CSE276C - Regression and Classification

Henrik I. Christensen

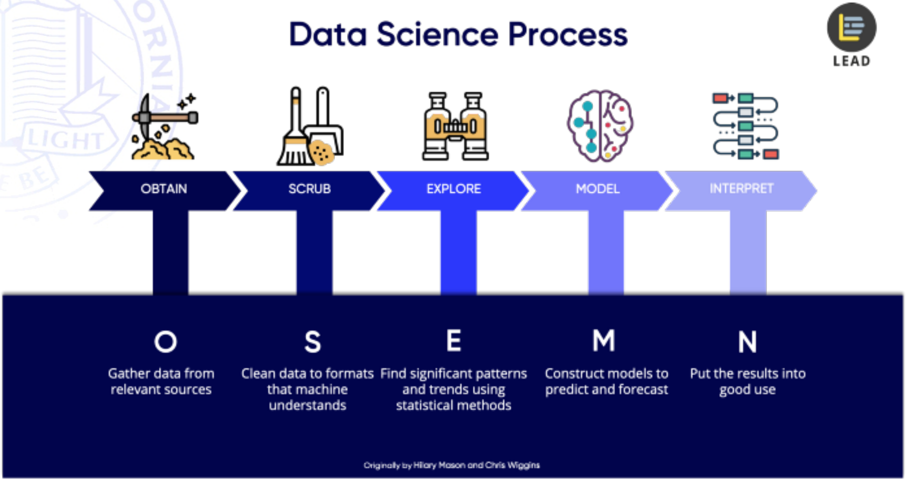
Computer Science and Engineering
University of California, San Diego

November 2021

Introduction

- Data science is a big part of robotics
- Many aspects of robotics rely on data analysis
  - Recognition of objects
  - Adpative Control
  - Clean-up of sensor information
  - ...

A general framework



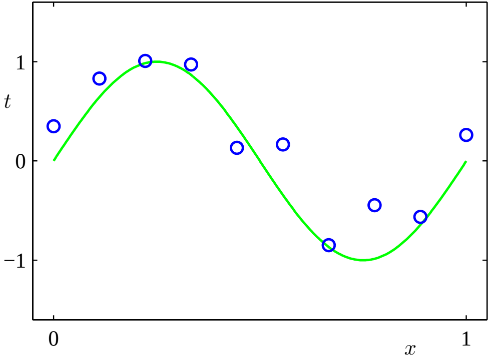
Outline

Introduction

- The objective of regression is to enable prediction of a value based on modeling over a dataset X.
- Consider a set of D observations over a space
- How can we generate estimates for the future?
  - Battery time?
  - Time to completion?
  - Position of doors?

Introduction (2)

- Example



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_m x^m = \sum_{i=0}^{m} w_i x^i$$

Introduction (3)

- In general the functions could be beyond simple polynomials
- The "components" ($\phi_i(x)$) are termed *basis functions*, i.e.

$$y(x, \mathbf{w}) = \sum_{i=0}^{m} w_i \phi_i(x) = \vec{w}^T \phi(\vec{x})$$

Outline

Loss Function

- For optimization we need a penalty / loss function

$$L(t, y(x))$$

- Expected loss is then

$$E[L] = \int \int L(t, y(x)) p(x, t) dx dt$$

- For the squared loss function we have

$$E[L] = \int \int \{y(x) - t\}^2 p(x, t) dx dt$$

- Goal: choose y(x) to minimize expected loss (E[L])

