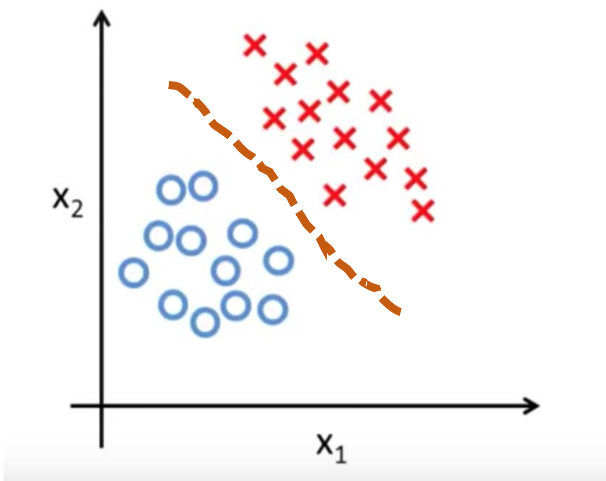
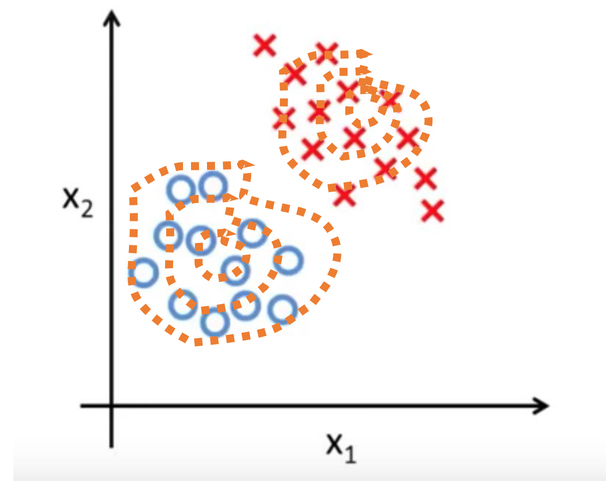
Gaussian Bayes Classifiers
(Gaussian Discriminant Analysis)

CSC411/2515: Machine Learning and Data Mining, Winter 2018

Michael Guerzhoy and Lisa Zhang

Discriminative vs Generative Models





Build a classifier that learns a decision boundary

Models $p(y|x)$ directly

Build a model of what data in each class looks like

Models $p(x|y)$ for each value of y and $p(y)$ and then use Bayes' rule to find $p(y|x)$

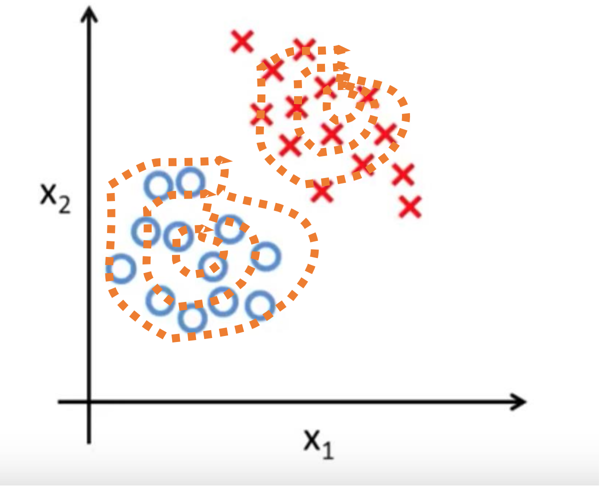
Discriminative vs Generative Models

Last Class:

- Single binary feature
- Single Gaussian feature
- Many binary features (Naïve Bayes)

This Hour:

- Many Gaussian features
    - Gaussian Bayes Classifier
    - (aka) Gaussian Discriminant Analysis



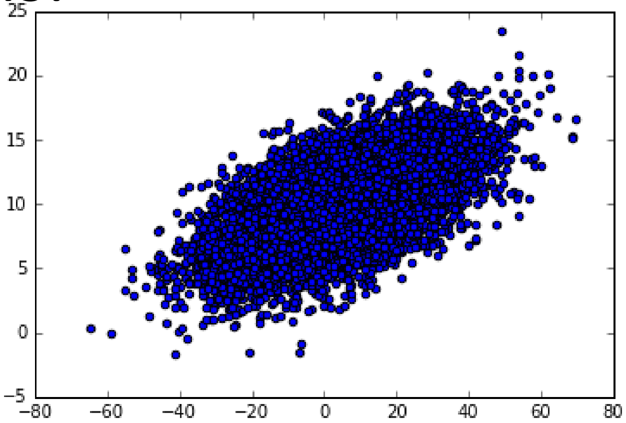
But first, let's review the Multivariate Gaussian

Multivariate Gaussian: a quick intro (1)

