

Introduction to Classification

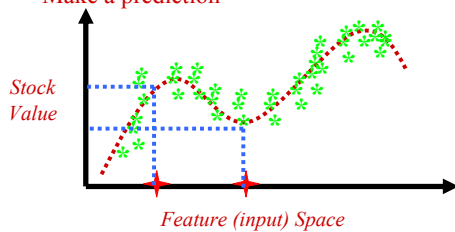
Greg Grudic

Today's Lecture Goals

- Introduction to classification
- Generative Models
 - Fisher (Linear Discriminative Analysis)
 - Gaussian Mixture Models
- Discriminative Models
 - Rosenblatt's Perceptron Learning Algorithm
- Nonlinear Extensions

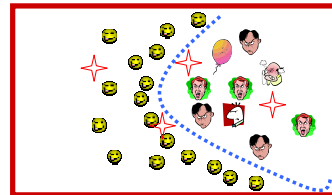
Last Week: Learning Regression Models

- Collect Training data
- Build Model: stock value = $F(\text{feature space})$
- Make a prediction



This Class: Learning Classification Models

- Collect Training data
- Build Model: happy = $F(\text{feature space})$
- Make a prediction



High
Dimensional
Feature (input)
Space

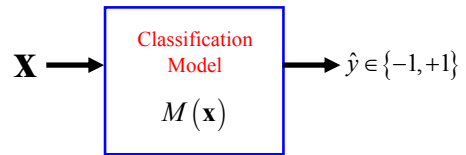
Binary Classification

- A binary classifier is a mapping from a set of d inputs to a single output which can take on one of **TWO** values
- In the most general setting
 - inputs: $\mathbf{x} \in \mathbb{R}^d$
 - output: $y \in \{-1, +1\}$
- Specifying the output classes as -1 and +1 is arbitrary!
 - Often done as a mathematical convenience

A Binary Classifier

Given learning data: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

A model is constructed:



The Learning Data

- **Learning algorithms don't care where the data comes from!**
- Here is a toy example from robotics...
 - Inputs from two sonar sensors:
 - sensor 1: $x_1 \in \mathbb{R}$
 - sensor 2: $x_2 \in \mathbb{R}$
 - Classification output:
 - Robot in Greg's office: $y = +1$
 - Robot NOT in Greg's office: $y = -1$

Classification Learning Data...

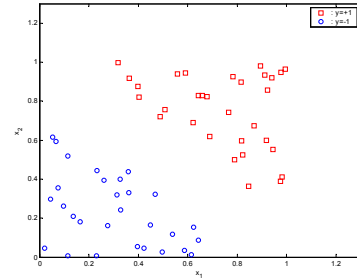
	x_1	x_2	y
<i>Example 1</i>	0.95013	0.58279	1
<i>Example 2</i>	0.23114	0.4235	-1
<i>Example 3</i>	0.8913	0.43291	1
<i>Example 4</i>	0.018504	0.76037	-1
...

The Learning Data

- Symbolic Representation of N learning examples of d dimensional inputs

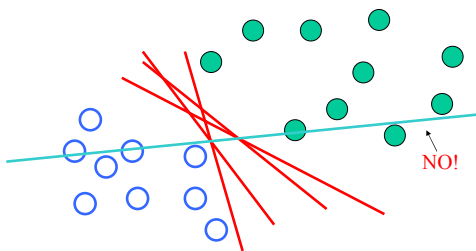
$$\begin{pmatrix} x_{11} & \cdots & x_{1d} & y_1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{N1} & \cdots & x_{Nd} & y_N \end{pmatrix}$$

Graphical Representation of Classification Training Data

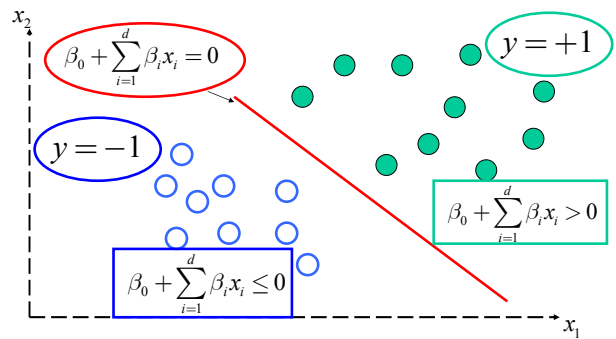


Linear Separating Hyper-Planes

How many lines can separate these points?



