Introduction to Machine Learning in Geosciences GEO371T/GEO399D.1

Naive Bayes Classifier

Mrinal K. Sen
Geosciences
UT Austin

October 12, 2023

Content

- Classification – Multi-dimensional (Gaussian) Bayes classifier
- Estimate probability densities from data
- Naive Bayes classifier

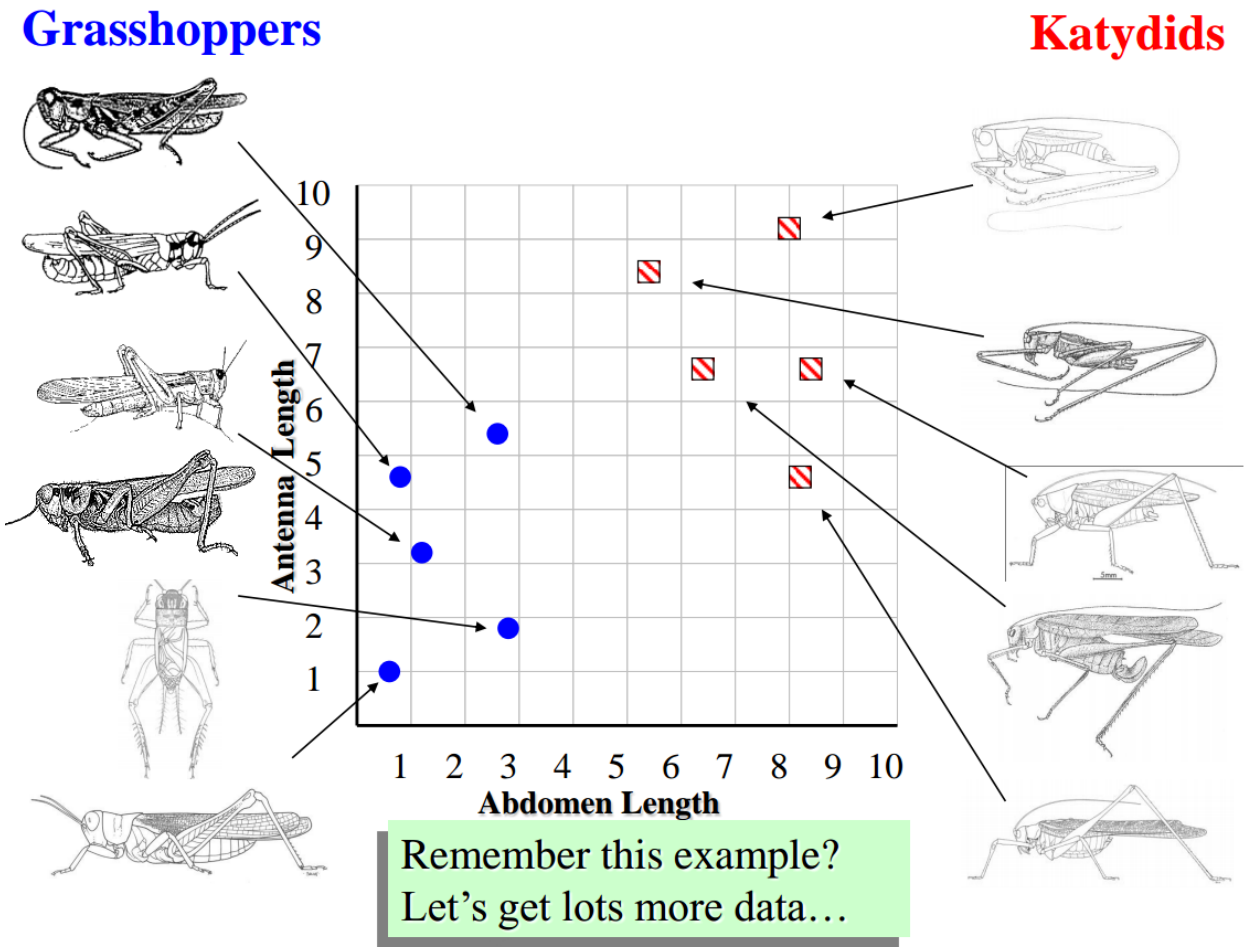
https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/09_naive_bayes.pdf

Generative vs Discriminative

Two approaches to classification

- Discriminative classifiers estimate parameters of decision boundary/class separator directly from labeled examples
  - Learn $p(y|x)$ directly (logistic regression models)
  - Learn mappings from inputs to classes (least-squares, neural nets)
- Generative approach: model the distribution of inputs characteristic of the class (Bayes classifier)
  - Build a model of $p(\mathbf{x}|y)$
  - Apply Bayes Rule.

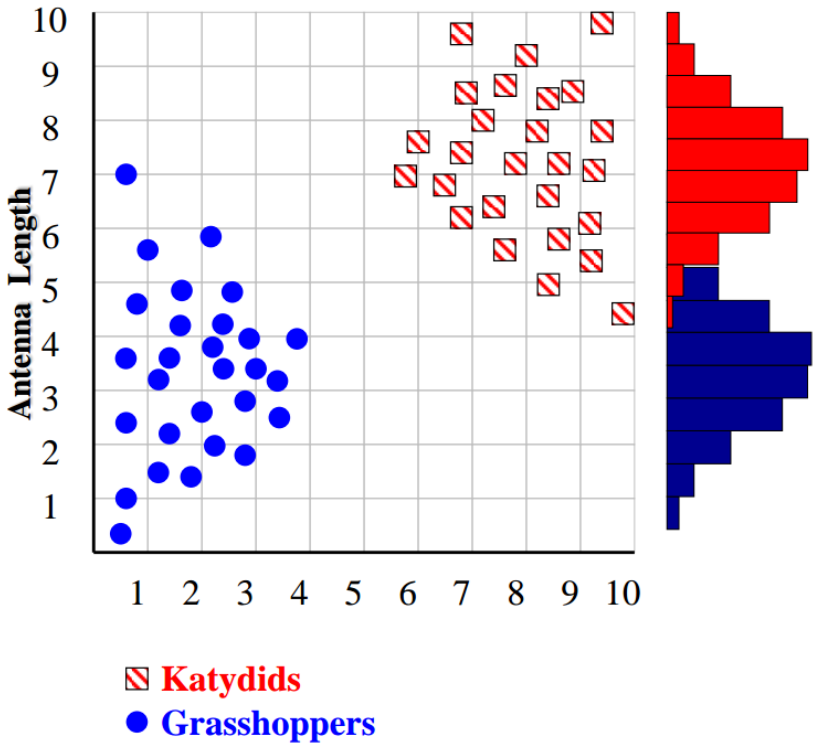
Example



https://www.cs.ucr.edu/~eamonn/CE/Bayesian

Example

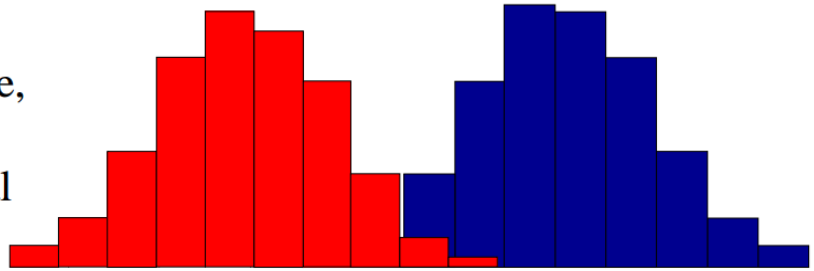
With a lot of data, we can build a histogram. Let us just build one for "Antenna Length" for now…



