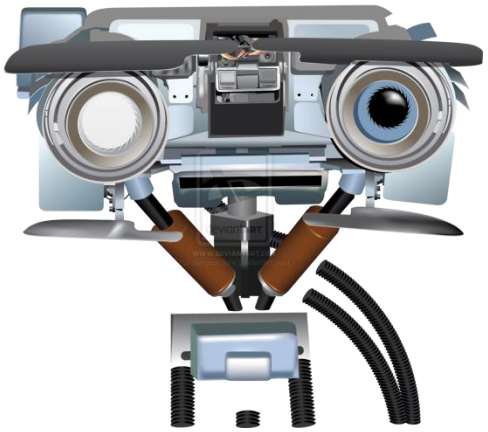
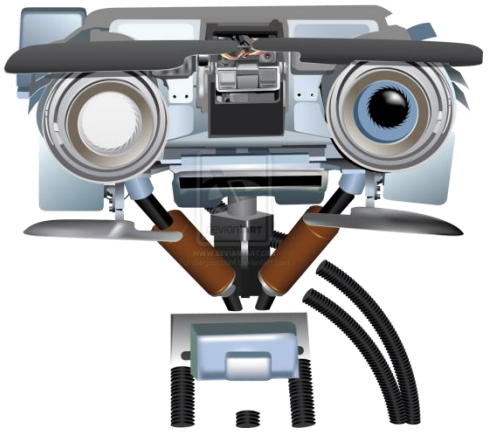


Announcements

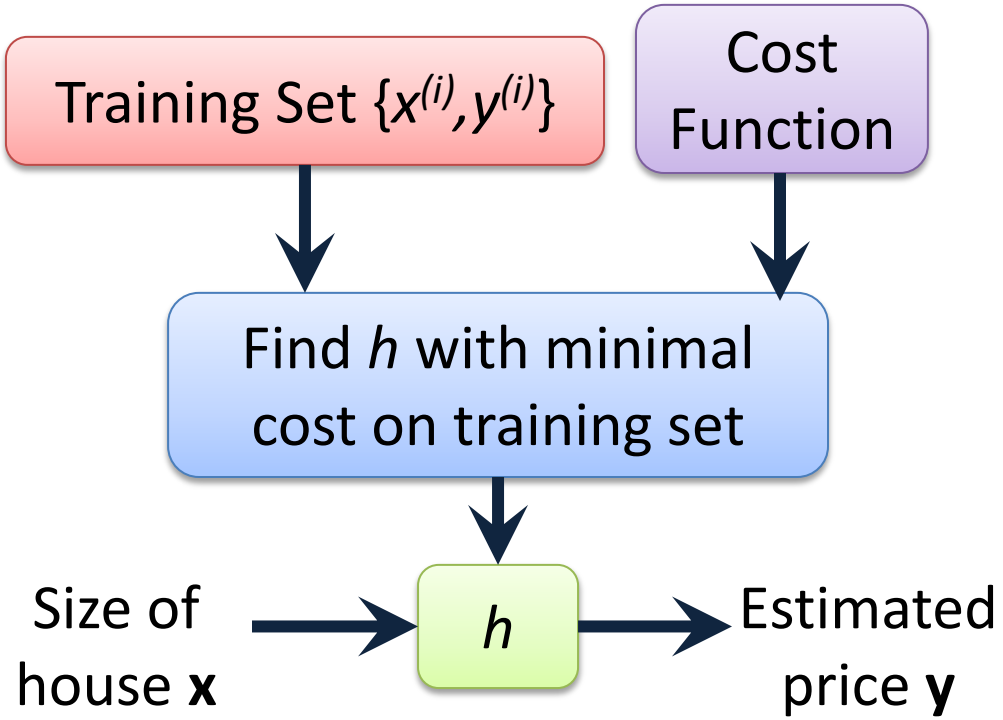
- ps 1 due today
- I will go over Quiz0 after class
- ps2 will be released today



Last time: Linear Regression

Recap

Supervised Learning



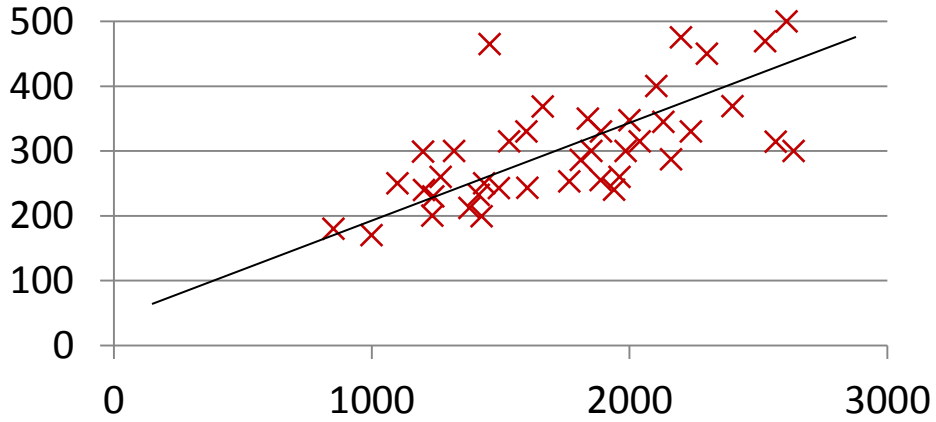
Last Time: Maximum Likelihood Solution for Linear Regression

Hypothesis:

$h_\theta(x) = \theta^T x$

θ: parameters

$D = \left(x^{(i)}, y^{(i)}\right)$: data



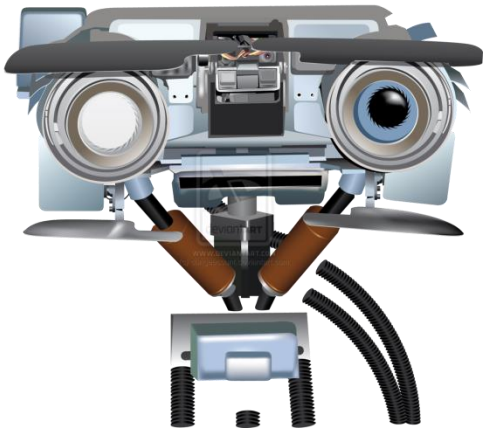
Likelihood:

$$p(\boldsymbol{t}|\boldsymbol{x}, \theta, \beta) = \prod_{i=1}^{m} N(t^{(i)}|h_\theta\left(x^{(i)}\right), \beta^{-1})$$

Goal: maximize likelihood, equiv. to

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - t^{(i)}\right)^2$$

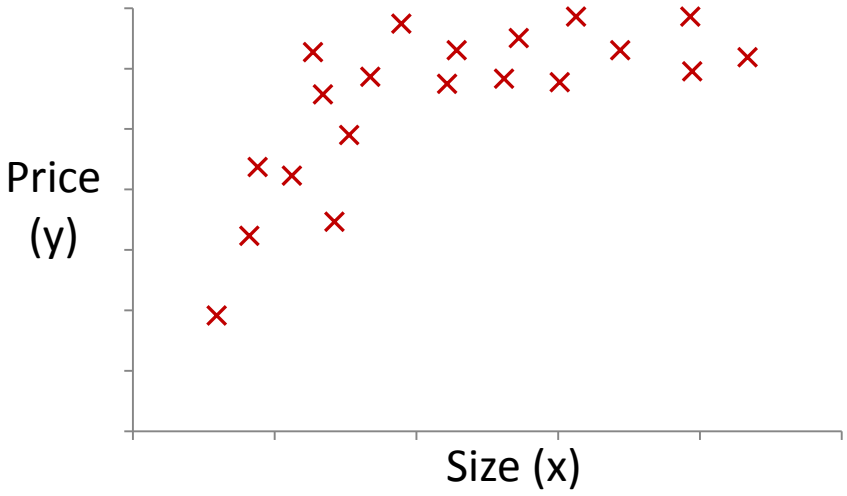
(same as minimizing SSE)