Loss Function

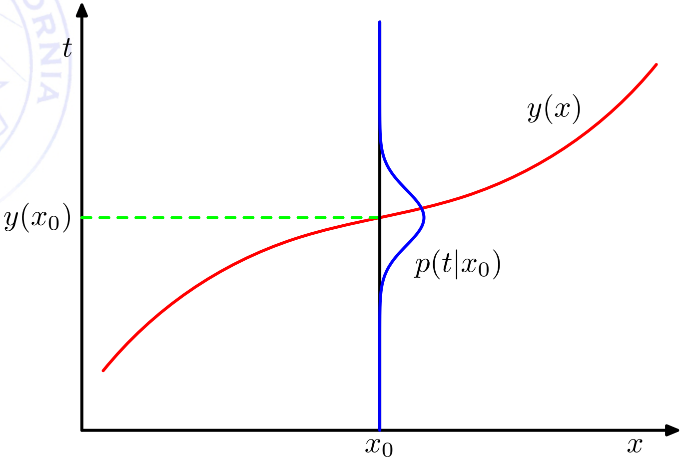
- Derivation of the extrema

$$\frac{\delta E[L]}{\delta y(x)} = 2\int \{y(x) - t\} p(x,t) dt = 0$$

- Implies that

$$y(x) = \frac{\int tp(x,t)dt}{p(x)} = \int tp(t|x)dt = E[t|x]$$

Loss Function - Interpretation



Alternative

- Consider a small rewrite

$$\{y(x) - t\}^2 = \{y(x) - E[t|x] + E[t|x] - t\}^2$$

- The expected loss is then

$$E[L] = \int \{y(x) - E[t|x]\}^2 p(x) dx + \int \{E[t|x] - t\}^2 p(x) dx$$

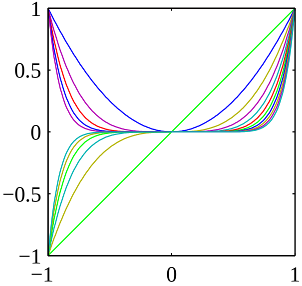
Outline

Polynomial Basis Functions

Basic Definition:

$$\phi_i(x) = x^i$$

Global functions
Small change in x affects all of them



Gaussian Basis Functions

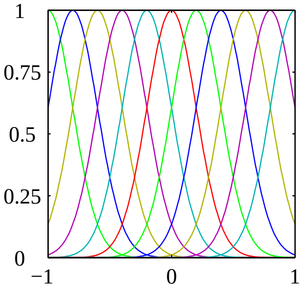
Basic Definition:

$$\phi_i(x) = e^{-\frac{(x-\mu_i)^2}{2s^2}}$$

A way to Gaussian mixtures, local impact

Not required to have probabilistic interpretation.

μ control position and s control scale



Sigmoid Basis Functions

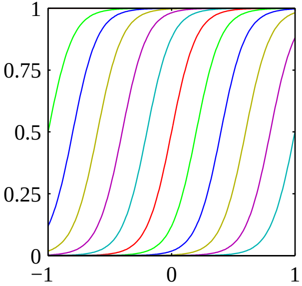
Basic Definition:

$$\phi_i(x) = \sigma\left(\frac{x - \mu_i}{s}\right)$$

where

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

μ controls location and s controls slope



Maximum Likelihood & Least Squares

- Assume observation from a deterministic function contaminated by Gaussian Noise

$$t = y(x, w) + \epsilon \quad p(\epsilon|\beta) = N(\epsilon|0, \beta^{-1})$$

  the problem at hand is then

$$p(t|x, w, \beta) = N(t|y(x, w), \beta^{-1})$$

- From a series of observations we have the likelihood

$$p(\mathbf{t}|\mathbf{X}, w, \beta) = \prod_{i=1}^{N} N(t_i|w^T\phi(x_i), \beta^{-1})$$

Maximum Likelihood & Least Squares (2)

- This results in

$$\ln p(\mathbf{t}|\mathbf{w}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})$$

- where

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \{t_i - \mathbf{w}^T \phi(x_i)\}^2$$

is the sum of squared errors

Maximum Likelihood & Least Squares (3)

- Computing the extrema yields:

$$\mathbf{w}_{ML} = \left(\Phi^T\Phi\right)^{-1}\Phi^T\mathbf{t}$$

- where

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_1) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix}$$

Line Estimation

- Least square minimization:
  - Line equation: $y = ax + b$
  - Error in fit: $\sum_i (y_i - ax_i - b)^2$
  - Solution:

$$\begin{pmatrix} \bar{y^2} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} \bar{x^2} & \bar{x} \\ \bar{x} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

  - So what is the problem?

Line Estimation

- Least square minimization:
  - Line equation: $y = ax + b$
  - Error in fit: $\sum_i (y_i - ax_i - b)^2$
  - Solution:

$$\begin{pmatrix} \bar{y^2} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} \bar{x^2} & \bar{x} \\ \bar{x} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

  - So what is the problem?
  - Minimizes vertical errors. Non-robust!

