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(feature space)
• Make a prediction

This Class: Learning Classification Models

• Collect Training data
• Build Model: happy = F(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: $x \in \mathbb{R}^d$
  - 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: $(x_1, y_1), \ldots, (x_N, y_N)$

A model is constructed:

$$
\begin{align*}
\hat{y} &\in \{-1, +1\} \\
M(x) &\xrightarrow{\text{Model}} \hat{y}
\end{align*}
$$

## 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...

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example 1</strong></td>
<td>0.95013</td>
<td>0.58279</td>
<td>1</td>
</tr>
<tr>
<td><strong>Example 2</strong></td>
<td>0.23114</td>
<td>0.4235</td>
<td>-1</td>
</tr>
<tr>
<td><strong>Example 3</strong></td>
<td>0.8913</td>
<td>0.43291</td>
<td>1</td>
</tr>
<tr>
<td><strong>Example 4</strong></td>
<td>0.018504</td>
<td>0.76037</td>
<td>-1</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
The Learning Data

- Symbolic Representation of $N$ learning examples of $d$ dimensional inputs

\[
\begin{bmatrix}
  x_{11} & \ldots & x_{1d} & y_1 \\
  \vdots & \ddots & \vdots & \vdots \\
  x_{N1} & \ldots & x_{Nd} & y_N
\end{bmatrix}
\]

Graphical Representation of Classification Training Data

Linear Separating Hyper-Planes

How many lines can separate these points?

Linear Separating Hyper-Planes

\[
\beta_0 + \sum_{i=1}^{d} \beta_i x_i = 0
\]

\[
\beta_0 + \sum_{i=1}^{d} \beta_i x_i \leq 0
\]

\[
\beta_0 + \sum_{i=1}^{d} \beta_i x_i > 0
\]
## Linear Separating Hyper-Planes

### The Model:
\[
\hat{y} = M(x) = \text{sgn}\left[\beta_0 + \left(\beta_1, \ldots, \beta_d\right)x^T\right]
\]

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

### The decision boundary:
\[
\hat{\beta}_0 + \left(\hat{\beta}_1, \ldots, \hat{\beta}_d\right)x^T = 0
\]

---

## Rosenblatt’s Preceptron Learning Algorithm

### Dates back to the 1950’s and is the motivation behind Neural Networks

### The algorithm:
- Start with a random hyperplane \(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_d\)
- 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

---

## Linear Separating Hyper-Planes

### The model parameters are:
\[
\left(\beta_0, \beta_1, \ldots, \beta_d\right)
\]

### 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 \(\left(\beta_0, \beta_1, \ldots, \beta_d\right)\)
Rosenblatt’s Minimization Function

- This is classic Machine Learning!
- First define a cost function in model parameter space
  \[ D(\beta_0, \beta_1, \ldots, \beta_d) = -\sum_{i=1}^{N} \frac{1}{2} \left[ y_i - \sum_{j=1}^{d} \beta_j x_{ij} \right]^2 \]
- Then find an algorithm that modifies \((\beta_0, \beta_1, \ldots, \beta_d)\) such that this cost function is minimized
- One such algorithm is **Gradient Descent**

The Gradient Descent Algorithm

\[
\hat{\beta}_i \leftarrow \hat{\beta}_i - \rho \frac{\partial D(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_d)}{\partial \hat{\beta}_i}
\]

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

The Gradient Descent Algorithm for the Perceptron

\[
\frac{\partial D(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_d)}{\partial \hat{\beta}_i} = \sum_{x \in D} y_i x_{ij} - \rho y_i 
\]

\[
\begin{bmatrix}
\hat{\beta}_0 \\
\hat{\beta}_1 \\
\vdots \\
\hat{\beta}_d \\
\end{bmatrix} - \rho 
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N \\
\end{bmatrix} 
\]
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?

a) b)

Not Linearly Separable!

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:
  \[ \hat{y} = M(x) = \text{sgn}\left[ \beta_0 + \sum_{i=1}^{n} \beta_i x_i \right] \]
- The Nonlinear (basis function) Model:
  \[ \hat{y} = M(x) = \text{sgn}\left[ \beta_0 + \sum_{i=1}^{n} \beta_i \phi_i(x) \right] \]
- Examples of Nonlinear Basis Functions:
  \[ \phi_1(x) = x^2 \quad \phi_2(x) = x^3 \quad \phi_3(x) = x_1 x_2 \quad \phi_4(x) = \sin(x_3) \]

Linear Separating Hyper-Planes In Nonlinear Basis Function Space

An Example
Kernels as Nonlinear Transformations

- Polynomial
  \[ K(x_i, x_j) = (\langle x_i, x_j \rangle + q)^k \]
- Sigmoid
  \[ K(x_i, x_j) = \tanh(\kappa \langle x_i, x_j \rangle + \theta) \]
- Gaussian or Radial Basis Function (RBF)
  \[ K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_j\|^2\right) \]

The Kernel Model

Training Data: \((x_1, y_1), \ldots, (x_N, y_N)\)

\[ \hat{y} = M(x) = \text{sgn}\left(\beta_0 + \sum_{i=1}^{N} \beta_i K(x_i, x)\right) \]

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: \((x_1, y_1), \ldots, (x_N, y_N)\)

\[ K = \begin{pmatrix}
K(x_1, x_1) & \cdots & K(x_1, x_N) \\
\vdots & \ddots & \vdots \\
K(x_N, x_1) & \cdots & K(x_N, x_N)
\end{pmatrix} \]

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?