Linear Regression

Nonlinear Feature Mapping

Nonlinear Regression



Polynomial

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

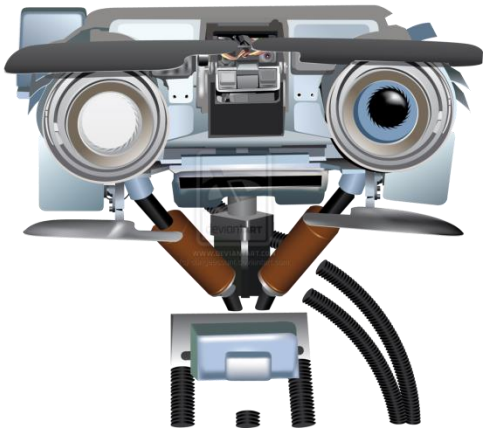
$$= \theta_0 + \theta_1(size) + \theta_2(size)^2 + \theta_3(size)^3$$

Other

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$$

Nonlinear feature transformation

- replace with functions of original features
- hypothesis is still linear in the new feature space
- hypothesis is nonlinear in original feature space



Linear Regression

Normal Equations

Direct solution

Want to minimize SSD:

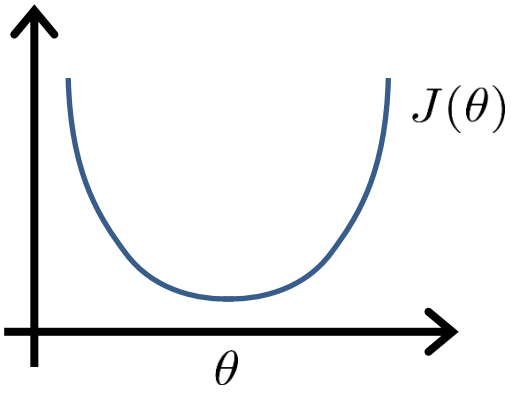
$$J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Find minima of function:

$\theta \in \mathbb{R}^{n+1}$

$\frac{\partial}{\partial \theta_j} J(\theta) = \cdots = 0$ (for every j)

Solve for $\theta_0, \theta_1, \ldots, \theta_n$



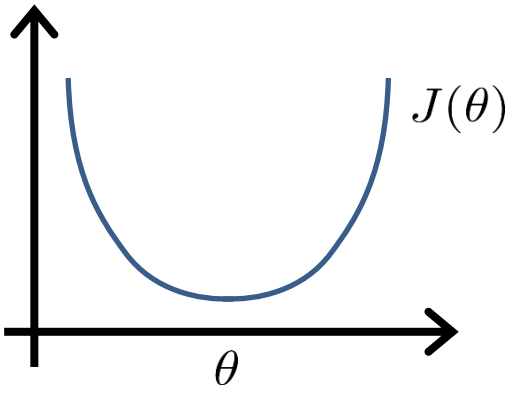
Direct solution

Re-write SSD using vector-matrix notation:

$$J(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y)$$

Where:

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$



Solution: Normal Equation

$$\Theta = (X^T X)^{-1} X^T y$$

Derivation of Normal Equations

- SSE in matrix form:

$$\mathrm{J}(\theta) = \frac{1}{2m}(X\theta - y)^T(X\theta - y) =$$

$$= \frac{1}{2m}\{\theta^T\{X^T X\}\,\theta - 2\{X^T y\}^T\theta + const\}$$

- Take derivative with respect to θ, set to 0

$$\frac{\partial J}{\partial \theta} \propto X^T X\,\theta - X^T y = 0$$

$$\theta^{ML} = (X^T X)^{-1}\ X^T y$$

Derivative w.r.t. vector!

- Also known as the least mean squares, or least squares solution

Example: $m = 4.$

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

Design Matrix

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

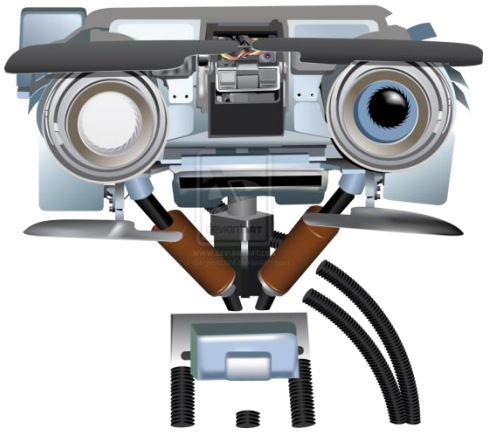
Normal Equation

$$\theta = (X^T X)^{-1} X^T y$$

Trade-offs

m training examples, n features.

Gradient Descent	Normal Equation
• Need to choose α. • Needs many iterations. • Works well even when n is large.	• No need to choose α. • Don't need to iterate. • Need to compute $(X^T X)^{-1}$ • Slow if n is very large.



Classification

Logistic Regression

Ng Video Lectures

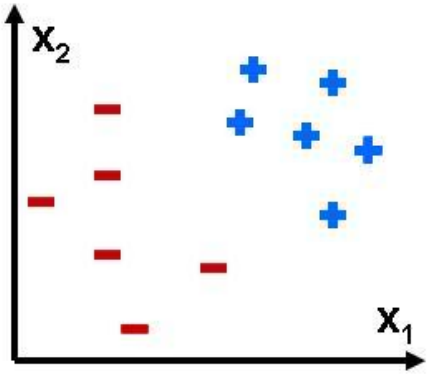
- Logistic Regression
- Please watch the videos
- I will only recap some points
- You are responsible for all assigned material

Classification

Email: Spam / Not Spam?

Online Transactions: Fraudulent (Yes / No)?

Tumor: Malignant / Benign ?



$$y \in \{0, 1\}$$

0: “Negative Class” (e.g., benign tumor)

1: “Positive Class” (e.g., malignant tumor)

SSE for classification?

- Why not use the least-squares cost for classification, as we did for regression?
- Indeed, it is possible!

  let $y^{(i)}$ be the label of i^{th} training sample $x^{(i)}$

  $y^{(i)}(k) = 1$ if the it belongs to class k, and 0 otherwise

  for example, $y^{(i)} = [0 \quad 1]^T$

  Then, apply least squares as before

  (see Bishop 4.1.3 for more details if interested)
- However, this is less robust than logistic regression…