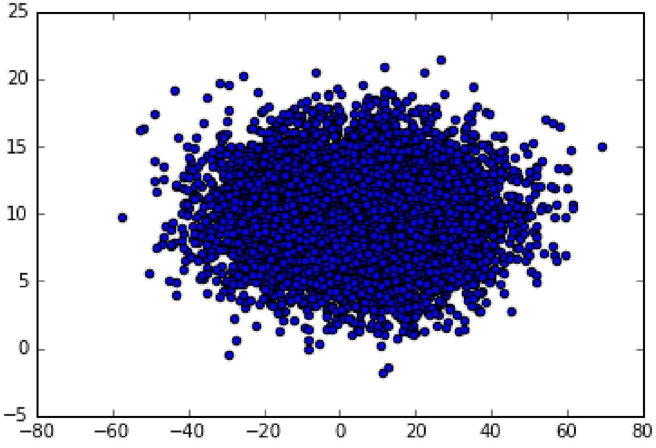
- Consider $\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \sim \begin{pmatrix} N(\mu_1, \sigma_1^2) \\ N(\mu_2, \sigma_2^2) \\ \dots \\ N(\mu_n, \sigma_n^2) \end{pmatrix}$
- Here, we are sampling an n-dimensional point, with every dimension sampled independently
- If we sample a lot of points, we'll get something that looks like a cloud, where large σ_k means that the cloud is "wider" along dimension k
- The cloud will be "axis-aligned," in the sense that it won't be tilted.

Multivariate Gaussian: a quick intro (2)

- The cloud will not look like this:



- But it could look like this:



Multivariate Gaussian: a quick intro (3)

- The mathematical way to describe an "axis-aligned" cloud is to say
  - $Cov(x_i, x_j) = 0$ for $i \neq j$
  - I.e., the coordinates along axes i and j are uncorrelated
- A multivariate Gaussian distribution allows as to specify the covariances between coordinates along axes *i and j.*
  - *Reminder:* $Cov(x_1, x_2) = E[(x_2 - \mu_1)(x_2 - \mu_2)]$
    $$\approx \frac{1}{N}\sum(x_1^{(i)} - \overline{x_1})(x_2^{(i)} - \overline{x_2})$$

Multivariate Gaussian: a quick intro (4)

- Specify the covariance matrix Σ:

$$\Sigma = \begin{pmatrix} Cov(x_1, x_1) & \dots & Cov(x_1, x_n) \\ \dots & \dots & \dots \\ Cov(x_n, x_1) & \dots & Cov(x_n, x_n) \end{pmatrix}$$

- We can have a multivariate Gaussian distribution that's specified by

$$X \sim N(\mu, \Sigma)$$

- It generates a cloud of points, but this time the coordinates might be correlated

Multivariate Gaussian: a quick intro (5)

- Suppose $X \sim \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & .2 \\ .2 & 1 \end{pmatrix} \right)$
- That means that
  - $Var(x_1) = Var(x_2) = Cov(x_1, x_1) = Cov(x_2, x_2) = 1$
  - $Cov(x_1, x_2) = .2$
- The larger x_1, the larger we expect x_2 to be

Multivariate Gaussian: a quick intro (6)

- The density of the multivariate Gaussian:
  $$f(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\ )$$
- k is the dimensionality of x (so $\dim(\Sigma) = k \times k$)
- $|\Sigma| = \det(\Sigma)$

Learning a Gaussian

- We observe a bunch of points $D = \{x^{(1)}, x^{(2)}, \dots\}$
- We assume that they were all generated by a single (multivariate) Gaussian
- We can learn it using maximum likelihood: maximize the probability $P(D|\theta)$ that the data was generated using a Gaussian parameterized by $\theta = \{\mu, \Sigma\}$.
- We can show (using calculus) that the ML estimates are:

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}, \hat{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \bar{x}\right)\left(x^{(i)} - \bar{x}\right)^{T}$$

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}, \hat{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \bar{x}\right)\left(x^{(i)} - \bar{x}\right)^T$$

- $\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$ makes sense: the mean of the Gaussian is the mean of the vectors $x^{(i)}$
- The (k, n)-th component of $\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \bar{x}\right)\left(x^{(i)} - \bar{x}\right)^T$ is the estimated $Cov(x_k, x_n)$, $\frac{1}{m}\sum_{i=1}^{m}\left(x_k^{(i)} - \overline{x_k}\right)\left(x_n^{(i)} - \overline{x_n}\right)$