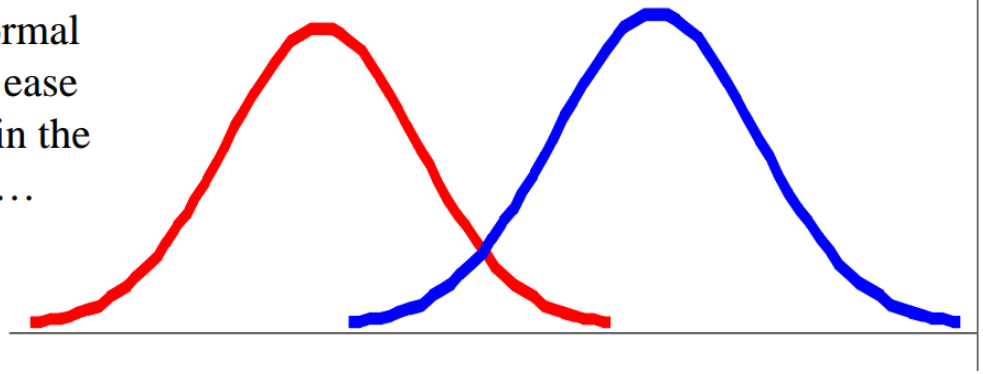
https://www.cs.ucr.edu/~eamonn/CE/Bayesian

Example

We can leave the histograms as they are, or we can summarize them with two normal distributions.



Let us us two normal distributions for ease of visualization in the following slides…



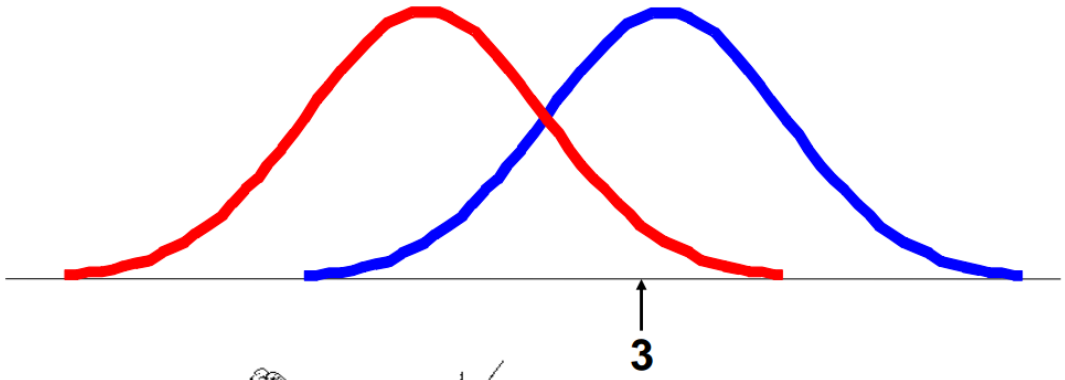
https://www.cs.ucr.edu/~eamonn/CE/Bayesian

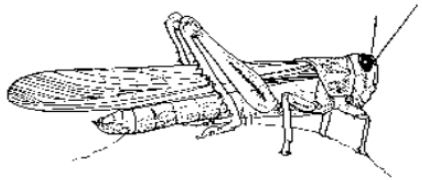
Example

• We want to classify an insect we have found. Its antennae are 3 units long. How can we classify it?

• We can just ask ourselves, give the distributions of antennae lengths we have seen, is it more *probable* that our insect is a **Grasshopper** or a **Katydid**.
• There is a formal way to discuss the most *probable* classification…

$p(c_j \mid d)$ = probability of class c_j, *given* that we have observed d





Antennae length is **3**

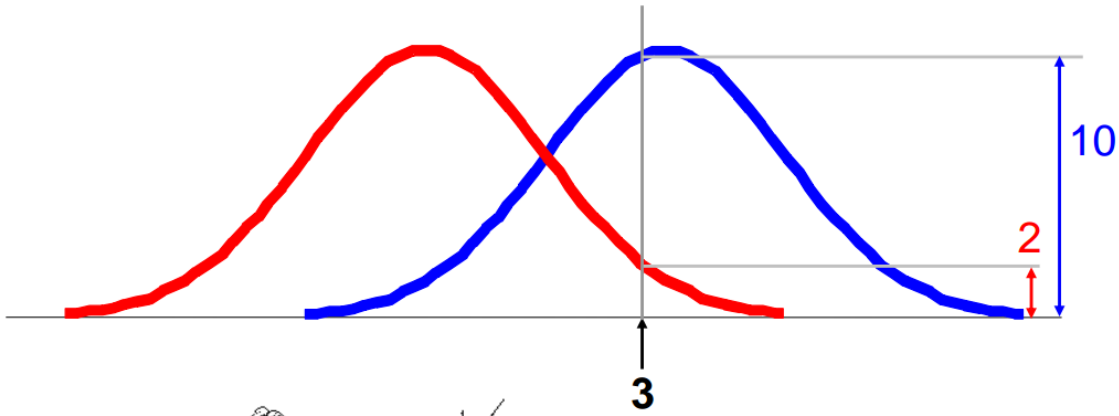
https://www.cs.ucr.edu/~eamonn/CE/Bayesian

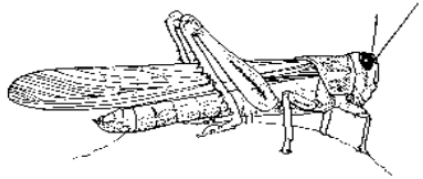
Example

$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **3**) = 10 / (10 + 2) = 0.833