Least Squares vs. Logistic Regression for Classification

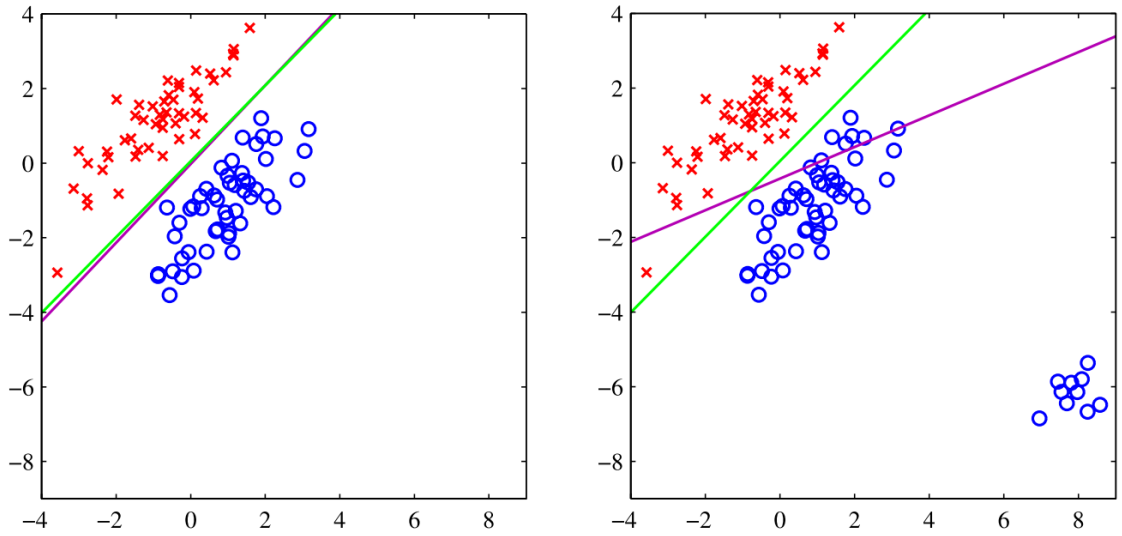


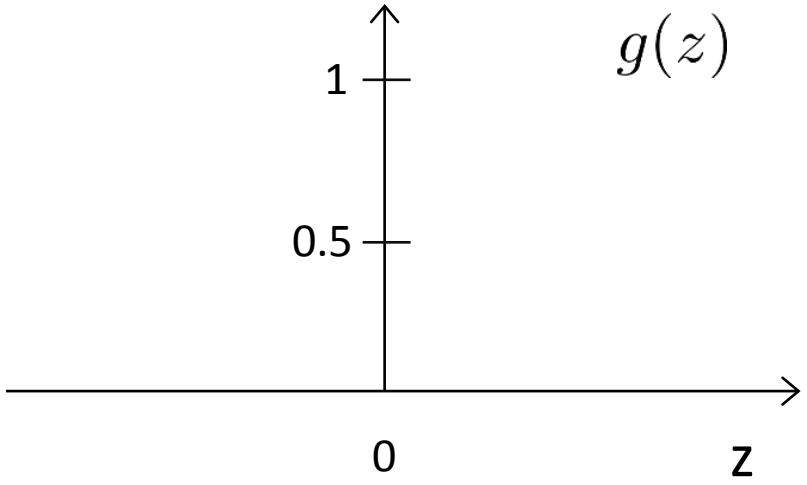
Figure 4.4 from Bishop. The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve). The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Logistic Regression

$$0 \leq h_\theta(x) \leq 1$$

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$g(z) = \frac{1}{1 + e^{-z}}$$



$$p(y = 1|x) = \frac{1}{1 + e^{-\theta^T x}}$$

Predict "$y = 1$" if $h_\theta(x) \geq 0.5$

predict "$y = 0$" if $h_\theta(x) < 0.5$

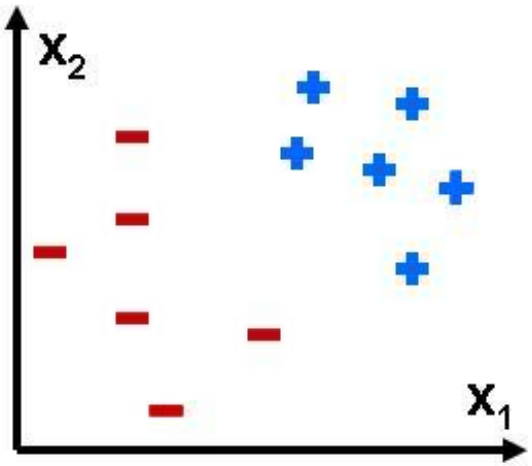
Logistic Regression

Hypothesis:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

θ: parameters

$D = \left(x^{(i)}, y^{(i)}\right)$: data



Cost Function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m} [\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Goal: minimize cost $\min_\theta J(\theta)$

Maximum Likelihood Derivation of Logistic Regression Cost

We can derive the Logistic Regression cost

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m} [\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

using Maximum Likelihood, and find its derivative w.r.t θ

No direct closed-form solution

Gradient descent

Cost

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$\}$

(simultaneously update all θ_j)

Gradient descent

Cost

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

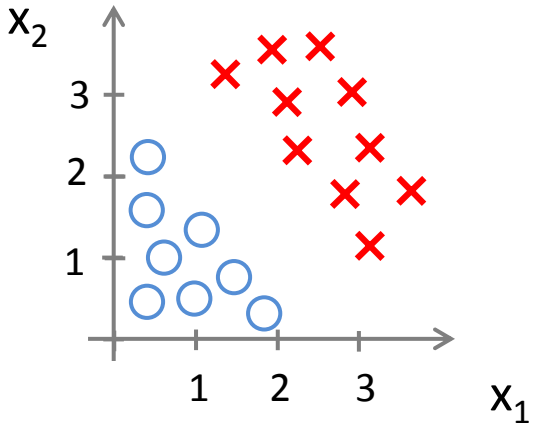
Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

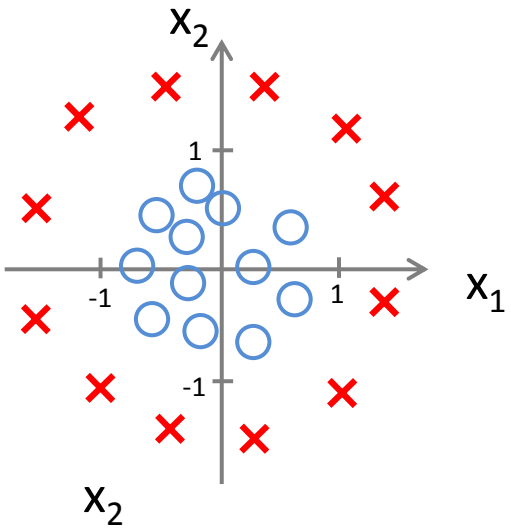
Decision boundary



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

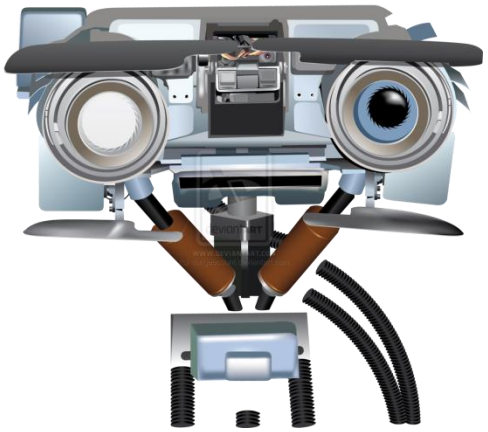
Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundaries



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$



Classification

Generative vs. Discriminative

Two approaches to classification

- We want to model conditional distribution, $p(C_k|x)$, $C_k \in \{0,1\}$, then assign label based on it[1]
- Two options
  - Discriminative: represent $p(C_k|x)$ as function of parameters θ, then learn θ from training data
  - Generative: use Bayes Rule to write

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

then learn parameters of class-conditional density $p(x|C_k)$ and class prior $p(C_k)$

[1]Can model output value directly (0/1), but having a probability is more useful

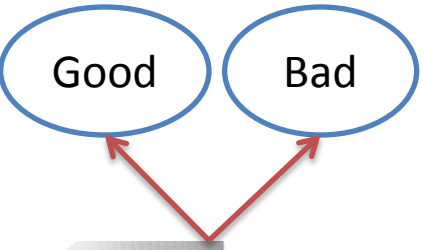
Generative vs Discriminative

some intuition…

Cookie Robots

- Suppose you own a cookie factory
- Want to detect bad cookies and discard them

Cookie Robots

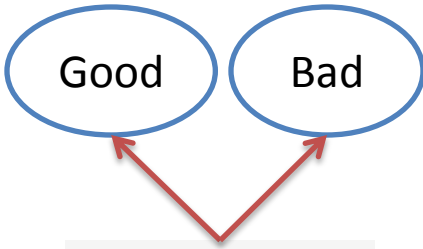


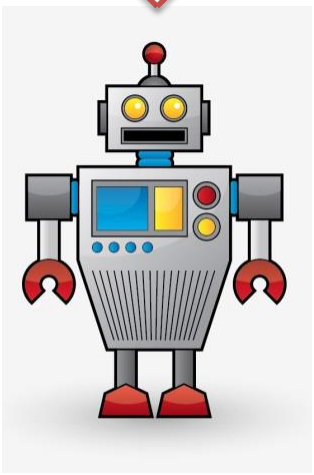
"The Chef"

- Can make good and bad cookies
- Compares new cookie to those
- Decides if it is good or bad