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

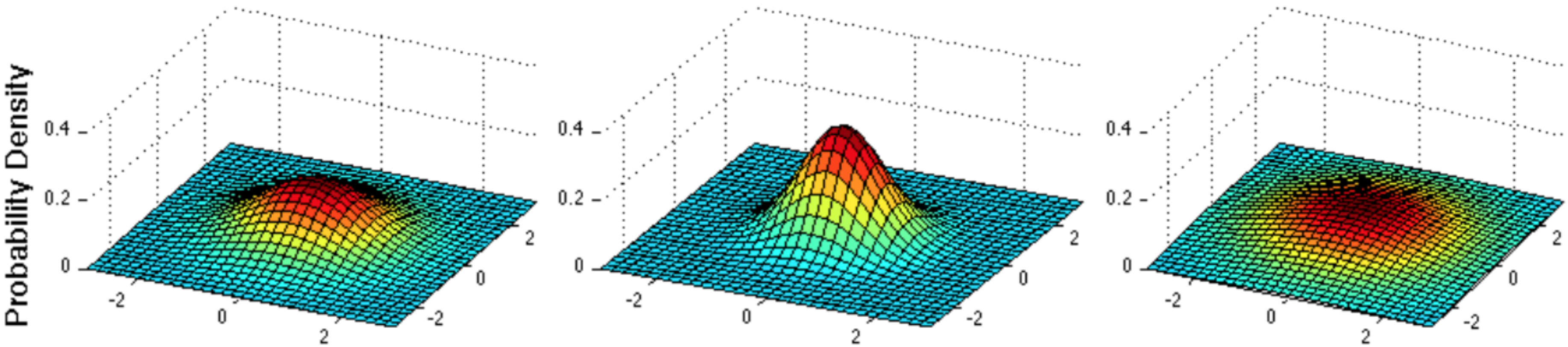


Figure: Probability density function

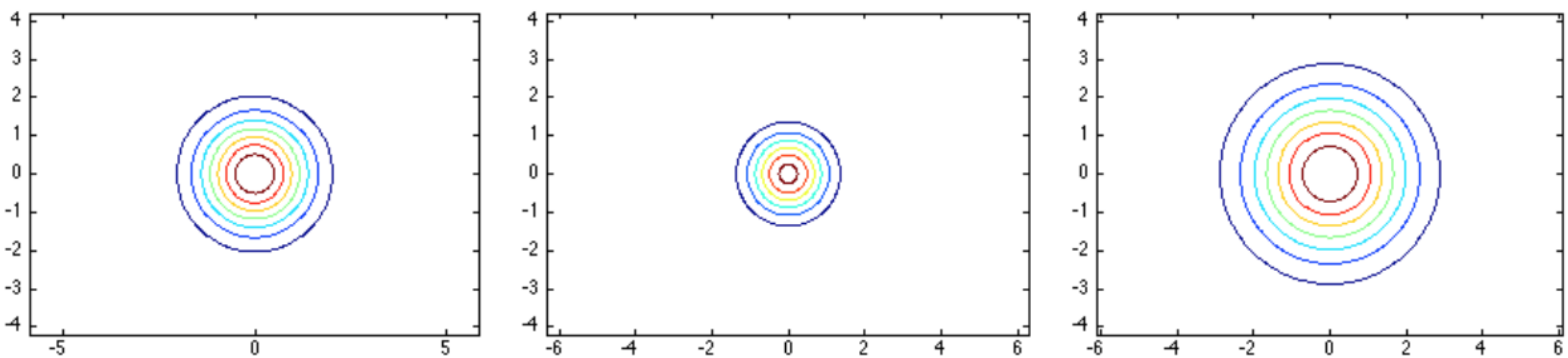


Figure: Contour plot of the pdf

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

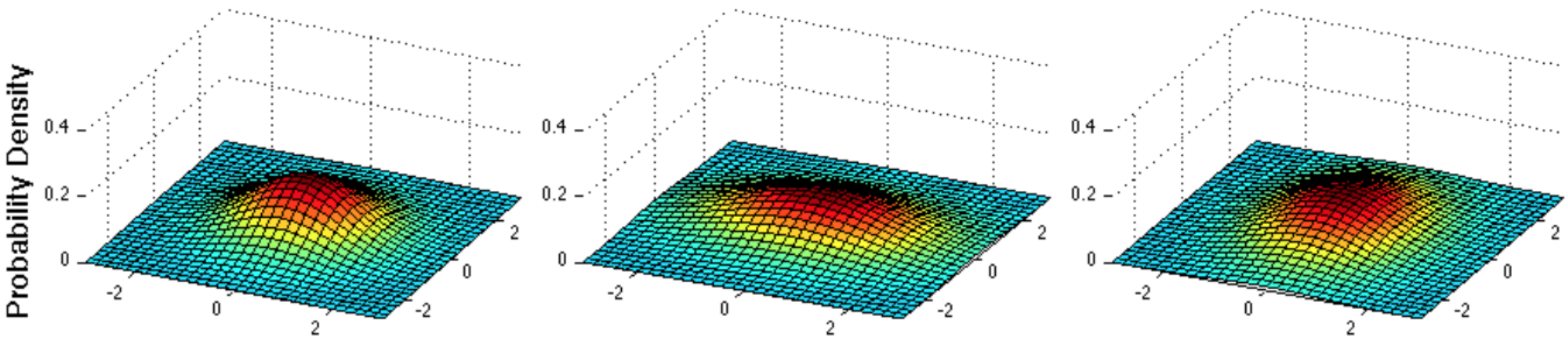


Figure: Probability density function