P(**Katydid** | **3**) = 2 / (10 + 2) = 0.166





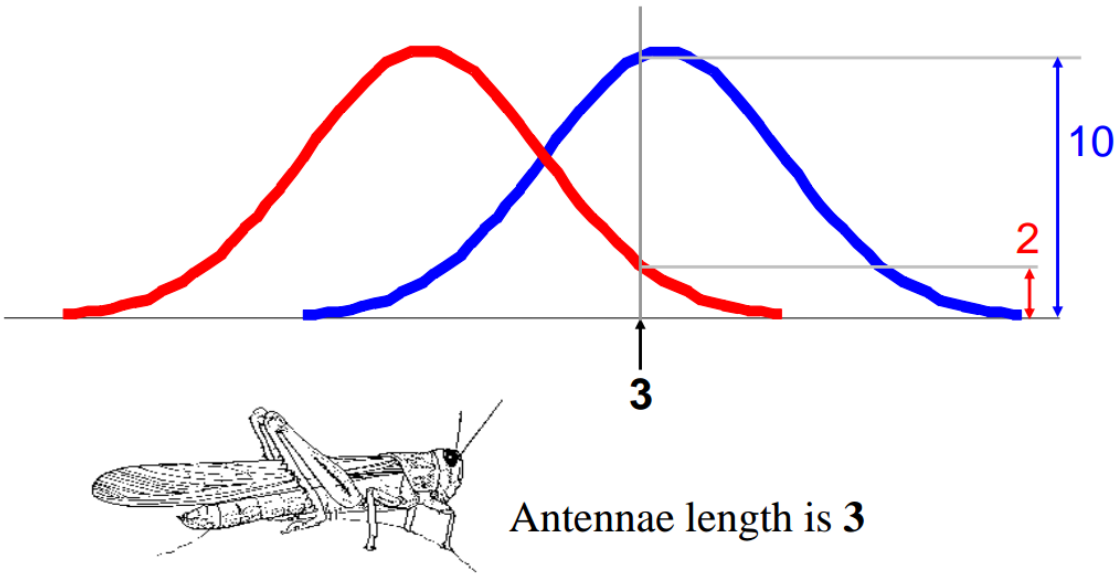
Antennae length is **3**

https://www.cs.ucr.edu/~eamonn/CE/Bayesian

Example

P(**Grasshopper** | **3**) = 10 / (10 + 2) = 0.833

P(**Katydid** | **3**) = 2 / (10 + 2) = 0.166



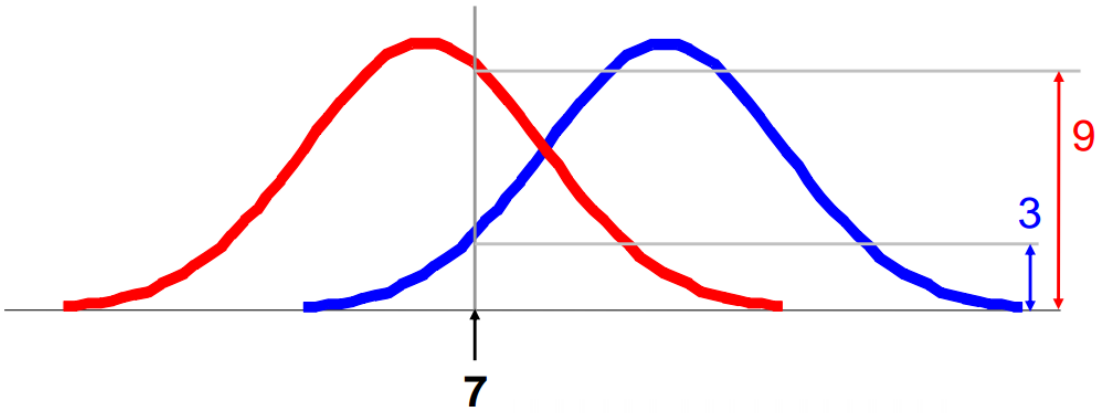
https://www.cs.ucr.edu/~eamonn/CE/Bayesian2

Example

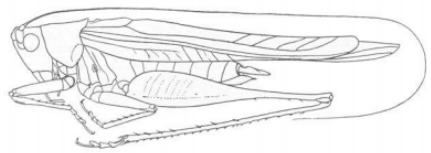
$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **7**) = 3 / (3 + 9) = 0.250

P(**Katydid** | **7**) = 9 / (3 + 9) = 0.750



Antennae length is **7**



https://www.cs.ucr.edu/~eamonn/CE/Bayesian2

Example

Bayes Classifiers

That was a visual intuition for a simple case of the Bayes classifier, also called:

- Idiot Bayes
- Naïve Bayes
- Simple Bayes

We are about to see some of the mathematical formalisms, and more examples, but keep in mind the basic idea.

Find out the probability of the previously unseen instance belonging to each class, then simply pick the most probable class.

`https://www.cs.ucr.edu/~eamonn/CE/Bayesian2`

Example

Bayes Classifiers

- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

- $p(c_j \mid d)$ = probability of instance d being in class c_j,
  This is what we are trying to compute
- $p(d \mid c_j)$ = probability of generating instance d given class c_j,
  We can imagine that being in class c_j, causes you to have feature d with some probability
- $p(c_j)$ = probability of occurrence of class c_j,
  This is just how frequent the class c_j, is in our database
- $p(d)$ = probability of instance d occurring

  This can actually be ignored, since it is the same for all classes

https://www.cs.ucr.edu/~eamonn/CE/Bayesian2

Bayes Classifier

- Aim to diagnose whether patient has diabetes: classify into one of two classes (yes C=1; no C=0)
- Run a series of tests
- Given patient's results: $x = [x_1, x_2, ..., x_d]^T$, we want to update class probabilities using Bayes Rule:

$$p(C|\mathbf{x}) = \frac{p(\mathbf{x}|C)p(C)}{p(x)}$$

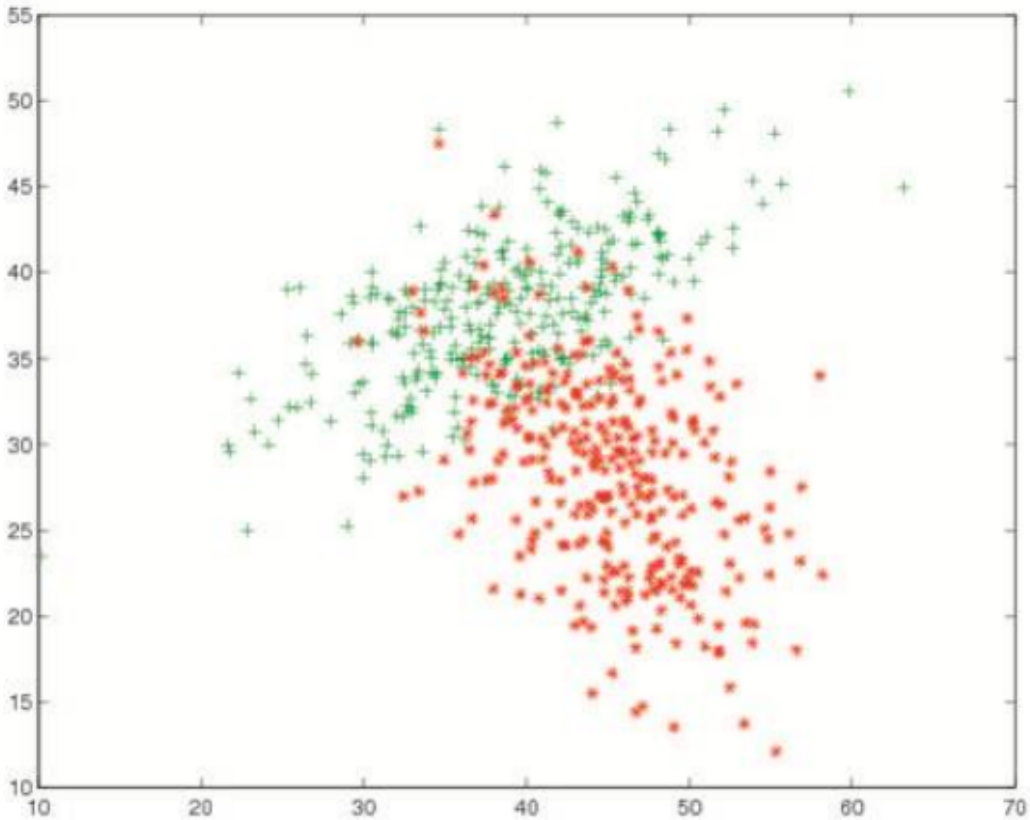
- More formally

$$\text{posterior} = \frac{\text{Class Likelihood} X \text{Prior}}{\text{Evidence}}$$