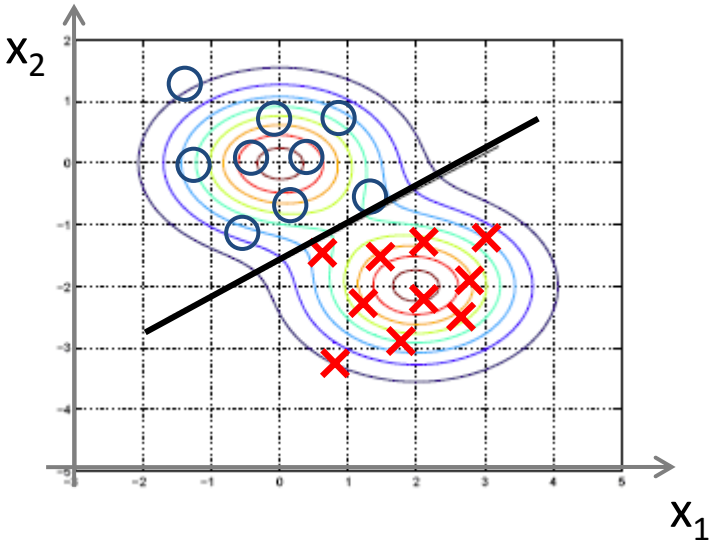
"The Critic"

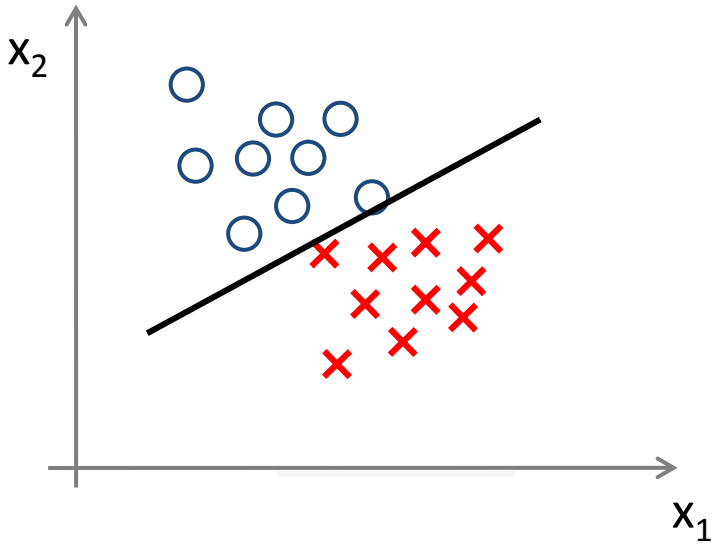
- Cannot make cookies
- Has seen lots of good and bad cookies
- Decides if it is good or bad





Generative vs Discriminative



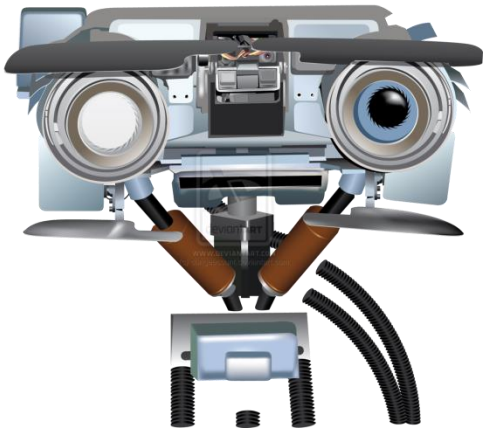


- Generative: model the class-conditional distribution of features, e.g. LDA

Can sample from distribution

- Discriminative: model the decision boundary directly, e.g. Logistic Regression, SVM

Cannot sample from distribution



Classification

Generalized Linear Models

Model $p(C_k|x)$ as sigmoid

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

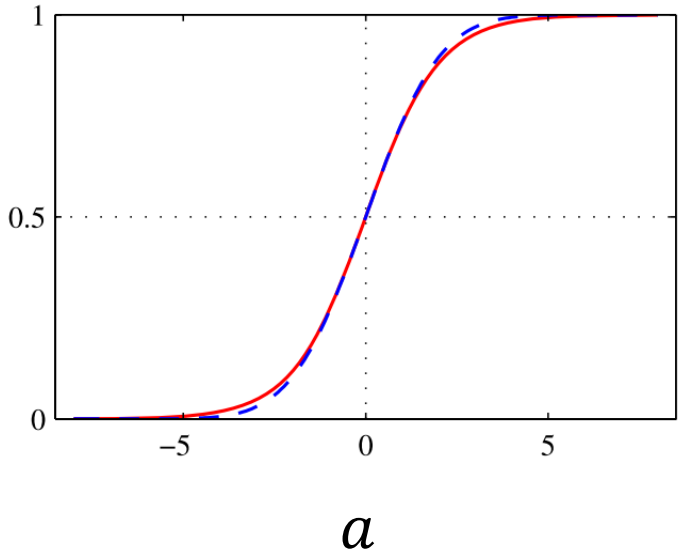
$$= \frac{1}{1 + \exp(-a)} = \sigma(a)$$

where we used

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

Sigmoid/Logistic: “Squashing” function

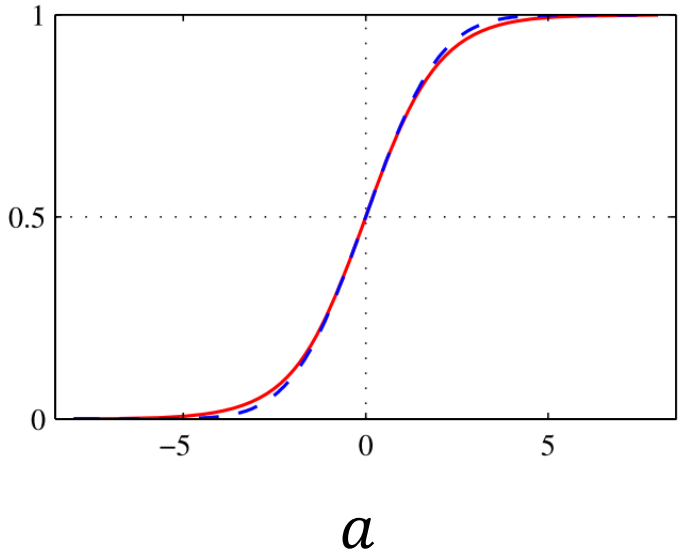
$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$



a(x): “Log-odds” or decision function

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$



When $a \geq 0,\ \sigma(a) \geq 0.5$

$a \geq 0$ is equivalent to $p(C_1|x) \geq p(C_2|x)$

Generalized Linear Models

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

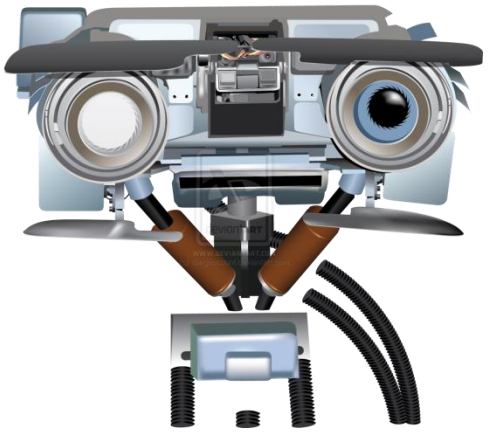
If a(x) is a linear function of x, then we have a generalized linear model:

Generative

- Use Gaussian distribution with same covariance for class-conditional density
- Leads to Linear Discriminant Analysis (next slide)

Discriminative

- Fit linear function $a(x) = \theta^T x$
- E.g., Logistic Regression



Classification

Linear Discriminant Analysis

Recall: Generative Model

- We want to model conditional distribution, $p(C_k|x)$,

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

and assign label C_1 if $p(C_1|x) > p(C_2|x)$, or, equivalently, the decision function $a \geq 0$, where

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

- Do this by learning parameters of class-conditional density $p(x|C_k)$ and class prior $p(C_k)$

Let's see an example…

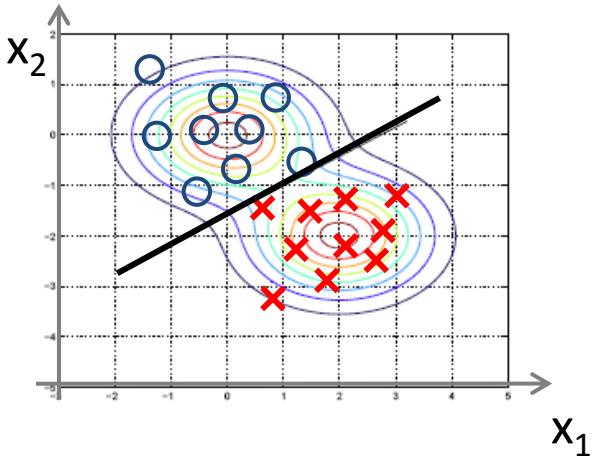
Class Prior

- What form should the class-conditional density $p(x|C_k)$ and prior $p(C_k)$ take?