LSQ on Lasers

- Line model: $r_i \cos(\phi_i - \theta) = \rho$
- Error model: $d_i = r_i \cos(\phi_i - \theta) - \rho$
- Optimize: $\mathrm{argmin}_{(\rho,\theta)} \sum_i (r_i \cos(\phi_i - \theta) - \rho)^2$
- Error model derived in **(author?)** [1]
- Well suited for "clean-up" of Hough lines

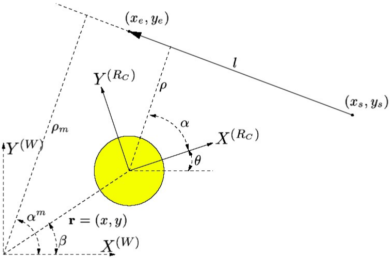
Total Least Squares

- Line equation: $ax + by + c = 0$
- Error in fit: $\sum_i (ax_i + by_i + c)^2$ where $a^2 + b^2 = 1$.
- Solution:

$$\left(\begin{array}{cc} \bar{x^2} - \bar{x}\bar{x} & \bar{xy} - \bar{x}\bar{y} \\ \bar{xy} - \bar{x}\bar{y} & \bar{y^2} - \bar{y}\bar{y} \end{array} \right) \left(\begin{array}{c} a \\ b \end{array} \right) = \mu \left(\begin{array}{c} a \\ b \end{array} \right)$$

  where μ is a scale factor.
- $c = -a\bar{x} - b\bar{y}$

Line Representations



- The line representation is crucial
- Often a redundant model is adopted
- Line parameters vs end-points
- Important for fusion of segments.
- End-points are less stable

Sequential Adaptation

- In some cases one at a time estimation is more suitable
- Also known as gradient descent

$$
\begin{aligned}
\mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} - \eta \nabla E_n \\
&= \mathbf{w}^{(\tau)} - \eta (t_n - \mathbf{w}^{(\tau)T} \phi(x_n)) \phi(x_n)
\end{aligned}
$$

- Knows as least-mean square (LMS). An issue is how to choose η?

Regularized Least Squares

- As seen in previous lecture sometime control of parameters might be useful.
- Consider the error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

- which generates

$$\frac{1}{2}\sum_{i=1}^{N}\{t_i - w^t\phi(x_i)\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

- which is minimized by

$$w = \left(\lambda I + \Phi^T\Phi\right)^{-1}\Phi^T\mathbf{t}$$

Outline

Bayesian Linear Regression

- Define a conjugate prior over w

$$p(w) = N(w|m_0, S_0)$$

- given the likelihood function and regular from Bayesian analysis we can derive

$$p(w|t) = N(w|m_N, S_N)$$

- where

$$\begin{aligned} m_N &= S_N \left(S_0^{-1} m_0 + \beta \Phi^T t\right) \\ S_N^{-1} &= S_0^{-1} + \beta \Phi^T \Phi \end{aligned}$$

Bayesian Linear Regression (2)

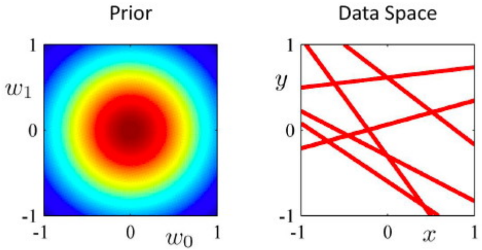
- A common choice is

$$p(w) = N(w|0, \alpha^{-1}I)$$

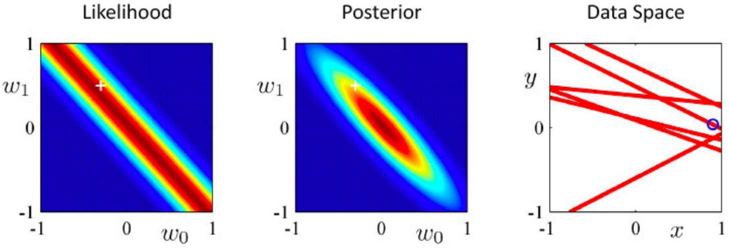
- So that

$$\begin{aligned} m_N &= \beta S_N \Phi^T t \\ S_N^{-1} &= \alpha I + \beta \Phi^T \Phi \end{aligned}$$

