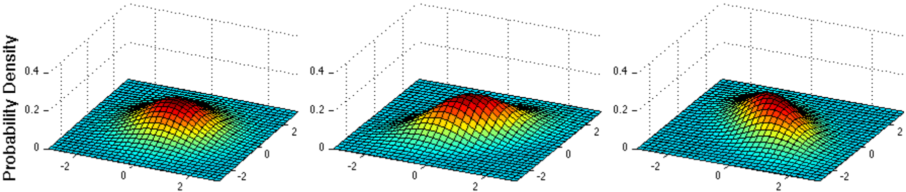


Figure: Probability density function

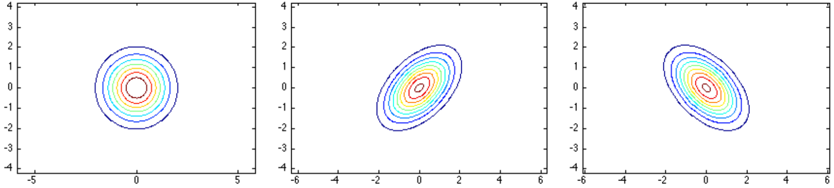
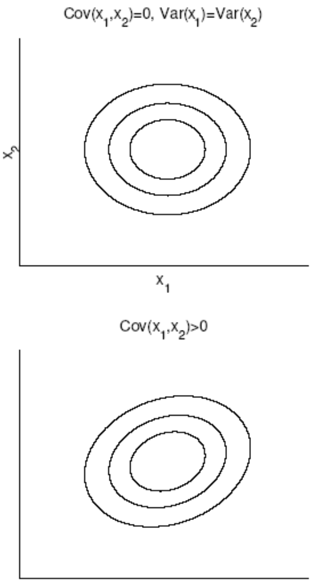
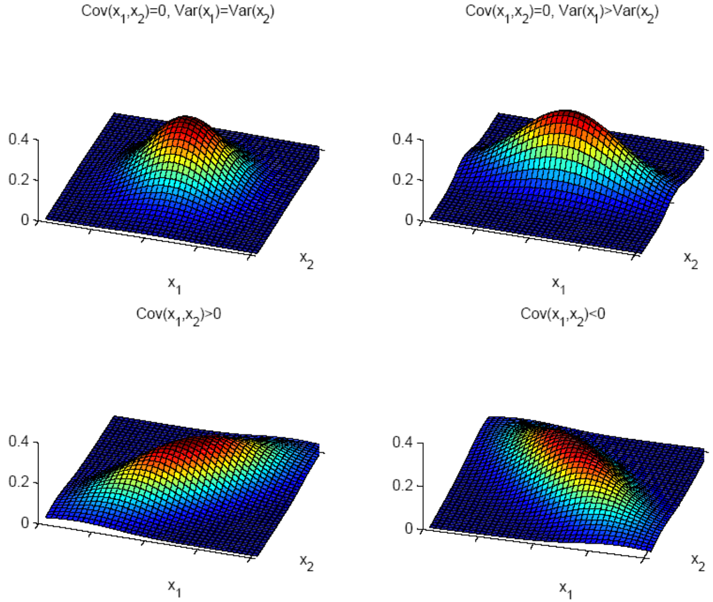


Figure: Contour plot of the pdf

Bivariate Gaussian



Bivariate Gaussian



Gaussian Discriminant Analysis

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

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

  where $|\boldsymbol{\Sigma}_k|$ denotes the determinant of the matrix.

- Each class k has associated mean vector $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$
- How many parameters?
  - Each $\boldsymbol{\mu}_k$ has D parameters, for DK total.
  - Each $\boldsymbol{\Sigma}_k$ has $\mathcal{O}(D^2)$ parameters, for $\mathcal{O}(D^2K)$ — could be hard to estimate (more on that later).

GDA: Learning

- Learn the parameters for each class using maximum likelihood
- For simplicity, assume binary classification

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

- You can compute the ML estimates in closed form (ϕ and $\boldsymbol{\mu}_k$ are easy, $\boldsymbol{\Sigma}_k$ is tricky)

$$\begin{aligned}
\phi &= \frac{1}{N} \sum_{i=1}^{N} r_1^{(i)} \\
\boldsymbol{\mu}_k &= \frac{\sum_{i=1}^{N} r_k^{(i)} \cdot \mathbf{x}^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}} \\
\boldsymbol{\Sigma}_k &= \frac{1}{\sum_{i=1}^{N} r_k^{(i)}} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top \\
r_k^{(i)} &= \mathbb{1}[t^{(i)} = k]
\end{aligned}$$

GDA Decision Boundary

- Recall: for Bayes classifiers, we compute the decision boundary with Bayes' Rule:

$$p(t \mid \mathbf{x}) = \frac{p(t)\, p(\mathbf{x} \mid t)}{\sum_{t'} p(t')\, p(\mathbf{x} \mid t')}$$

- Plug in the Gaussian $p(\mathbf{x} \mid t)$:

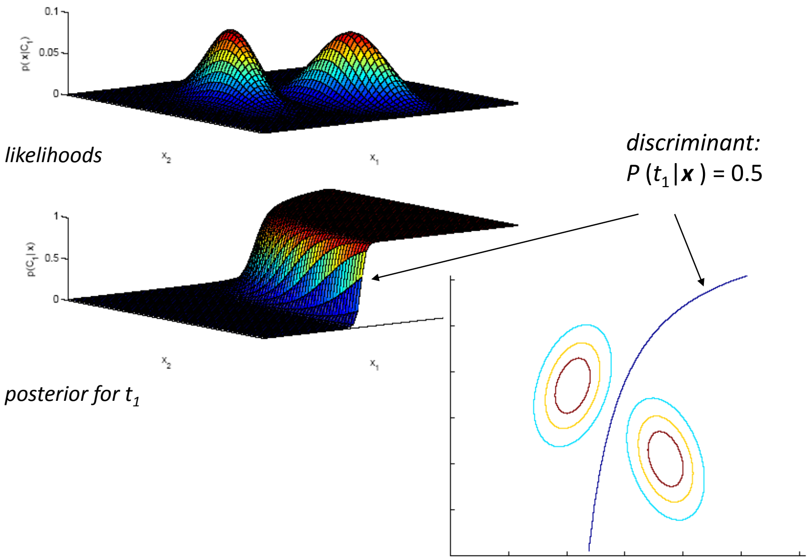
$$\begin{aligned}
\log p(t_k|\mathbf{x}) &= \log p(\mathbf{x}|t_k) + \log p(t_k) - \log p(\mathbf{x}) \\
&= -\frac{D}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) + \\
&\quad + \log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

- Decision boundary:

$$(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) = (\mathbf{x} - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1}(\mathbf{x} - \boldsymbol{\mu}_\ell) + \mathrm{Const}$$

- What's the shape of the boundary?
  - We have a quadratic function in $\mathbf{x}$, so the decision boundary is a conic section!

GDA Decision Boundary



GDA Decision Boundary

- Our equation for the decision boundary:

$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) = (\mathbf{x}-\boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}_\ell^{-1}(\mathbf{x}-\boldsymbol{\mu}_\ell) + \mathrm{Const}$$

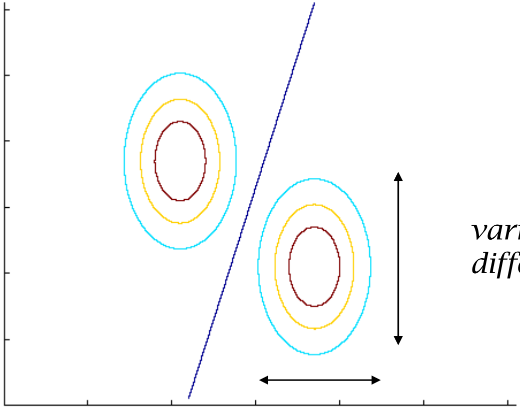
- Expand the product and factor out constants (w.r.t. $\mathbf{x}$):

$$\mathbf{x}^\top \boldsymbol{\Sigma}_k^{-1}\mathbf{x} - 2\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}_k^{-1}\mathbf{x} = \mathbf{x}^\top \boldsymbol{\Sigma}_\ell^{-1}\mathbf{x} - 2\boldsymbol{\mu}_\ell^\top \boldsymbol{\Sigma}_\ell^{-1}\mathbf{x} + \mathrm{Const}$$

- What if all classes share the same covariance $\boldsymbol{\Sigma}$?
  - We get a linear decision boundary!

$$\begin{aligned} -2\boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} &= -2\boldsymbol{\mu}_\ell^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} + \mathrm{Const} \\ (\boldsymbol{\mu}_k - \boldsymbol{\mu}_\ell)^\top \boldsymbol{\Sigma}^{-1}\mathbf{x} &= \mathrm{Const} \end{aligned}$$

GDA Decision Boundary: Shared Covariances



variances may be different

GDA vs Logistic Regression

- Binary classification: If you examine $p(t = 1 \mid \mathbf{x})$ under GDA and assume $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \mathbf{\Sigma}$, you will find that it looks like this:

$$p(t \mid \mathbf{x}, \phi, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \mathbf{\Sigma}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x} - b)}$$

  where $(\mathbf{w}, b)$ are chosen based on $(\phi, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \mathbf{\Sigma})$.

- Same model as logistic regression!

GDA vs Logistic Regression

When should we prefer GDA to LR, and vice versa?

- GDA makes a stronger modeling assumption: assumes class-conditional data is multivariate Gaussian
  - If this is true, GDA is asymptotically efficient (best model in limit of large N)
  - If it's not true, the quality of the predictions might suffer.
- Many class-conditional distributions lead to logistic classifier.
  - When these distributions are non-Gaussian (i.e., almost always), LR usually beats GDA
- GDA can handle easily missing features (how do you do that with LR?)