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

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

Example



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:

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

- Each class k has associated mean vector μ_k and covariance matrix Σ_k
- Typically the classes share a single covariance matrix ("share" means that they have the same parameters; the covariance matrix in this case):

$$\Sigma_1 = \Sigma_2 = ... = \Sigma_k$$

Multivariate Data

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

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}$$

Multivariate Parameters

- Mean

$$\mathbb{E}[\mathbf{x}] = [\mu_1, \cdots, \mu_d]^T$$

- Covariance

$$\Sigma = Cov(\mathbf{x}) = \mathbb{E}[(\mathbf{x} - \mu)^T(\mathbf{x} - \mu)] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

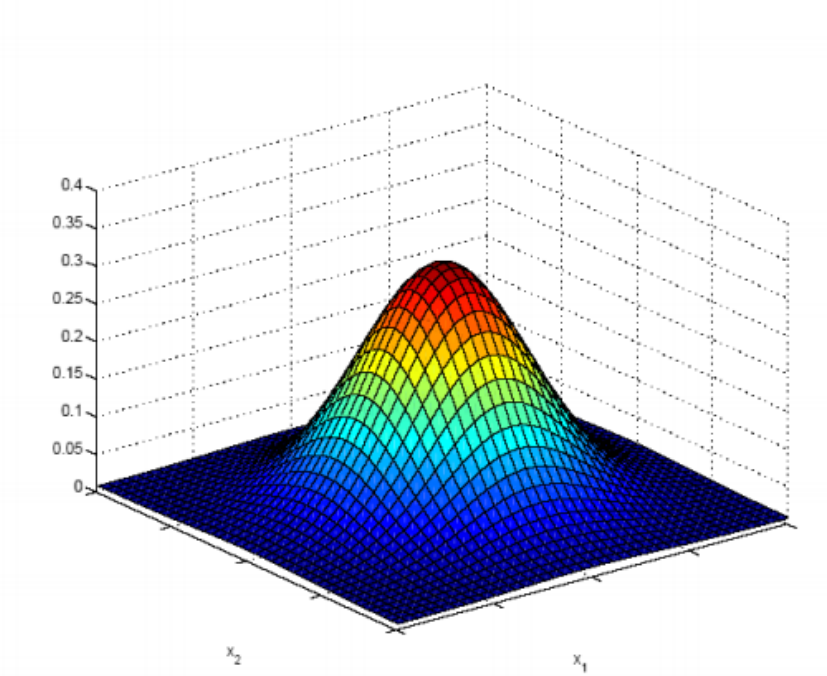
- Correlation $= Corr(\mathbf{x})$ is the covariance divided by the product of standard deviation

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

Multivariate Gaussian Distribution

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

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



- Mahalanobis distance $(\mathbf{x}-\mu_k)^T\Sigma^{-1}(\mathbf{x}-\mu_k)$ measures the distance from $\mathbf{x}$ to μ in terms of Σ