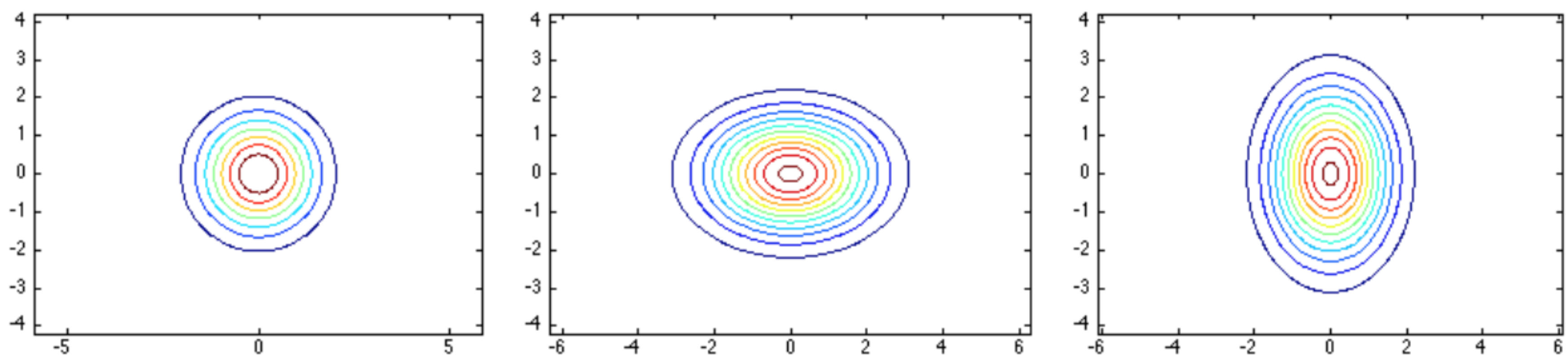


Figure: Contour plot of the pdf

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

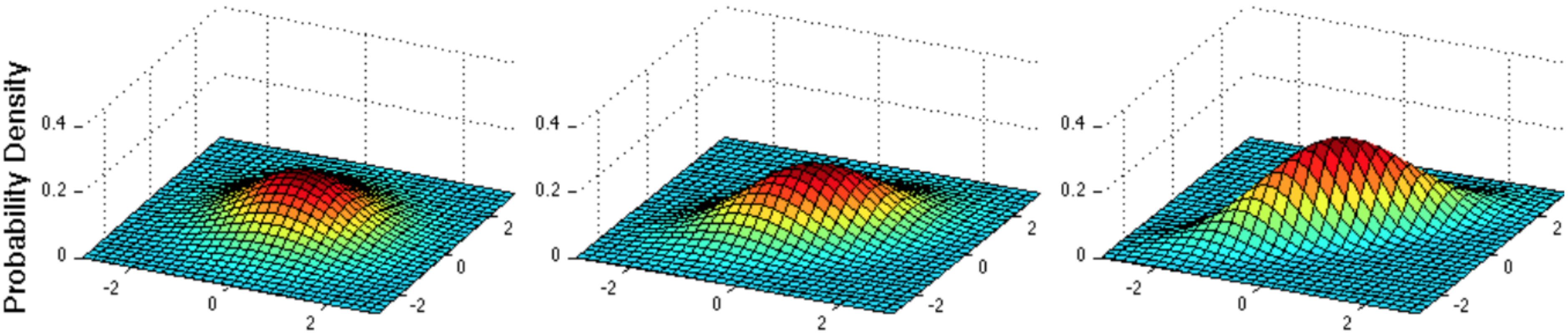


Figure: Probability density function

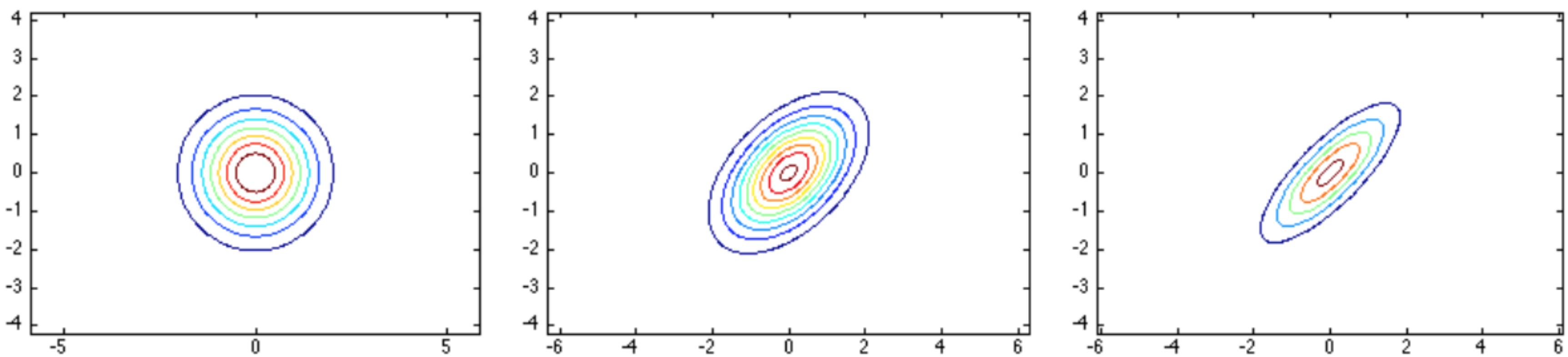


Figure: Contour plot of the pdf

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