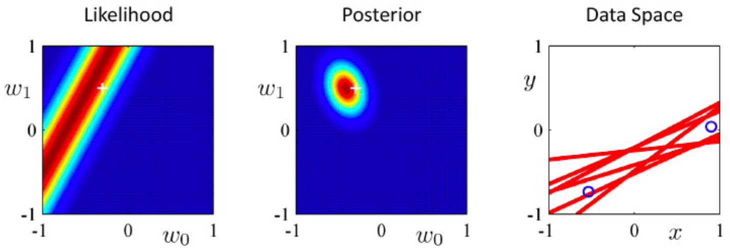
Example - No Data



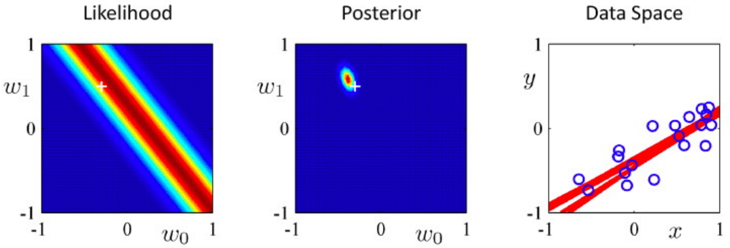
Example - 1 Data Point



Example - 2 Data Points



Example - 20 Data Points



Outline

Bayesian Model Comparison

- How does one select an appropriate model?
- Assume for a minute we want to compare a set of models M_i, $i \in 1, \dots L$ for a dataset D
- We could compute

$$p(M_i|D) \propto p(D|M_i)p(M_i)$$

- Bayes Factor: Ratio of evidence for two models

$$\frac{p(D|M_i)}{p(D|M_j)}$$

The mixture distribution approach

- We could use all the models:

$$p(t|x, D) = \sum_{i=1}^{L} p(t|x, M_i, D)p(M_i|D)$$

- Or simply go with the most probably/best model.

Model Evidence

- We can compute model evidence

$$p(D|M_i) = \int p(D|w, M_i)p(w|M_i)dw$$

- Allow computation of model fit based on parameter range

Evaluation of Parameters

- Evaluation of posterior over parameters

$$p(w|D, M_i) = \frac{P(D|w, M_i)p(w|M_i)}{P(D|M_i)}$$

- There is a need to understand how good is a model?

Model Comparison

- Consider evaluation of a model w. parameters w

$$p(D) = \int p(D|w)p(w)dw \approx p(D|w_{map})\frac{\sigma_{posterior}}{\sigma_{prior}}$$

- Then

$$\ln p(D) \approx \ln p(D|w_{map}) + \ln\left(\frac{\sigma_{posterior}}{\sigma_{prior}}\right)$$

Model Comparison as Kullback-Leibler

- From earlier we have comparison of distributions

$$KL = \int p(D|M_1) \ln \frac{p(D|M_1)}{p(D|M_2)} dD$$

- Enables comparison of two different models

Outline

Regression Summary

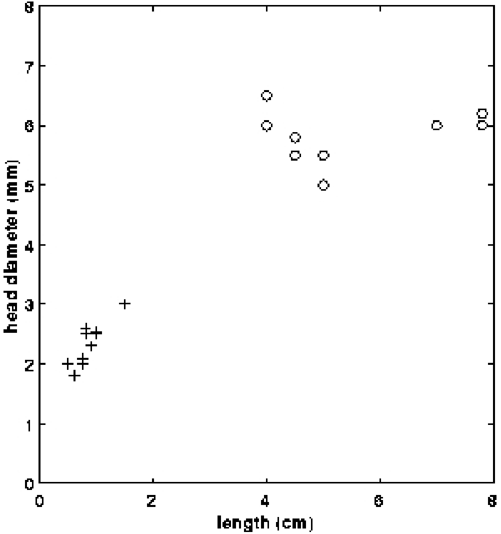
- Brief intro to linear methods for estimation of models
- Prediction of values and models
  - Needed for adaptive selection of models (black-box/grey-box)
  - Evaluation of sensor models, ...
- Consideration of batch and recursive estimation methods
- Significant discussion of methods for evaluation of models and parameters.
- This far purely a discussion of linear models

Outline

Classification Introduction

- Linear classification of data
  - Basic pattern recognition
  - Separation of data: buy/sell
  - Segmentation of line data, . . .