- Assume:

$$p(C_k) = \pi_k,$$
$$\pi_k \geq 0,$$
$$\sum_{k=1}^{2} \pi_k = 1$$

Use Gaussian Class Density



- Assume that $p(x|C_k)$ is a Gaussian distribution for each class k:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

- p is the dimension of x and Σ_k is the covariance matrix. The vector x and the mean vector μ_k are both column vectors.

- Further, assume $\Sigma_k = \Sigma, \forall k$. By making this assumption, classifier becomes linear (proof to follow)

- This is called Linear discriminant analysis (LDA)

Decision Function

- picks a class that has the maximum posterior probability given the feature vector *X*

$$
\begin{aligned}
\hat{G}(x) &= \arg\max_k Pr(C = C_k | X = x) \\
&= \arg\max_k f_k(x)\pi_k \\
&= \arg\max_k \log(f_k(x)\pi_k) \\
&= \arg\max_k \left[-\log((2\pi)^{p/2}|\Sigma|^{1/2}) \right. \\
&\quad \left. -\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \right] \\
&= \arg\max_k \left[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) + \log(\pi_k) \right]
\end{aligned}
$$

Simplified Decision Function

- Note:

$$-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k - \frac{1}{2}x^T\Sigma^{-1}x$$

- After simplification we obtain this formula:

exercise

$$\hat{G}(x) = \arg\max_k \left[x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log(\pi_k)\right]$$

- Linear in x! (quadratic term dropped)

Simplified Decision Function

- Define the linear discriminant function

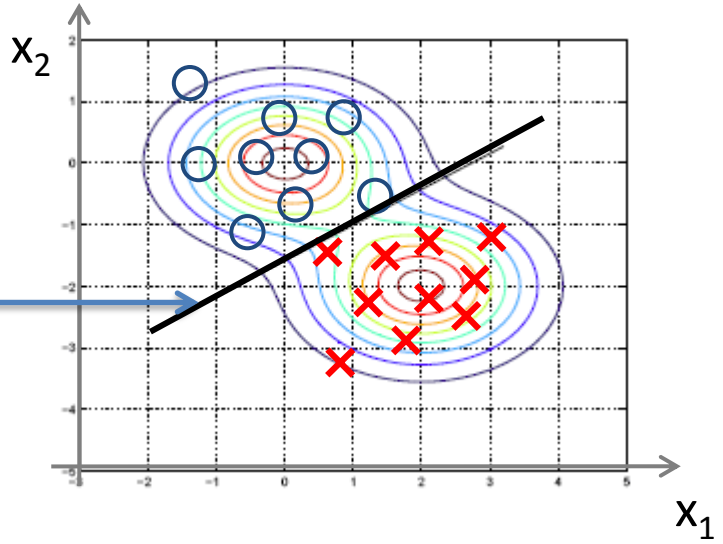
$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log(\pi_k)$$

- Then $\hat{G}(x) = \arg\max_k \delta_k(x)$
- The decision boundary between class *k* and *l* is: {x : δ_k (*x*) = δ_l(*x*)}, or equivalently, log-odds function must be 0:

$$\log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T\Sigma^{-1}(\mu_k - \mu_l) + x^T\Sigma^{-1}(\mu_k - \mu_l) = 0$$

Illustration of Decision Boundary

Can re-write as $\theta^T x + \theta_0$



$$\log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l) = 0$$

class prior log-ratio

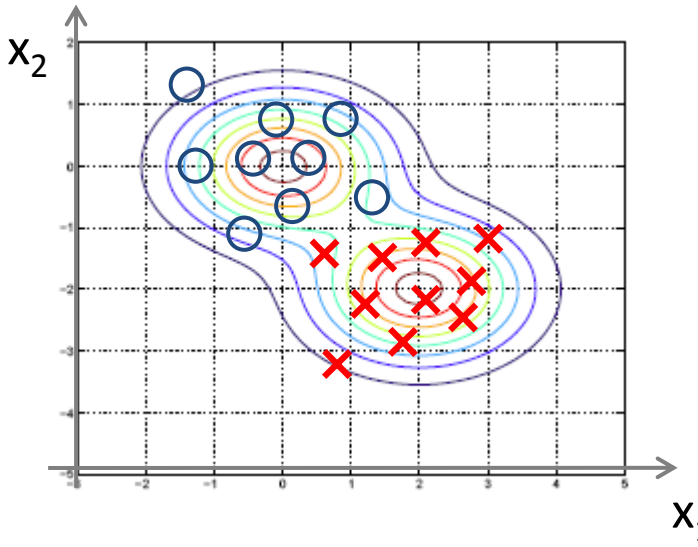
constant

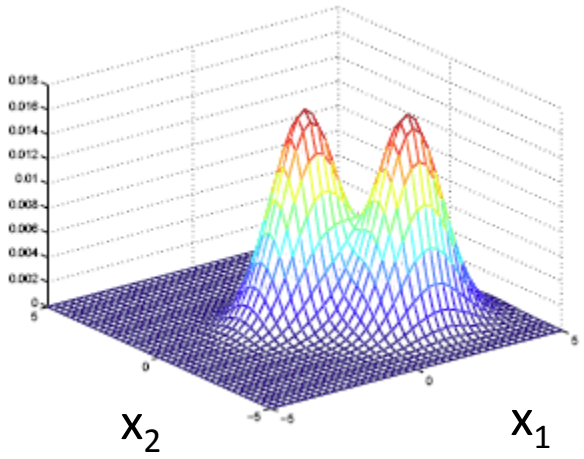
input

covariance

diff. in class means

Effect of Covariance Matrix





- covariance matrix determines the shape of the Gaussian density, so
- in LDA, the Gaussian densities for different classes have the same shape, but are shifted versions of each other (different mean vectors).