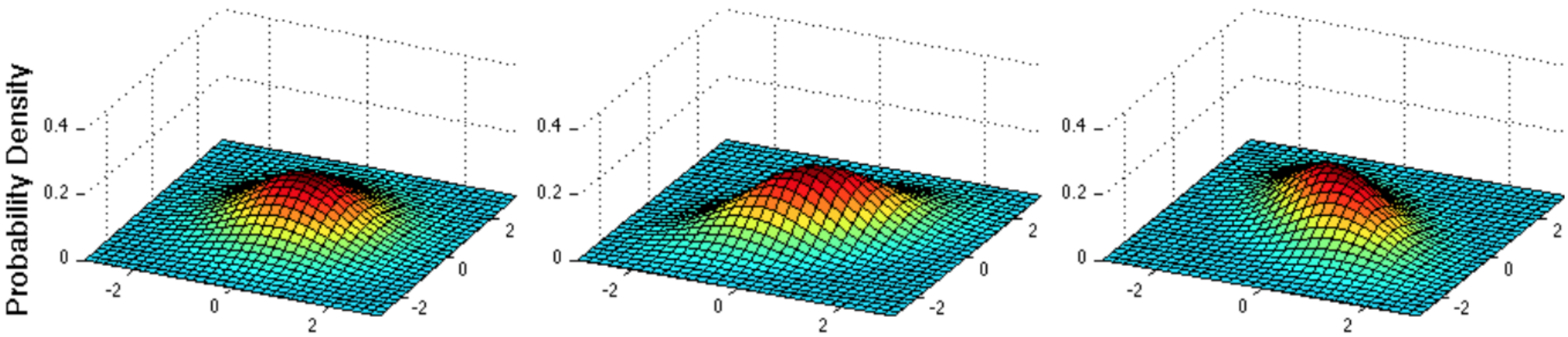


Figure: Probability density function

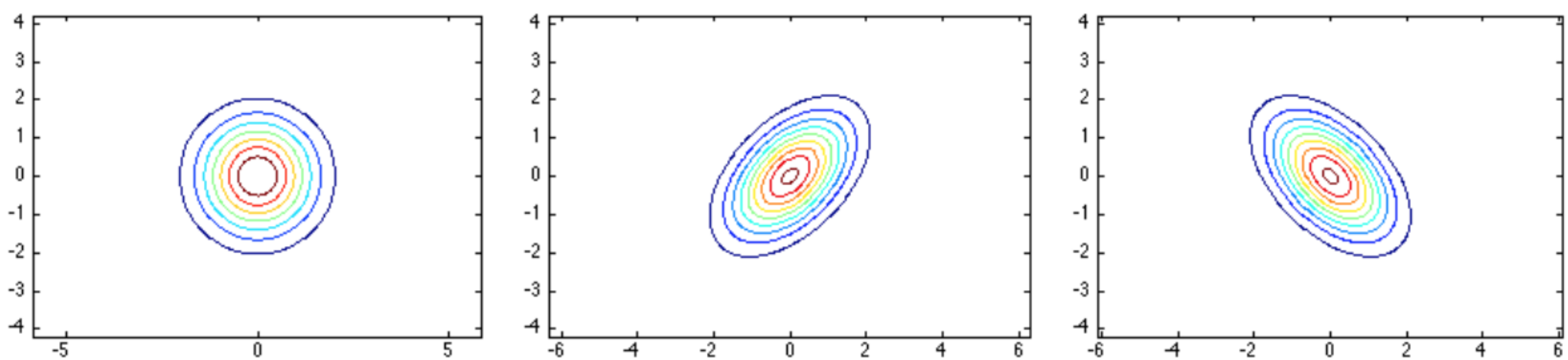


Figure: Contour plot of the pdf

Gaussian Bayes Classifier

- Classifier:

$$P(y = c|\boldsymbol{x}) = \frac{P(y = c)P(\boldsymbol{x}|y = c)}{\sum_{c'} P(y = c')P(\boldsymbol{x}|y = c')}$$