Simple Example - Bolts or Needles



Classification

- Given
  - An input vector: X
  - A set of classes: $c_i \in \mathcal{C}, \quad i = 1, \ldots, k$
- Mapping $m : X \to \mathcal{C}$
- Separation of space into decision regions
- Boundaries termed decision boundaries/surfaces

Basis Formulation

- It is a 1-of-K coding problem
- Target vector: $\mathbf{t} = (0, \ldots, 1, \ldots, 0)$
- Consideration of 3 different approaches
  1. Optimization of discriminant function
  2. Bayesian Formulation: $p(c_i|x)$
  3. Learning & Decision fusion

Code for experimentation

- There are data sets and sample code available
  - NETLAB: `http://www.ncrg.aston.ac.uk/netlab/index.php`
  - Kaggle: `https://www.kaggle.com`
  - Lots of good robotics datasets too

Outline

Discriminant Functions

- Objective: input vector x assigned to a class c_i
- Simple formulation:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

- $\mathbf{w}$ is termed a weight vector
- w_0 is termed a bias
- Two class example: c_1 if $y(\mathbf{x}) \geq 0$ otherwise c_2

Basic Design

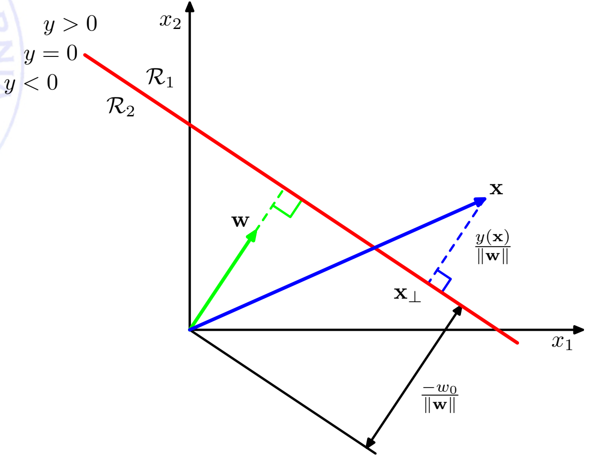
- Two points on decision surface $\mathbf{x}_a$ and $\mathbf{x}_b$
- $y(\mathbf{x}_a) = y(\mathbf{x}_b) = 0 \Rightarrow \mathbf{w}^T(\mathbf{x}_a - \mathbf{x}_b) = 0$
- $\mathbf{w}$ perpendicular to decision surface

$$\frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||} = -\frac{w_0}{||\mathbf{w}||}$$

- Define: $\tilde{w} = (w_0, \mathbf{w})$ and $\tilde{x} = (1, \mathbf{x})$ so that:

$$y(\mathbf{x}) = \tilde{w}^T\tilde{x}$$

Linear discriminant function



Multi Class Discrimination

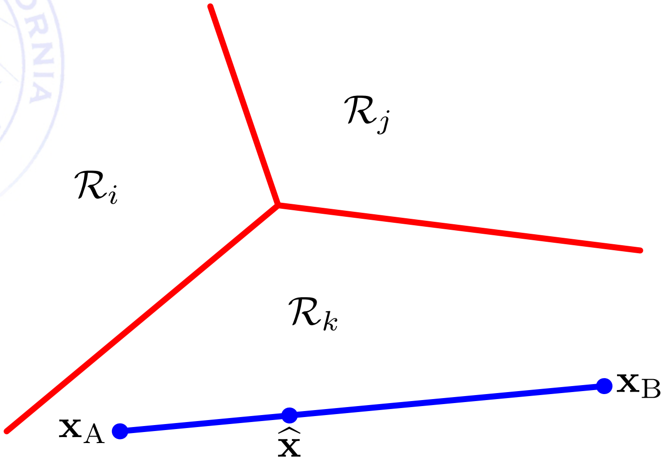
- Generation of multiple decision functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

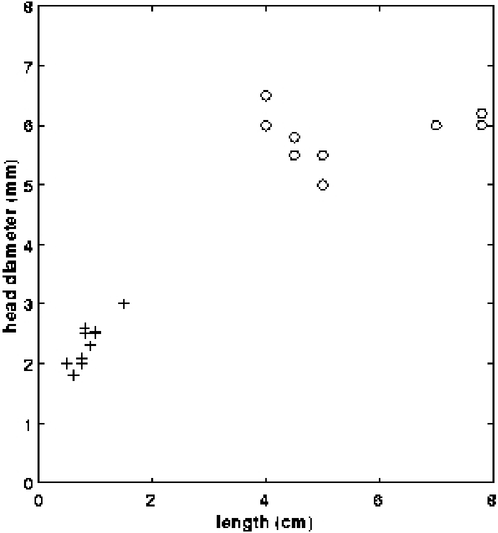
- Decision strategy

$$j = \arg \max_{i \in 1..k} y_i(\mathbf{x})$$

Multi-Class Decision Regions



Example - Bolts or Needles



Minimum distance classification

- Suppose we have computed the mean value for each of the classes
- $m_{needle} = [0.86, 2.34]^T$ and $m_{bolt} = [5.74, 5, 85]^T$
- We can then compute the minimum distance