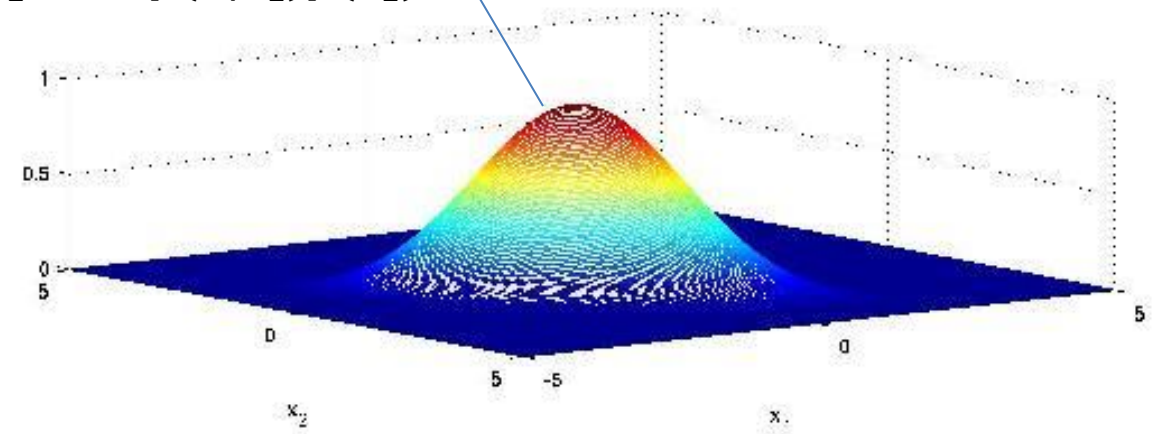
Effect of Class Prior

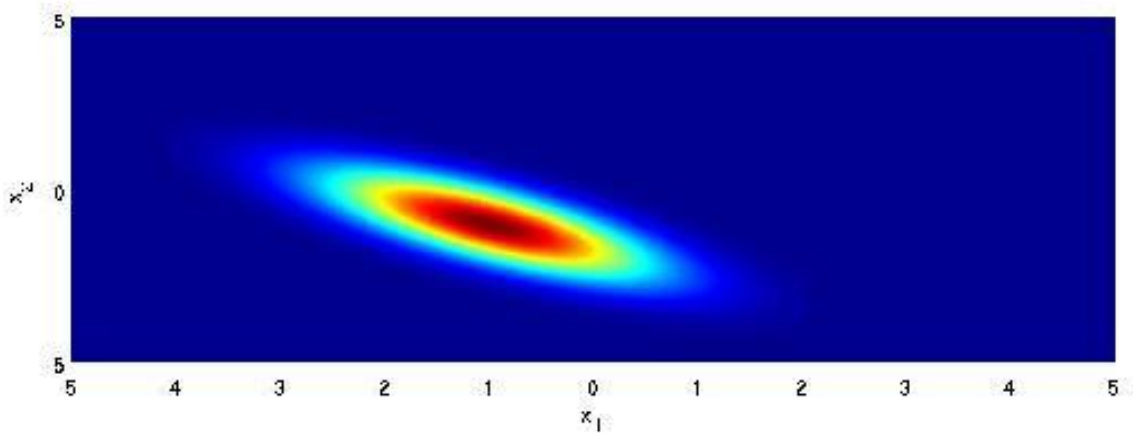
- What effect does the prior p(class), or π_k, have?
- Lets look at an example for 2 classes…

$$\log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l) = 0$$

class prior log-ratio

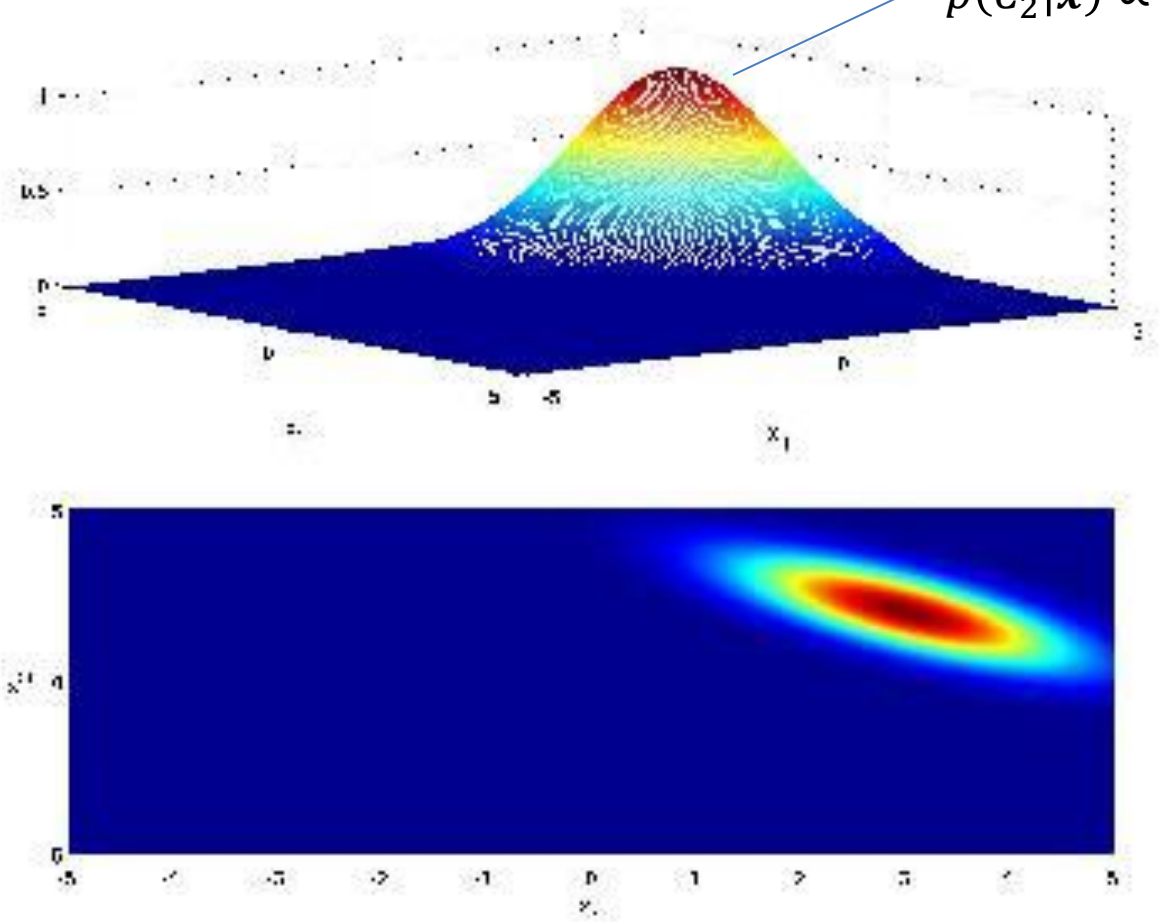
$$p(C_1|\boldsymbol{x}) \propto p(\boldsymbol{x}|C_1)p(C_1)$$





Model class-conditional probability of a 2D feature vector for class 1 as a multivariate Gaussian density.

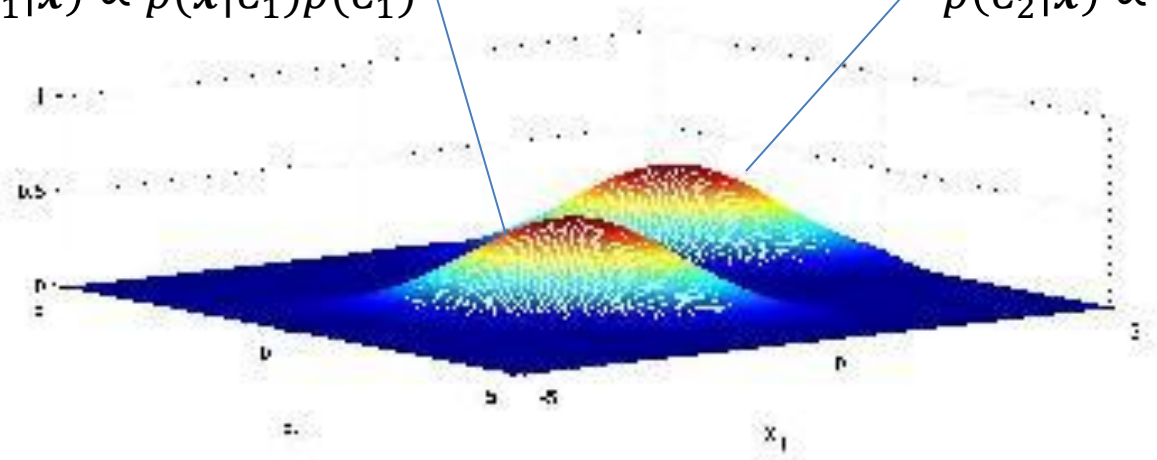
$p(C_2|\boldsymbol{x}) \propto p(\boldsymbol{x}|C_2)p(C_2)$

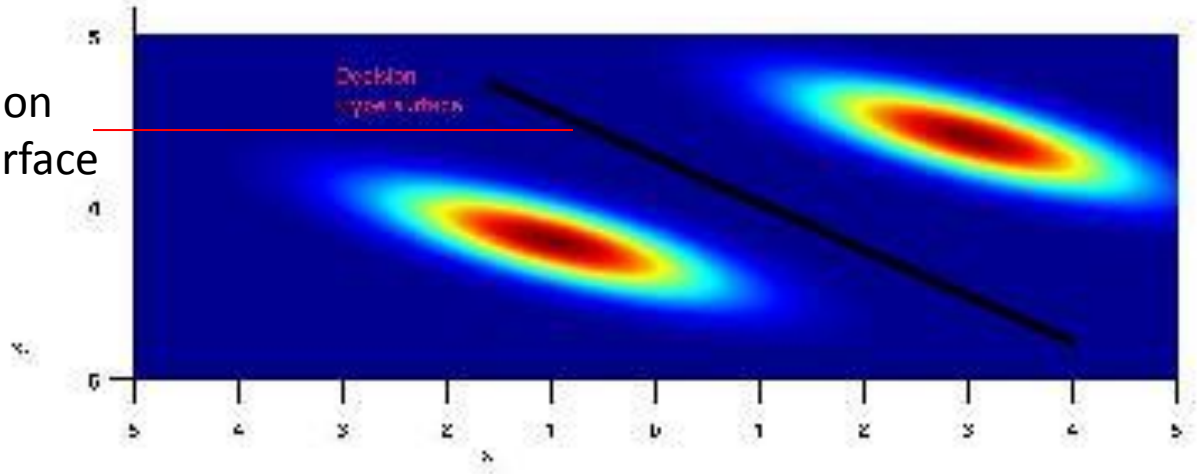


Now consider class 2 with a similar Gaussian conditional density, which has the same covariance but a different mean