Linear Separating Hyper-Planes



Linear Separating Hyper-Planes

- The Model:

$$\hat{y} = M(\mathbf{x}) = \text{sgn}[\hat{\beta}_0 + (\hat{\beta}_1, \dots, \hat{\beta}_d) \mathbf{x}^T]$$

- Where:

$$\text{sgn}[A] = \begin{cases} 1 & \text{if } A > 0 \\ -1 & \text{otherwise} \end{cases}$$

- The decision boundary:

$$\hat{\beta}_0 + (\hat{\beta}_1, \dots, \hat{\beta}_d) \mathbf{x}^T = 0$$

Linear Separating Hyper-Planes

- The model parameters are:

$$(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)$$

- The *hat* on the betas means that they are estimated from the data

– In the class notes... Sometimes the hat will be there and sometimes it won't!

- Many different learning algorithms have been proposed for determining $(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)$

Rosenblatt's Perceptron Learning Algorithm

- Dates back to the 1950's and is the motivation behind Neural Networks
- The algorithm:
 - Start with a random hyperplane $(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)$
 - Incrementally modify the hyperplane such that points that are misclassified move closer to the correct side of the boundary
 - Stop when all learning examples are correctly classified

Rosenblatt's Perceptron Learning Algorithm

- The algorithm is based on the following property:
 - **Signed** distance of any point \mathbf{x} to the boundary is *proportional to*
- Therefore, if M is the set of misclassified learning examples, we can push them closer to the boundary by minimizing the following

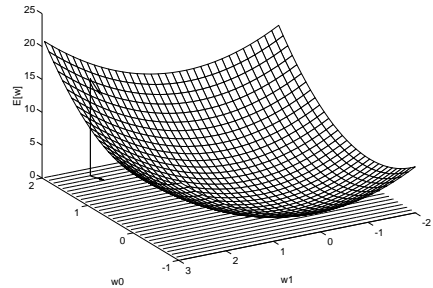
$$D(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d) = -\sum_{i \in M} y_i (\hat{\beta}_0 + (\hat{\beta}_1, \dots, \hat{\beta}_d) \mathbf{x}_i^T)$$

Rosenblatt's Minimization Function

- This is classic Machine Learning!
- First define a cost function in model parameter space

$$D(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d) = -\sum_{i \in M} y_i \left(\hat{\beta}_0 + \sum_{k=1}^d \hat{\beta}_k x_{ik} \right)$$
- Then find an algorithm that modifies $(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)$ such that this cost function is minimized
- One such algorithm is **Gradient Descent**

Gradient Descent



The Gradient Descent Algorithm

$$\hat{\beta}_i \leftarrow \hat{\beta}_i - \rho \frac{\partial D(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)}{\partial \hat{\beta}_i}$$

Where the learning rate is defined by: $\rho > 0$

The Gradient Descent Algorithm for the Perceptron

$$\frac{\partial D(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)}{\partial \hat{\beta}_0} = -\sum_{i \in M} y_i, \quad \frac{\partial D(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_d)}{\partial \hat{\beta}_j} = -\sum_{i \in M} y_i x_{ij}, \quad j = 1, \dots, d$$

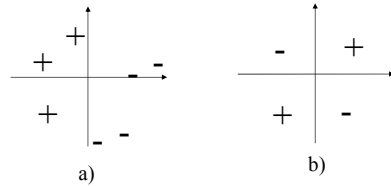
$$\begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_d \end{pmatrix} \leftarrow \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_1 \\ \hat{\beta}_d \end{pmatrix} - \rho \begin{pmatrix} y_i \\ y_i x_{i1} \\ \vdots \\ y_i x_{id} \end{pmatrix}$$

The Good Theoretical Properties of the Perceptron Algorithm

- If a solution exists the algorithm will always converge in a finite number of steps!
- Question: Does a solution always exist?

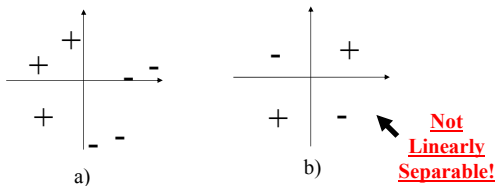
Linearly Separable Data

- Which of these datasets are separable by a linear boundary?