- Just like before:

$$\begin{aligned} c_{MAP} &= argmax_c P(c|\boldsymbol{x}) \\ &= argmax_c \frac{P(\boldsymbol{x}|c)P(c)}{P(\boldsymbol{x})} \\ &= argmax_c P(\boldsymbol{x}\,|c)P(c) \\ &= argmax_c f(\boldsymbol{x}; \mu_c, \Sigma_\mathrm{c})P(c) \end{aligned}$$

Gaussian Bayes Classifier

- Classifier:

$$P(y = c|\boldsymbol{x}) = \frac{P(y = c)P(\boldsymbol{x}|y = c)}{P(\boldsymbol{x})}$$

Recall:

$$P(\boldsymbol{x}|c) = \frac{1}{\sqrt{(2\pi)^k |\Sigma_\mathrm{c}|}} \exp(-\frac{1}{2}(\boldsymbol{x} - \mu_c)^T \Sigma_c^{-1}(x - \mu_c))$$

Then

$$\begin{aligned}
&\log P(c|\boldsymbol{x}) \\
&= \boxed{\log P(x|c)} + \log P(c) - \log P(x) \\
&= \boxed{-\frac{k}{2}\log(2\pi) - \frac{1}{2}\log\left|\Sigma_k^{-1}\right| - \frac{1}{2}(\boldsymbol{x} - \mu_c)^T \Sigma_c^{-1}(x - \mu_c)} \\
&\quad + \log P(c) - \log P(x)
\end{aligned}$$

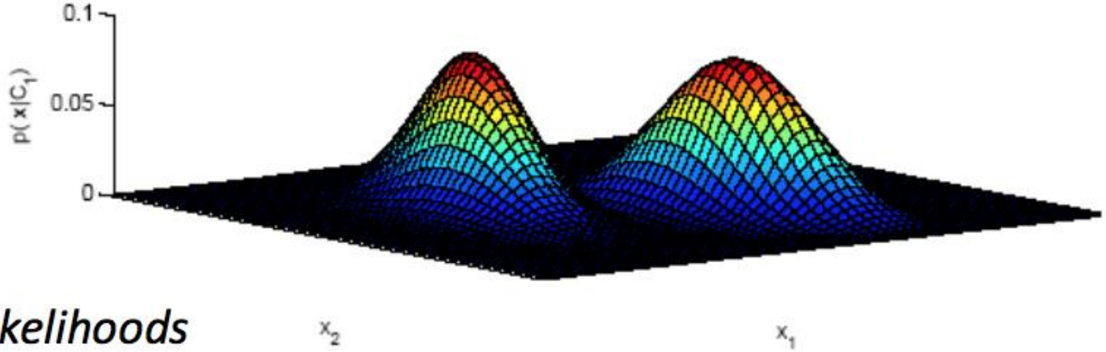
Gaussian Bayes Classifier

$$\log P(c|\boldsymbol{x})$$
$$= -\frac{k}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma_c^{-1}| - \frac{1}{2}(\boldsymbol{x} - \mu_c)^T\Sigma_c^{-1}(\boldsymbol{x} - \mu_c)$$
$$+ \log P(c) - \log P(x)$$

Decision Boundary:

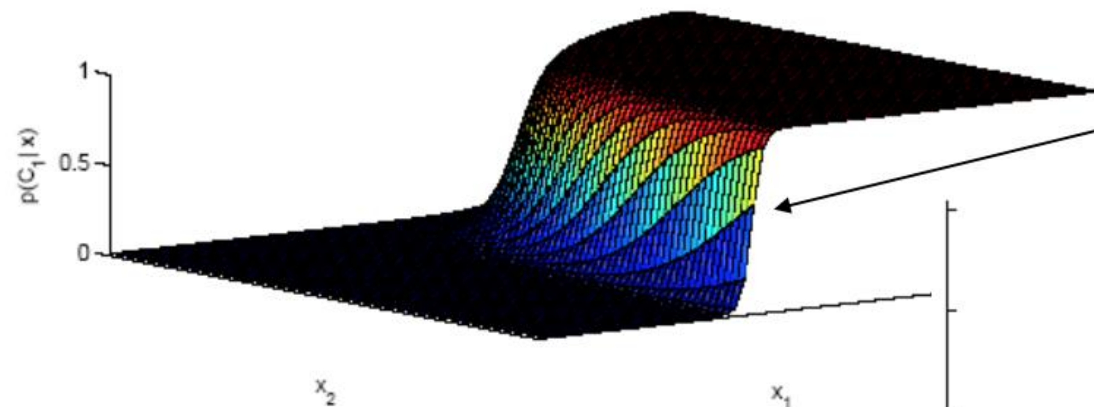
$$\frac{1}{2}(\boldsymbol{x} - \mu_c)^T\Sigma_c^{-1}(\boldsymbol{x} - \mu_c) = \frac{1}{2}(\boldsymbol{x} - \mu_{c\prime})^T\Sigma_{c\prime}^{-1}(\boldsymbol{x} - \mu_c{}') + CONST$$