Bivariate Normal

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

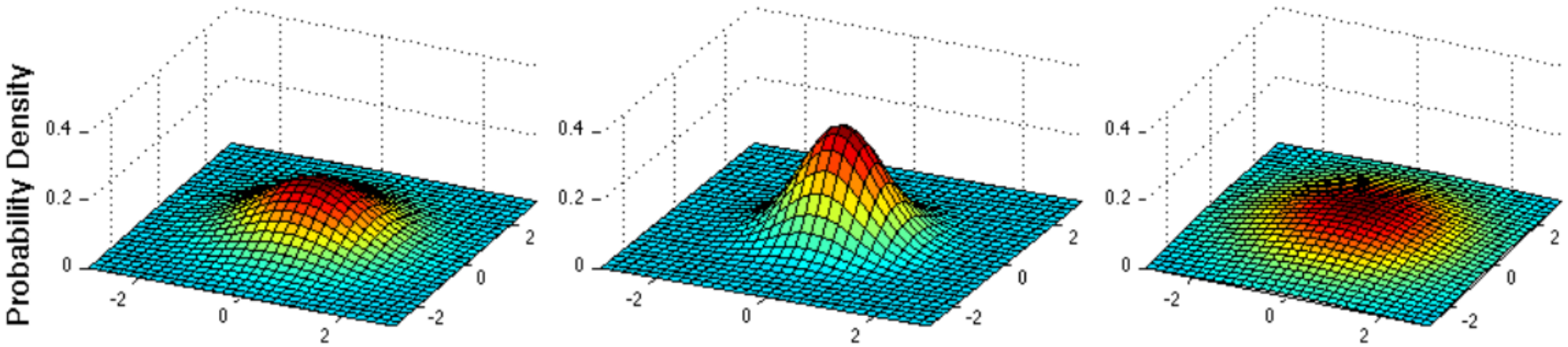


Figure : Probability density function

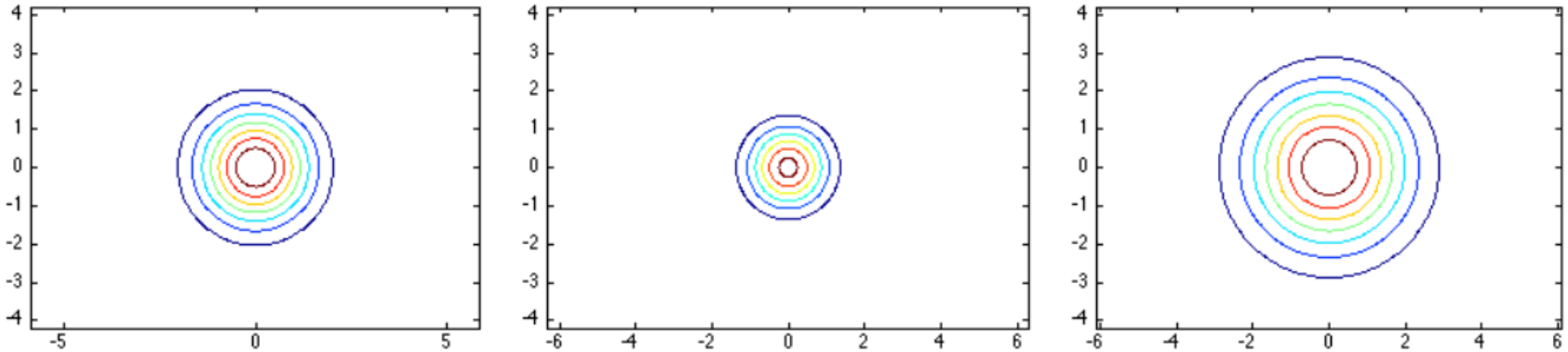


Figure : Contour plot of the pdf

Bivariate Normal

$var(x_1) = var(x_2)$ $var(x_1) > var(x_2)$ $var(x_1) < var(x_2)$

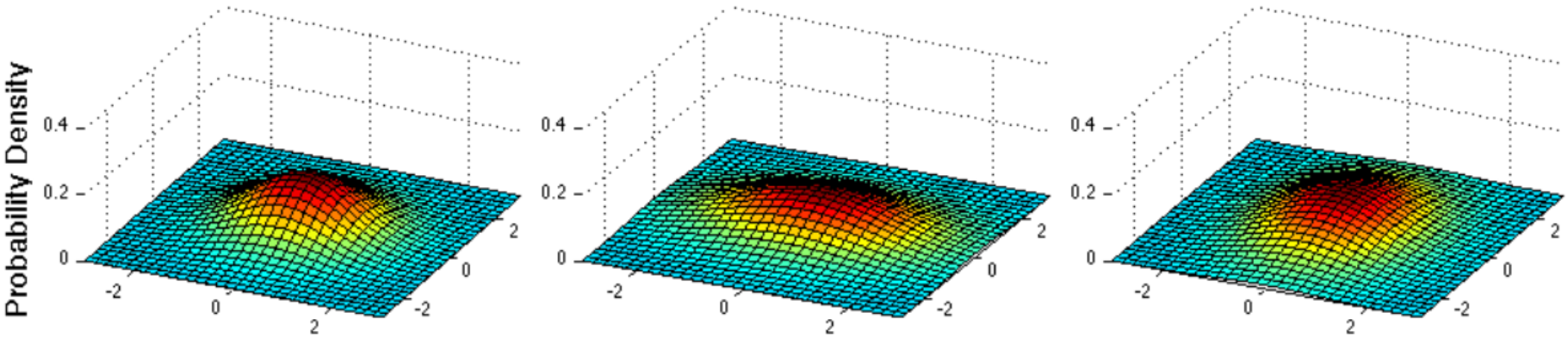


Figure : Probability density function

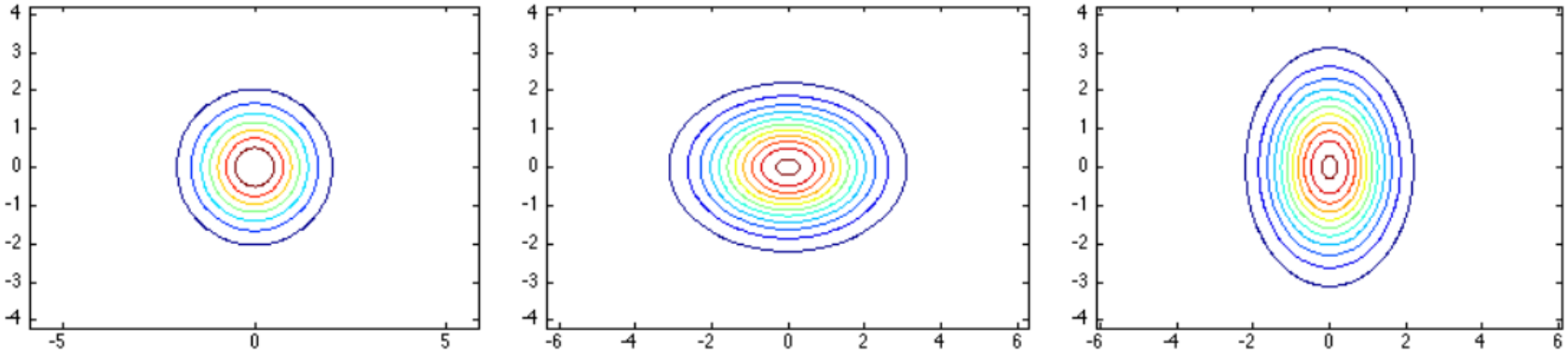


Figure : Contour plot of the pdf

Bivariate Normal

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

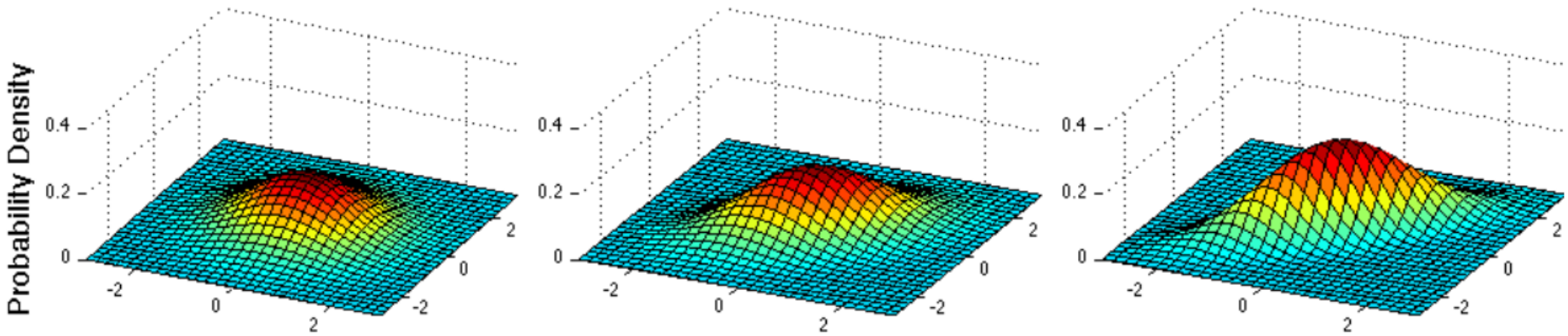


Figure : Probability density function

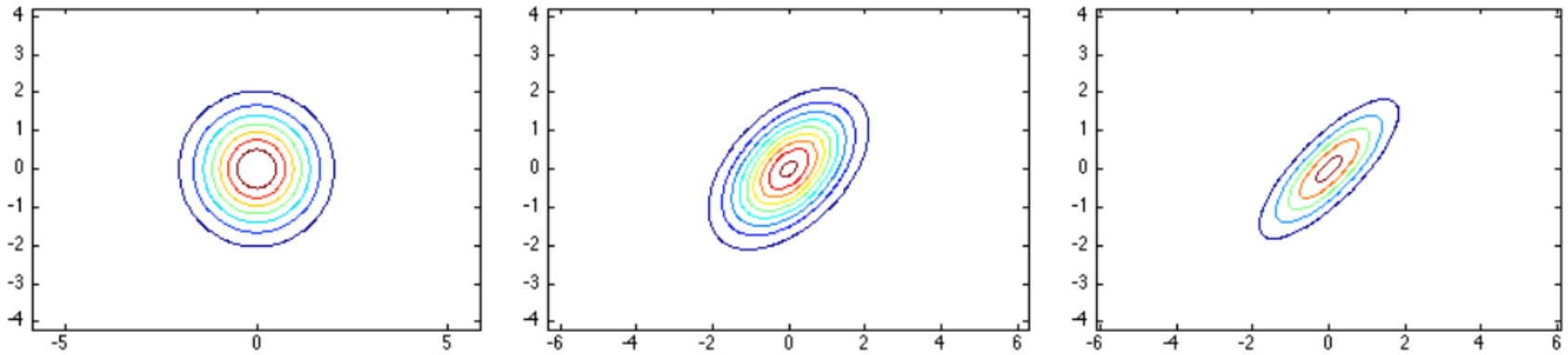


Figure : Contour plot of the pdf

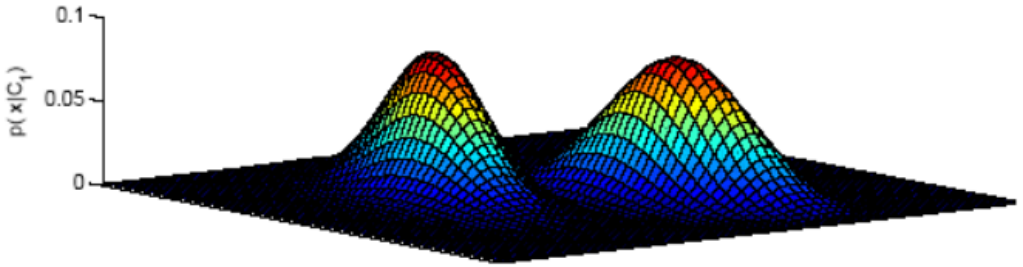
Gaussian Discriminant Analysis (Gaussian Bayes Classifier)

- GDA (GBC) decision boundary is based on class posterior:

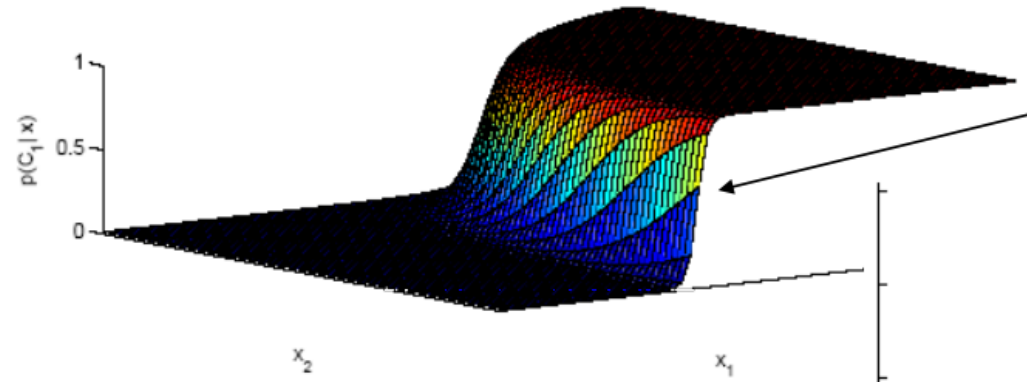
$$\begin{aligned}\log p(t_k|\mathbf{x}) &= \log p(\mathbf{x}|t_k) + \log p(t_k) - \log p(\mathbf{x}) \\ &= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_k^{-1}| - \frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k) + \\ &\quad + \log p(t_k) - \log p(\mathbf{x})\end{aligned}$$

- Decision: take the class with the highest posterior probability

Gaussian Discriminant Analysis (Gaussian Bayes Classifier)

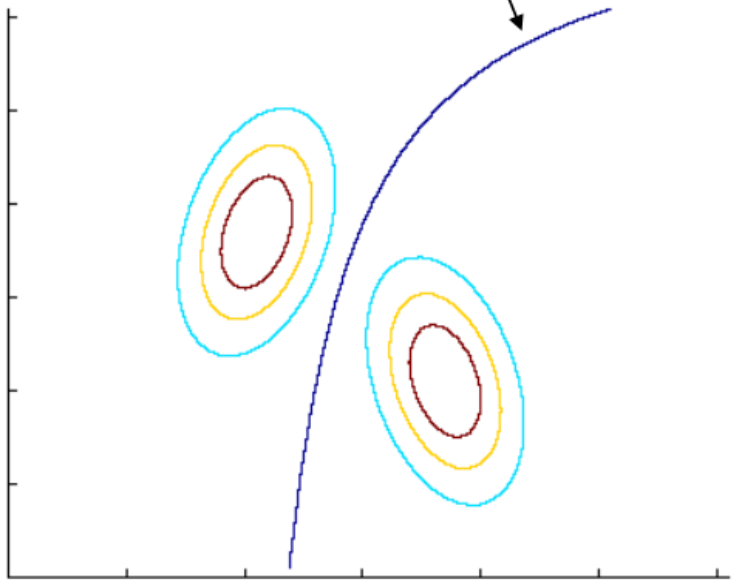


likelihoods



posterior for t_1

discriminant:
$P(t_1 | \boldsymbol{x}) = 0.5$



Learning

- Learn the parameters using maximum likelihood