$p(C_1|\boldsymbol{x}) \propto p(\boldsymbol{x}|C_1)p(C_1)$

$p(C_2|\boldsymbol{x}) \propto p(\boldsymbol{x}|C_2)p(C_2)$



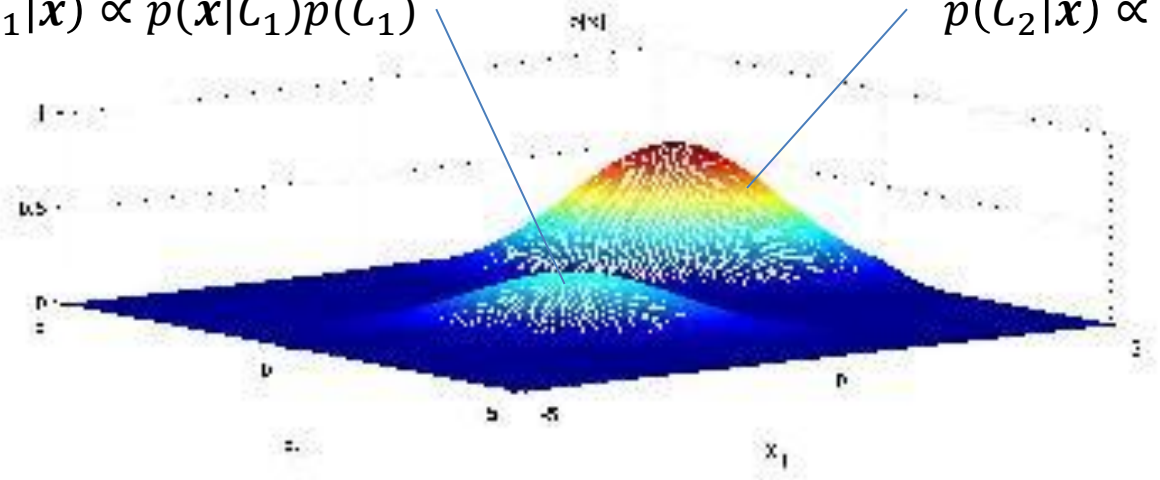


decision hypersurface

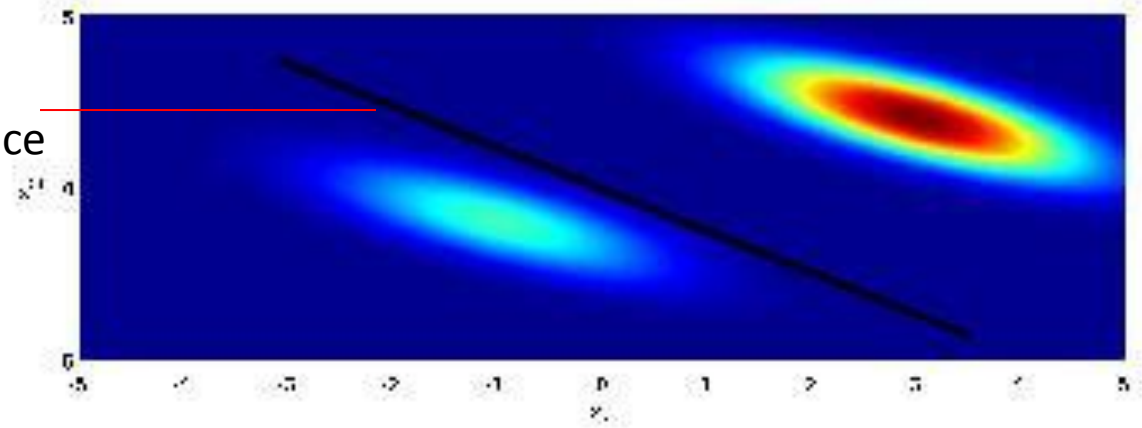
If the priors for each class are the same (i.e. 0.5), then the decision hypersurface cuts directly between the two means, with a direction parallel to the elliptical shape of the modes of the Gaussian densities shaped by their (identical) covariance matrices.

$p(C_1|\boldsymbol{x}) \propto p(\boldsymbol{x}|C_1)p(C_1)$

$p(C_2|\boldsymbol{x}) \propto p(\boldsymbol{x}|C_2)p(C_2)$

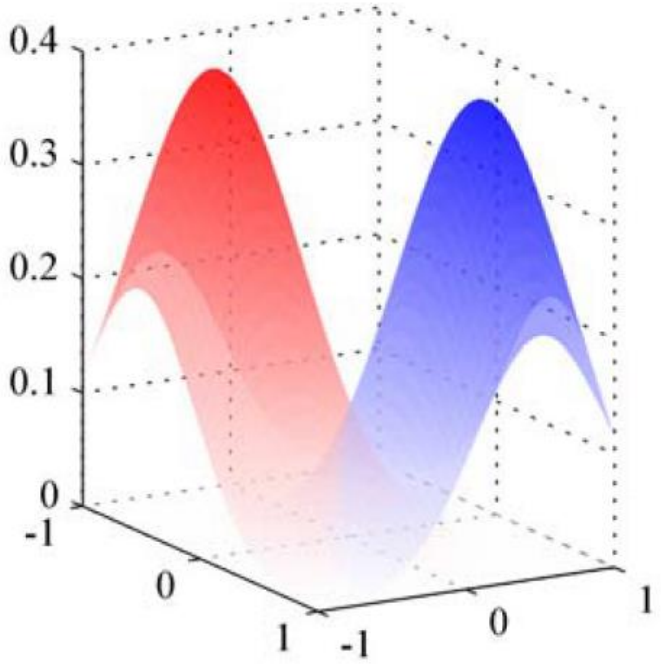


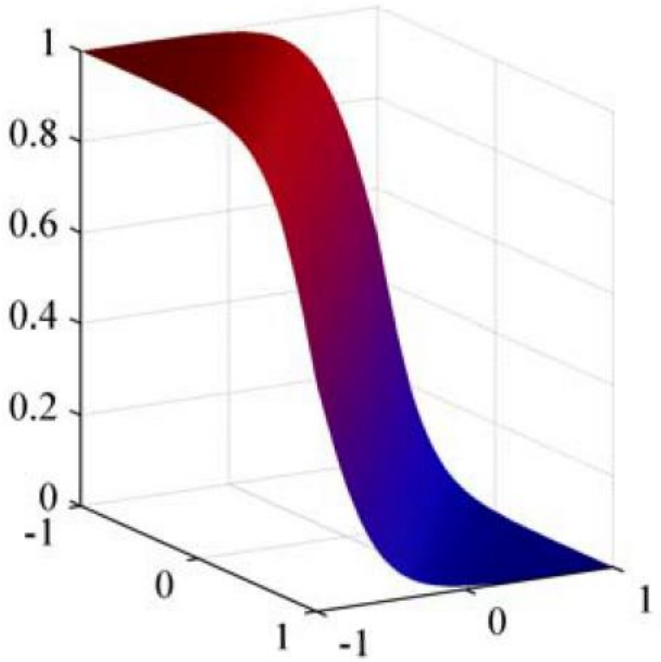
decision hypersurface



Now if the priors for each class are unequal, the decision hypersurface cuts between the two means with a direction as before, but now will be located further from the more likely class. This biases the predictor in favor of the more likely class.