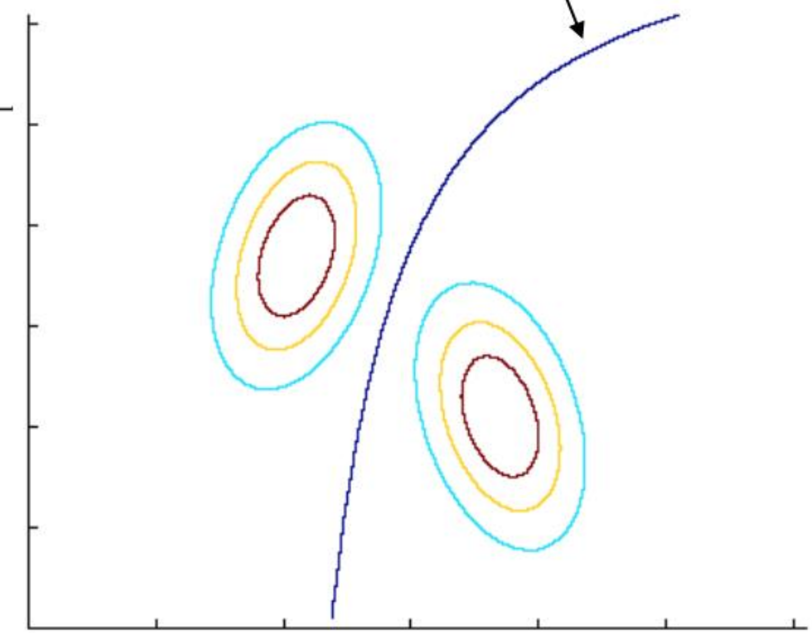
This is quadratic in terms of **x**!



likelihoods

discriminant:
$P\ (c_1|\boldsymbol{x}) = 0.5$





posterior for c_1

Learning

- Learn each Gaussian (as in this lecture)
- Learn $P(c)$ with the class as a Bernoulli variables (as for Naïve Bayes)

- **Simplification**:
  - If x is too high-dimensional, covariance matrix has many parameters
  - Can save parameters by using a shared covariance for both classes
  - In this case, the decision boundary is **linear**
    - Why? Set $P(x|c=0) = P(x|c=1)$, and see that the quadratic terms cancel out

Comparison to Logistic Regression

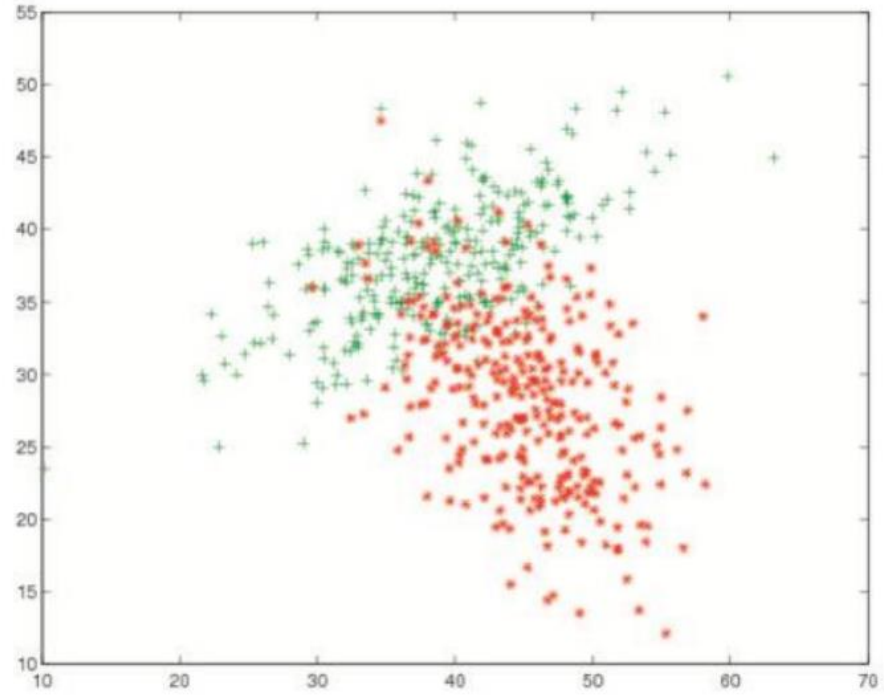
- We can show (analogously to what we did with Naïve Bayes) that if we share the covariance matrix between all classes, we retrieve the same form as logistic regression

$$\log \frac{P(y = c | x_1, \dots, x_p)}{P(y = c' | x_1, \dots, x_p)} = \beta_0 + \sum_j \beta_j x_j$$