$$\begin{aligned}\ell(\phi, \mu_0, \mu_1, \Sigma) &= -\log \prod_{n=1}^{N} p(\mathbf{x}^{(n)}, t^{(n)} | \phi, \mu_0, \mu_1, \Sigma) \\ &= -\log \prod_{n=1}^{N} p(\mathbf{x}^{(n)} | t^{(n)}, \mu_0, \mu_1, \Sigma) p(t^{(n)} | \phi)\end{aligned}$$

- What have we assumed?

More on MLE

- Assume the prior is Bernoulli (we have two classes)

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

- You can compute the ML estimate in closed form

$$\begin{aligned}
\phi &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{1}[t^{(n)}=1] \\
\mu_0 &= \frac{\sum_{n=1}^{N}\mathbb{1}[t^{(n)}=0]\cdot\mathbf{x}^{(n)}}{\sum_{n=1}^{N}\mathbb{1}[t^{(n)}=0]} \\
\mu_1 &= \frac{\sum_{n=1}^{N}\mathbb{1}[t^{(n)}=1]\cdot\mathbf{x}^{(n)}}{\sum_{n=1}^{N}\mathbb{1}[t^{(n)}=1]} \\
\Sigma &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)}-\mu_{t^{(n)}})(\mathbf{x}^{(n)}-\mu_{t^{(n)}})^T
\end{aligned}$$

Gaussian Discriminative Analysis or Logistic Regression

- If you examine $p(t = 1|\mathbf{x})$ under GDA, 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 $(\phi, \mu_0, \mu_1, \Sigma)$

- So the decision boundary has the same form as logistic regression!
- When should we prefer GDA to LR, and vice versa?

Gaussian Discriminative Analysis or Logistic Regression

- GDA makes 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)
- But LR is more robust, less sensitive to incorrect modeling assumptions
- Many class-conditional distributions lead to logistic classifier
- When these distributions are non-Gaussian, in limit of large N, LR beats GDA

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
- Save some parameters by using a shared covariance for the classes
- Any other idea you can think of?