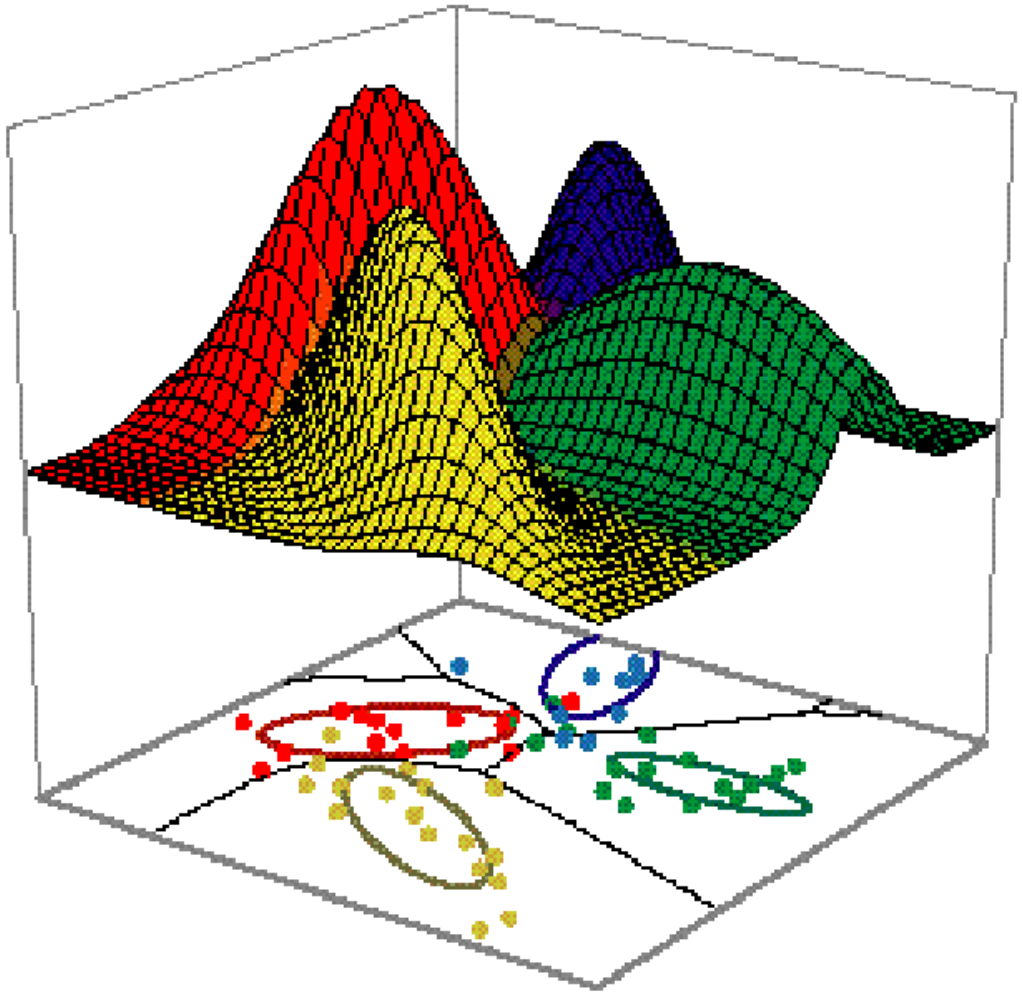
Log-odds for two classes



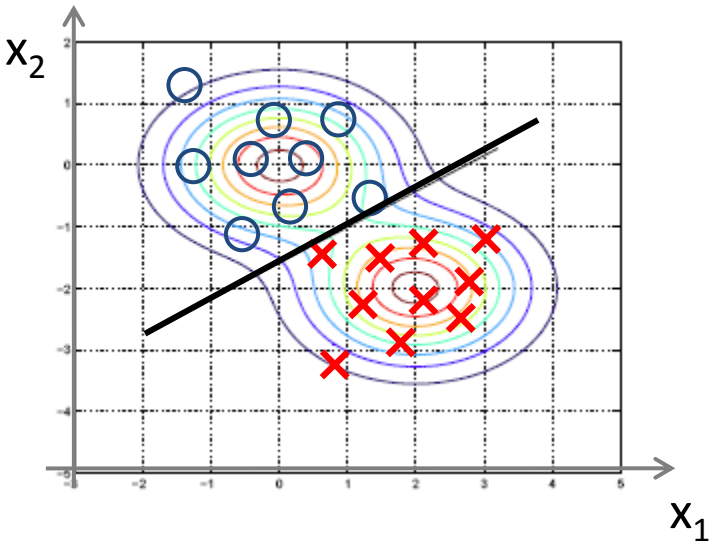


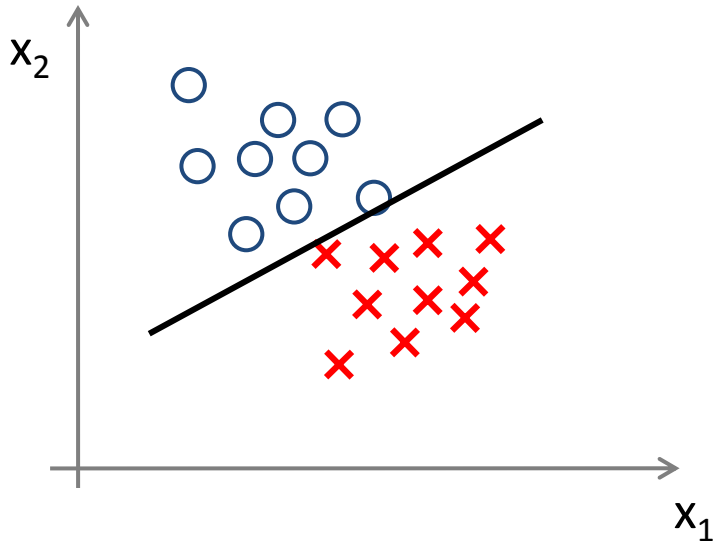
More than two classes, unequal covariances

- more general case of unequal covariances (here shown for four classes)
- the decision hypersurface is no longer a hyperplane, i.e. it is nonlinear.



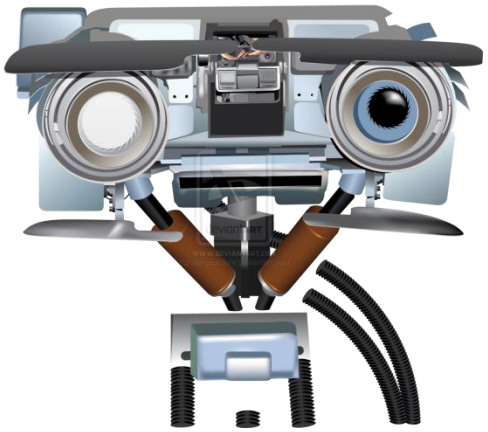
Generative vs Discriminative





- Generative: model the class-conditional distribution of features
- Pros: Can use it to generate new features
- Cons: more parameters O(n^2)

- Discriminative: model the decision boundary directly, e.g. Logistic Regression
- Pros: fewer parameters, O(n)
- Cons: Cannot generate new features



Linear Algebra Background

Matrix Differentiation

Matrix Differentiation

Jacobian

- (reading for today, see Schedule)

$$\mathbf{y} = \psi(\mathbf{x})$$

$\mathbf{x}$ is an n-element vector $\mathbf{y}$ is an m-element vector

Jacobian

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Matrix Differentiation

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

$\mathbf{y}$ *is* $\mathrm{m} \times 1$, $\mathbf{x}$ *is* $\mathrm{n} \times 1$, $\mathbf{A}$ *is* $\mathrm{m} \times \mathrm{n}$

$\mathbf{A}$ *does not depend on* $\mathbf{x}$

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}$$

Matrix Differentiation

$$\alpha = \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}$$

$\mathbf{x}$ *is* $\mathrm{n} \times 1$, $\mathbf{A}$ *is* $\mathrm{n} \times \mathrm{n}$, *and* $\mathbf{A}$ *does not depend on* $\mathbf{x}$

$$\alpha = \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}$$

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^\mathsf{T}\left(\mathbf{A} + \mathbf{A}^\mathsf{T}\right)$$

Special case:

$\mathbf{A}$ *is a symmetric matrix*

$$\frac{\partial \alpha}{\partial \mathbf{x}} = 2\mathbf{x}^\mathsf{T}\mathbf{A}$$