Gaussian Naive Bayes

- What if $\mathbf{x}$ is high-dimensional?
  - The $\mathbf{\Sigma}_k$ have $\mathcal{O}(D^2K)$ parameters, which can be a problem if D is large.
  - We already saw we can save some a factor of K by using a shared covariance for the classes.
  - Any other idea you can think of?
- **Naive Bayes**: Assumes features independent given the class

$$p(\mathbf{x} \mid t = k) = \prod_{j=1}^{D} p(x_j \mid t = k)$$

- Assuming likelihoods are Gaussian, how many parameters required for Naive Bayes classifier?
  - This is equivalent to assuming the x_j are uncorrelated, i.e. $\mathbf{\Sigma}$ is diagonal.
  - Hence, only D parameters for $\mathbf{\Sigma}$!

Gaussian Naïve Bayes

- Gaussian Naïve Bayes classifier assumes that the likelihoods are Gaussian:

$$p(x_j \mid t = k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left[\frac{-(x_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right]$$

(this is just a 1-dim Gaussian, one for each input dimension)

- Model the same as GDA with diagonal covariance matrix
- Maximum likelihood estimate of parameters

$$\mu_{jk} = \frac{\sum_{i=1}^{N} r_k^{(i)} x_j^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}}$$

$$\sigma_{jk}^2 = \frac{\sum_{i=1}^{N} r_k^{(i)} (x_j^{(i)} - \mu_{jk})^2}{\sum_{i=1}^{N} r_k^{(i)}}$$

$$r_k^{(i)} = \mathbb{1}[t^{(i)} = k]$$

Decision Boundary: Isotropic

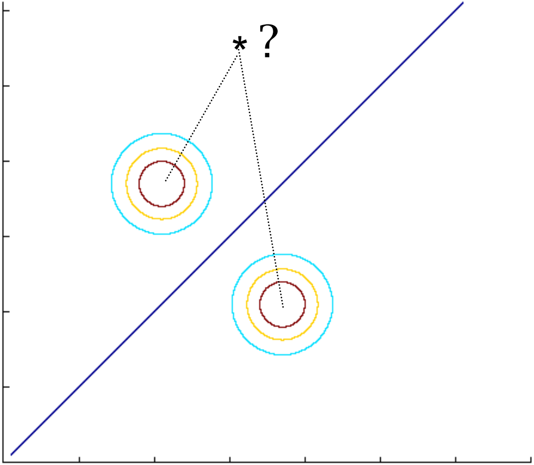
- We can go even further and assume the covariances are spherical, or isotropic.
- In this case: $\mathbf{\Sigma} = \sigma^2 \mathbf{I}$ (just need one parameter!)
- Going back to the class posterior for GDA:

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

- Suppose for simplicity that $p(t)$ is uniform. Plugging in $\mathbf{\Sigma} = \sigma^2 \mathbf{I}$ and simplifying a bit,

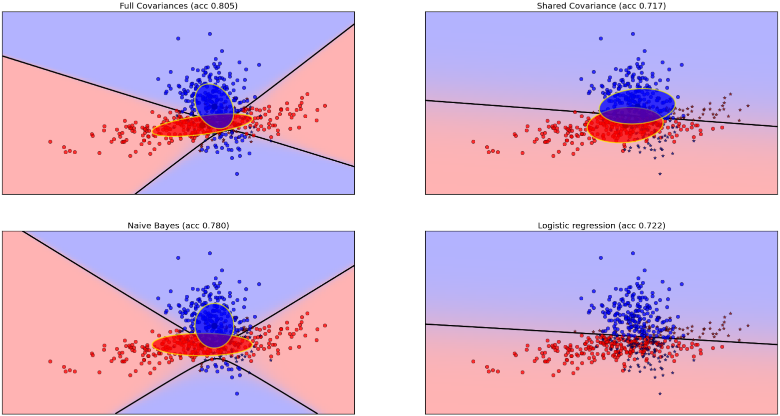
$$\begin{aligned}\log p(t_k \,|\, \mathbf{x}) - \log p(t_\ell \,|\, \mathbf{x}) &= -\frac{1}{2\sigma^2}\left[(\mathbf{x} - \boldsymbol{\mu}_k)^\top(\mathbf{x} - \boldsymbol{\mu}_k) - (\mathbf{x} - \boldsymbol{\mu}_\ell)^\top(\mathbf{x} - \boldsymbol{\mu}_\ell)\right] \\ &= -\frac{1}{2\sigma^2}\left[\|\mathbf{x} - \boldsymbol{\mu}_k\|^2 - \|\mathbf{x} - \boldsymbol{\mu}_\ell\|^2\right]\end{aligned}$$

Decision Boundary: Isotropic



- The decision boundary bisects the class means!

Example


Full Covariances (acc 0.805)
Shared Covariance (acc 0.717)
Naive Bayes (acc 0.780)
Logistic regression (acc 0.722)