Naive Bayes

- **Naive Bayes** is an alternative generative model: Assumes features independent given the class

$$p(\mathbf{x}|t=k) = \prod_{i=1}^{d} p(x_i|t=k)$$

- Assuming likelihoods are Gaussian, how many parameters required for Naive Bayes classifier?
- Important note: Naive Bayes does not assume a particular distribution

Naive Bayes

Given

- prior $p(t = k)$
- assuming features are conditionally independent given the class
- likelihood $p(x_i|t = k)$ for each x_i

The decision rule

$$y = arg\max_k p(t = k) \prod_{i=1}^{d} p(x_i|t = k)$$

- If the assumption of conditional independence holds, NB is the optimal classifier
- If not, a heavily regularized version of generative classifier
- What's the regularization?
- Note: NB's assumptions (cond. independence) typically do not hold in practice. However, the resulting algorithm still works well on many problems, and it typically serves as a decent baseline for more sophisticated models

Gaussian Naive Bayes

- **Gaussian Naive Bayes** classifier assumes that the likelihoods are Gaussian:

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

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

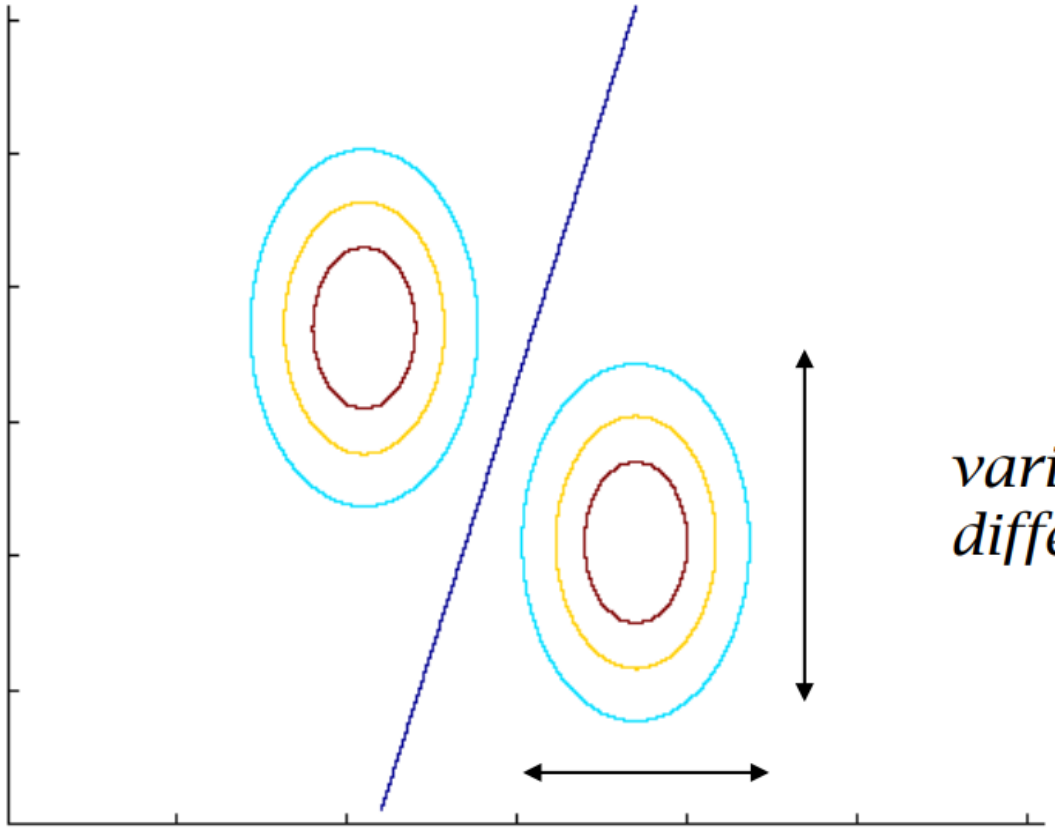
- Model the same as Gaussian Discriminative Analysis with diagonal covariance matrix
- Maximum likelihood estimate of parameters

$$\mu_{ik} = \frac{\sum_{n=1}^{N} \mathbb{1}[t^{(n)}=k] \cdot x_i^{(n)}}{\sum_{n=1}^{N} \mathbb{1}[t^{(n)}=k]}$$

$$\sigma_{ik}^2 = \frac{\sum_{n=1}^{N} \mathbb{1}[t^{(n)}=k] \cdot (x_i^{(n)}-\mu_{ik})^2}{\sum_{n=1}^{N} \mathbb{1}[t^{(n)}=k]}$$

Gaussian Naive Bayes

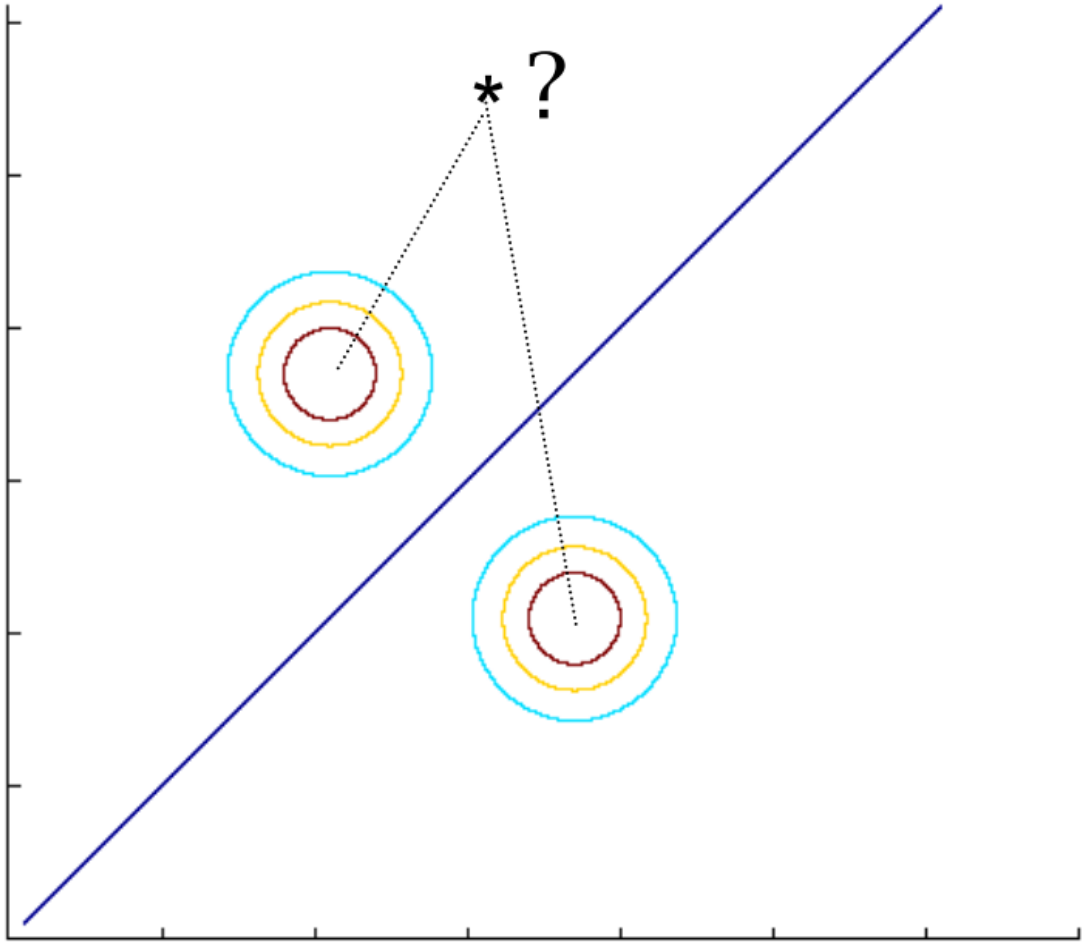
Decision Boundary: Shared Variances (between Classes)