Linearly Separable Data

- Which of these datasets are separable by a linear boundary?



Bad Theoretical Properties of the Perceptron Algorithm

- If the data is not linearly separable, algorithm cycles forever!
 - Cannot converge!
 - This property stopped research in this area between 1968 and 1984...
 - *Perceptrons*, Minsky and Pappert, 1969
- There are infinitely many solutions
- When data is linearly separable, the number of steps to converge can be very large (depends on size of gap between classes)

What about Nonlinear Data?

- Data that is not linearly separable is called nonlinear data
- Nonlinear data can often be mapped into a nonlinear space where it is linearly separable

Nonlinear Models

- The Linear Model:

$$\hat{y} = M(\mathbf{x}) = \text{sgn} \left[\hat{\beta}_0 + \sum_{i=1}^d \hat{\beta}_i x_i \right]$$

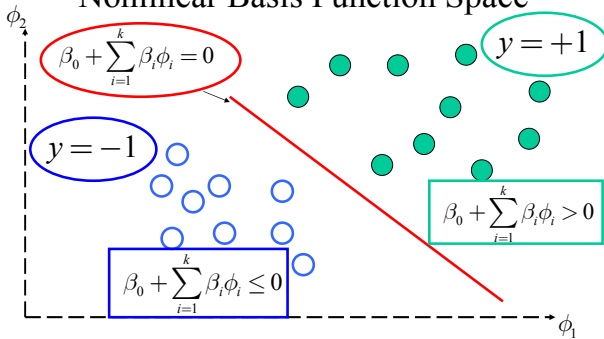
- The Nonlinear (basis function) Model:

$$\hat{y} = M(\mathbf{x}) = \text{sgn} \left[\hat{\beta}_0 + \sum_{i=1}^k \hat{\beta}_i \phi_i(\mathbf{x}) \right]$$

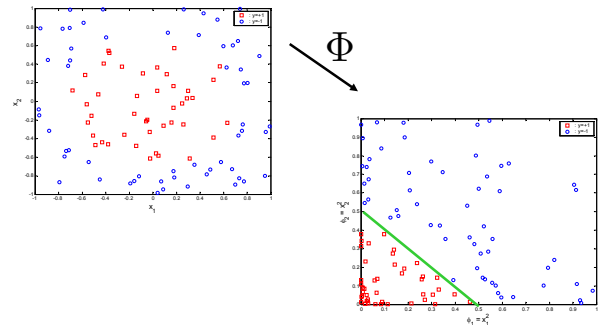
- Examples of Nonlinear Basis Functions:

$$\phi_1(\mathbf{x}) = x_1^2 \quad \phi_2(\mathbf{x}) = x_2^2 \quad \phi_3(\mathbf{x}) = x_1 x_2 \quad \phi_4(\mathbf{x}) = \sin(x_{55})$$

Linear Separating Hyper-Planes In Nonlinear Basis Function Space



An Example



Kernels as Nonlinear Transformations

- Polynomial

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + q)^k$$

- Sigmoid

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \theta)$$

- Gaussian or Radial Basis Function (RBF)

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

The Kernel Model

Training Data: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

$$\hat{y} = M(\mathbf{x}) = \text{sgn}\left[\hat{\beta}_0 + \sum_{i=1}^N \hat{\beta}_i K(\mathbf{x}_i, \mathbf{x})\right]$$

The number of basis functions equals the number of training examples!

- Unless some of the beta's get set to zero...

Gram (Kernel) Matrix

Training Data: $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$

$$K = \begin{pmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & \dots & K(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ K(\mathbf{x}_N, \mathbf{x}_1) & \dots & K(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix}$$

Properties:

- Positive Definite Matrix
- Symmetric
- Positive on diagonal
- N by N

Picking a Model Structure?

- How do you pick the Kernels?
 - Kernel parameters
- These are called **learning parameters** or **hyperparameters**
 - Two approaches choosing learning parameters
 - Bayesian
 - Learning parameters must maximize probability of correct classification based on prior biases
 - Frequentist
 - Use validation data
- More on learning parameter selection later

Perceptron Algorithm Convergence

- Two problems:
 - No convergence when data is not separable in basis function space
 - Gives infinitely many solutions when data is separable
- Can we modify the algorithm to fix these problems?