$$d_j(x) = ||x - m_j||$$

- $\mathrm{a}rgmin_i d_i(x)$ is the best fit
- Decision functions can be derived

Bolts / Needle Decision Functions

Needle $d_{needle}(x) = 0.86x_1 + 2.34x_2 - 3.10$

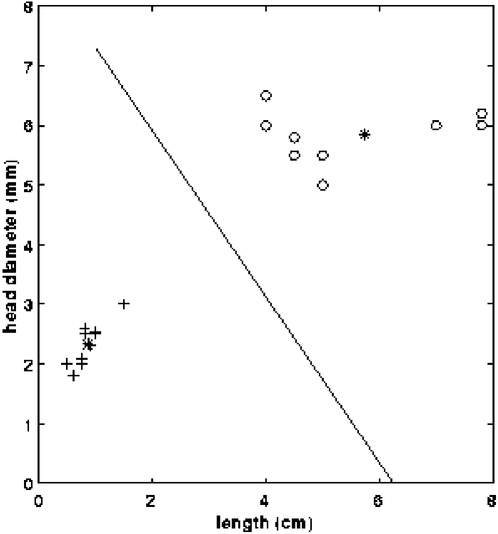
Bolt $d_{bolt}(x) = 5.74x_1 + 5.85x_2 - 33.59$

Decision boundary

$$d_i(x) - d_j(x) = 0$$

$$d_{needle/bolt}(x) = -4.88x_1 - 3.51x_2 + 30.49$$

Example decision surface



Outline

Least Squares for Classification

- Just like we could do LSQ for regression we can perform an approximation to the classification vector $\mathcal{C}$
- Consider again

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

- Rewrite to

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}$$

- Assuming we have a target vector $\mathbf{T}$

Least Squares for Classification

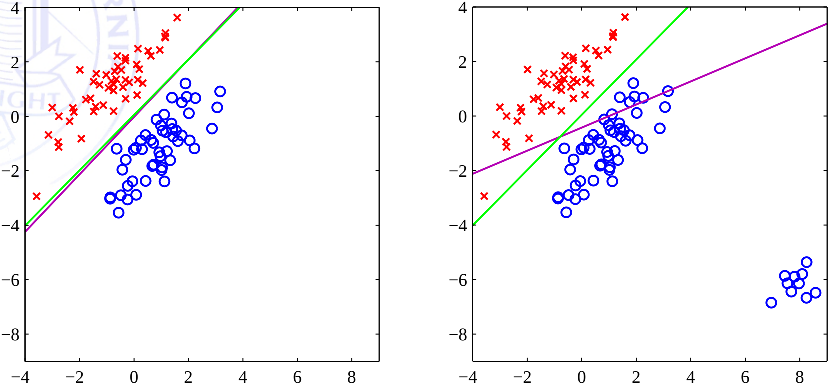
- The error is then:

$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2} Tr\left\{(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})\right\}$$

- The solution is then

$$\tilde{\mathbf{W}} = \left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\right)^{-1}\tilde{\mathbf{X}}^T\mathbf{T}$$

LSQ and Outliers



Outline

Fisher's linear discriminant

- Selection of a decision function that maximizes distance between classes
- Assume for a start

$$y = \mathbf{W}^T \mathbf{x}$$

- Compute m_1 and m_2

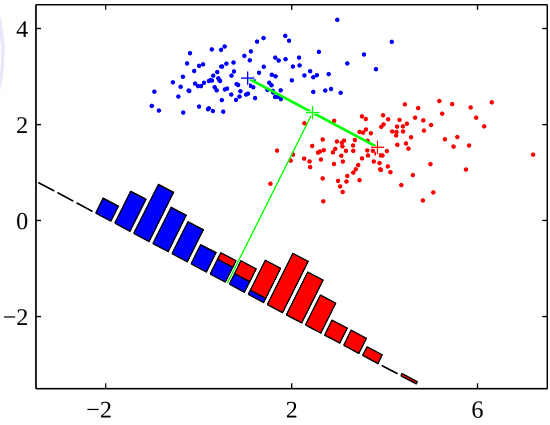
$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{i \in C_1} \mathbf{x}_i \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{j \in C_2} \mathbf{x}_j$$