variances may be different

Gaussian Naive Bayes

Decision Boundary: isotropic



- Same variance across all classes and input dimensions, all class priors equal

Decision Boundary: isotropic

- In this case: $\sigma_{i,k} = \sigma$ (just one parameter), class priors equal (e.g., $p(t_k) = 0.5$ for 2-class case)
- Going back to class posterior for GDA:

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

where we take $\Sigma_k = \sigma^2 I$ and ignore terms that don't depend on k (don't matter when we take max over classes):

$$\log p(t_k|\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\mu_k)^T(\mathbf{x}-\mu_k)$$

Example

Naïve Bayes vs. Logistic Regression

Set of Gaussian Naïve Bayes parameters (feature variance independent of class label)

⟺

Set of Logistic Regression parameters

- Representation equivalence
  - **But only in a special case!!!** (GNB with class-independent variances)
- But what's the difference???
- **LR makes no assumption about P(X|Y) in learning!!!**
- Loss function!!!
  - Optimize different functions! Obtain different solutions

slide by Aarti Singh & Barnabás

Example

Naïve Bayes vs. Logistic Regression

Verdict

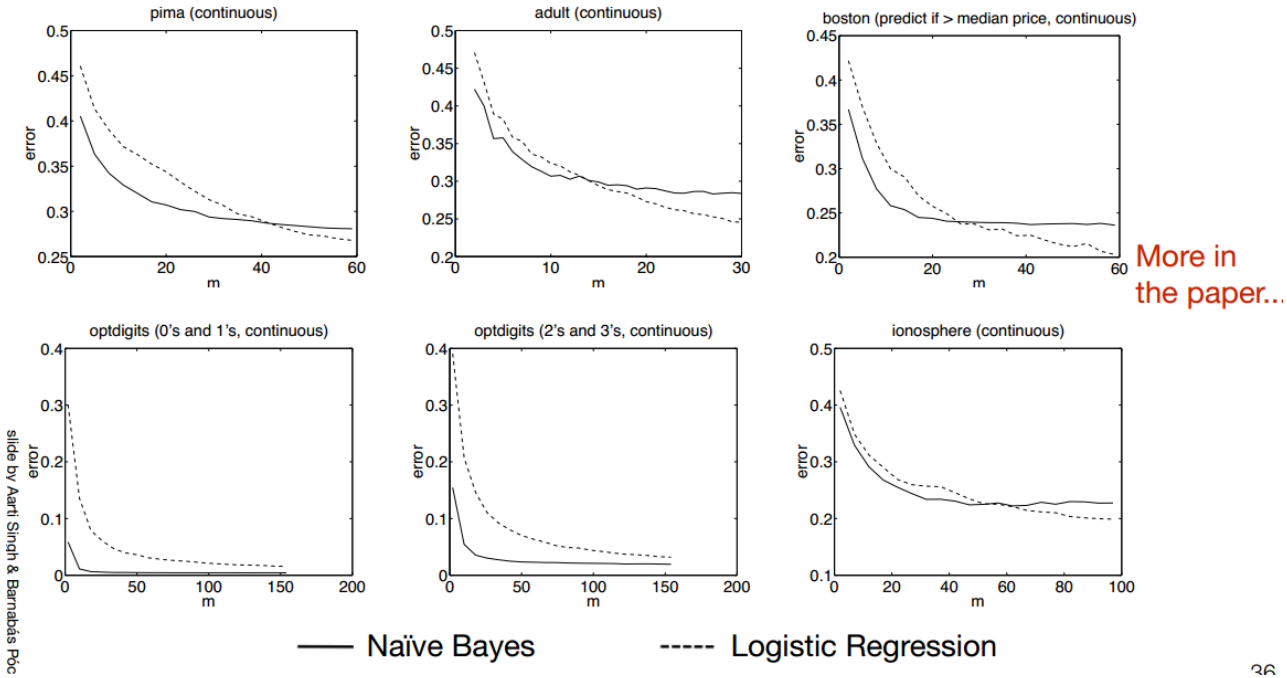
Both learn a linear decision boundary.
Naïve Bayes makes more restrictive assumptions and has higher asymptotic error,
BUT
converges faster to its less accurate asymptotic error.

slide by Aarti Singh & Barnabás Póczz

Example

Experimental Comparison (Ng-Jordan'01)

UCI Machine Learning Repository 15 datasets, 8 continuous features, 7 discrete features



More in the paper...

slide by Aarti Singh & Barnabás Póc

Summary

- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by maximizing conditional likelihood
  - no closed-form solution
  - concave ! global optimum with gradient ascent
- Gaussian Naïve Bayes with class-independent variances representationally equivalent to LR
  - Solution differs because of objective (loss) function
- In general, NB and LR make different assumptions
  - NB: Features independent given class! assumption on P($\mathbf{X}$|Y)
  - LR: Functional form of P(Y|$\mathbf{X}$), no assumption on P($\mathbf{X}$|Y)
- Convergence rates
  - GNB (usually) needs less data
  - LR (usually) gets to better solutions in the limit

slide by Aarti Singh & Barnabás Póczos

When to Use Naive Bayes

Because naive Bayesian classifiers make such stringent assumptions about data, they will generally not perform as well as a more complicated model. That said, they have several advantages:

They are extremely fast for both training and prediction

- They provide straightforward probabilistic prediction
- They are often very easily interpretable
- They have very few (if any) tunable parameters These advantages mean a naive Bayesian classifier is often a good choice as an initial baseline classification. If it performs suitably, then congratulations: you have a very fast, very interpretable classifier for your problem. If it does not perform well, then you can begin exploring more sophisticated models, with some baseline knowledge of how well they should perform.

Naive Bayes classifiers tend to perform especially well in one of the following situations:

- When the naive assumptions actually match the data (very rare in practice)
- For very well-separated categories, when model complexity is less important
- For very high-dimensional data, when model complexity is less important

The last two points seem distinct, but they actually are related: as the dimension of a dataset grows, it is much less likely for any two points to be found close together (after all, they must be close in every single dimension to be close overall). This means that clusters in high dimensions tend to be more separated, on average, than clusters in low dimensions, assuming the new dimensions actually add information. For this reason, simplistic classifiers like naive Bayes tend to work as well or better than more complicated classifiers as the dimensionality grows: once you have enough data, even a simple model can be very powerful.