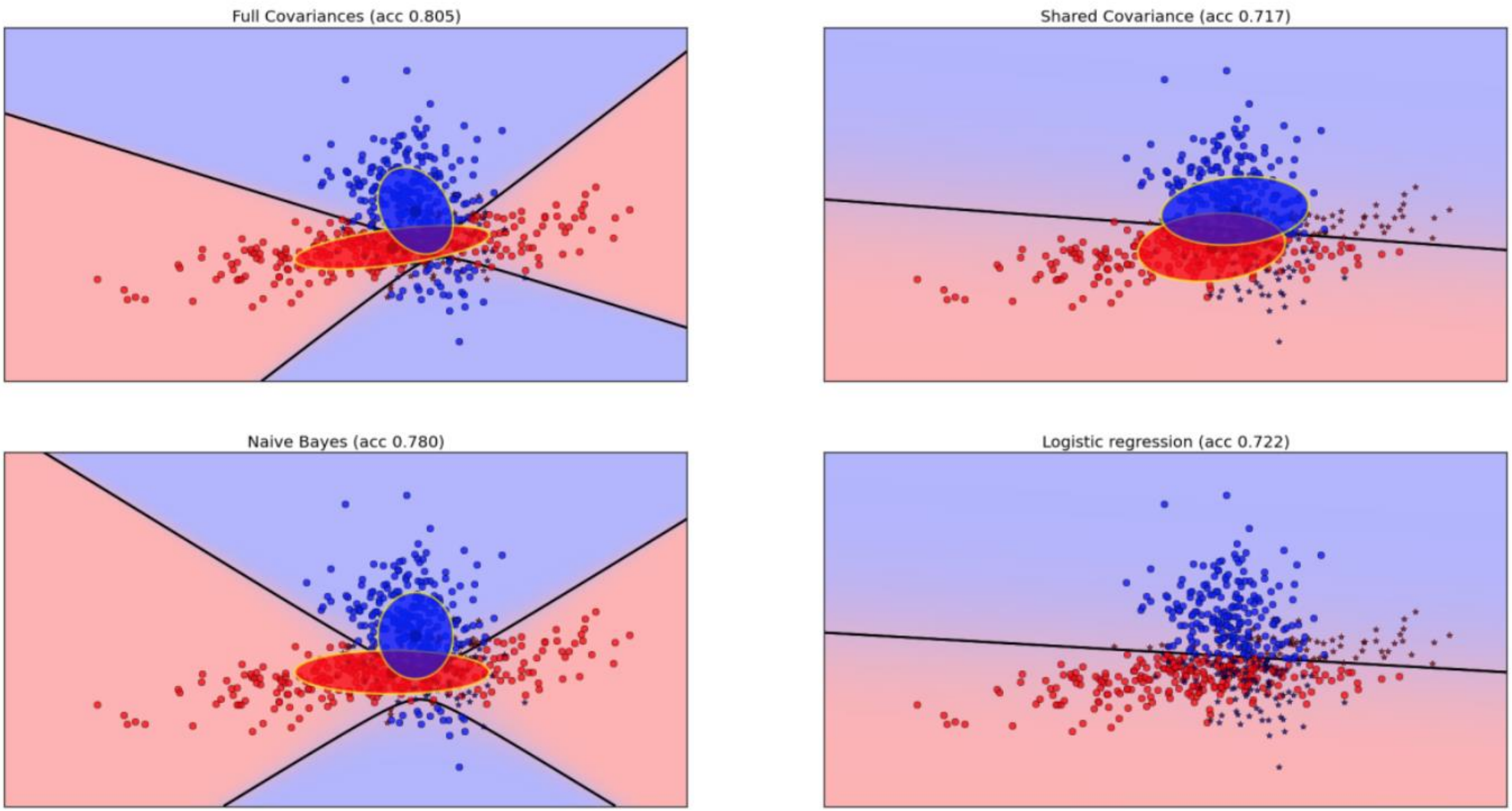
- **BUT** Gaussian Bayes makes stronger assumptions
  - Class-conditional data is multivariate Gaussian
  - If true, Gaussian Bayes is better
  - Logistic regression is more robust
    - If the model is not exactly correct, the outputs of GDA will make less sense

Example

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



Example


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