- Distance:

$$m_2 - m_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1)$$

- where $m_i = \mathbf{w}\mathbf{m}_i$

The suboptimal solution



The Fisher criterion

- Consider the expression

$$J(w) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

- where $\mathbf{S}_B$ is the between class covariance and $\mathbf{S}_W$ is the within class covariance, i.e.

$$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$$

and

$$\mathbf{S}_W = \sum_{i=\mathcal{C}_1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^T + \sum_{i=\mathcal{C}_2} (\mathbf{x}_i - \mathbf{m}_2)(\mathbf{x}_i - \mathbf{m}_2)^T$$

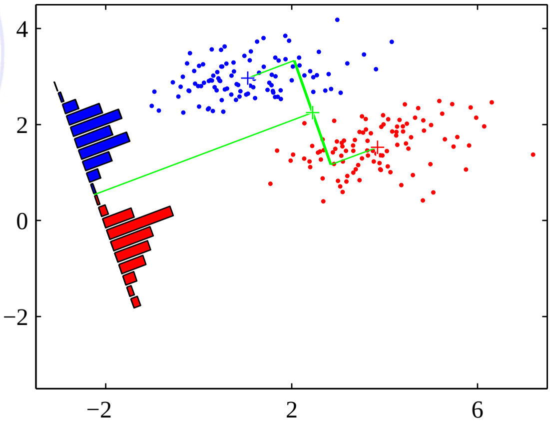
- Optimized when

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_w \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}$$

or

$$\mathbf{w} \propto \mathbf{S}_w^{-1} (\mathbf{m}_2 - \mathbf{m}_1)$$

The Fisher result



Generalization to $N > 2$

- Define a stacked weight factor

$$\mathbf{y} = \mathbf{W}^T \mathbf{x}$$

- The within class covariance generalizes to

$$\mathbf{S}_w = \sum_{k=1}^{K} \mathbf{S}_k$$

- The between class covariance is

$$\mathbf{S}_B = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$$

- It can be shown that $J(\mathbf{w})$ is optimized by the eigenvectors to the equation

$$S = \mathbf{S}_W^{-1} \mathbf{S}_B$$

Outline

Perceptron Algorithm

- Developed by Rosenblatt (1962)
- Formed an important basis for neural networks
- Use a non-linear transformation $\phi(x)$
- Construct a decision function

$$y(\mathbf{x}) = f\left(\mathbf{w}^T\phi(\mathbf{x})\right)$$

- where

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

The perceptron criterion

- Normally we want

$$w^t \phi(x_n) > 0$$

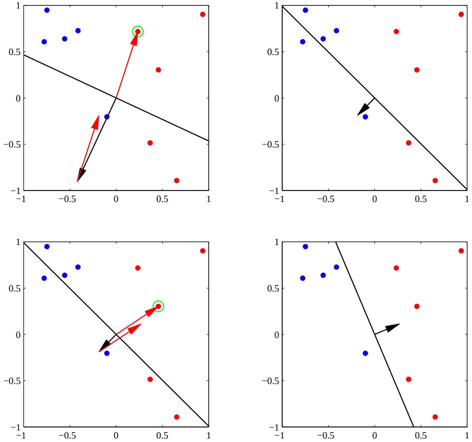
- Given the target vector definition

$$E_p(\mathbf{w}) = -\sum_{n \ in \mathcal{M}} \mathbf{w}^T \phi_n t_n$$

- Where $\mathcal{M}$ represents all the mis-classified samples
- We can make this a gradient descent as seen in last lecture

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \phi_n t_n$$

Perceptron learning example



Outline

Classification Summary

- Basics for discrimination / classification
- Obviously not all problems are linear
- Optimization of the distance/overlap between classes
  - Minimizing the probability of error classification
- Basic formulation as an optimization problem
- How to optimize between cluster distance? Covariance Weighted
- Basic recursive formulation
- Could we make it more robust?

Summary

- Data models are anchored in pure data driven or model based evaluation
- How can we use models to interpret data and extrapolate beyond the basic data?
- Covered basic models for regression and classification.