Rosenblatt’s Preceptron Learning Algorithm

Greg Grudic

Today’s Lecture Goals

• Introduction to binary classification
• The linear discriminative classifier
• Rosenblatt’s Preceptron Learning Algorithm
• Class projects

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:

$$
\begin{align*}
    \mathbf{X} &\rightarrow \hat{y} \in \{-1, +1\} \\
    \text{Classification Model} &\rightarrow M(\mathbf{x})
\end{align*}
$$
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: $y = -1$

---

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.95013</td>
<td>0.58279</td>
<td>1</td>
</tr>
<tr>
<td>0.23114</td>
<td>0.4235</td>
<td>-1</td>
</tr>
<tr>
<td>0.8913</td>
<td>0.43291</td>
<td>1</td>
</tr>
<tr>
<td>0.018504</td>
<td>0.76037</td>
<td>-1</td>
</tr>
</tbody>
</table>

---

The Learning Data

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

\[
\begin{pmatrix}
  x_{11} & \cdots & x_{1d} & y_1 \\
  \vdots & \ddots & \vdots & \vdots \\
  x_{N1} & \cdots & x_{Nd} & y_N \\
\end{pmatrix}
\]
Linear Separating Hyper-Planes

How many lines can separate these points?

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

\( y = +1 \)

\( y = -1 \)

\( \beta_0 + \sum_{i=1}^{d} \beta_i x_i \leq 0 \)

\( \beta_0 + \sum_{i=1}^{d} \beta_i x_i > 0 \)

**Linear Separating Hyper-Planes**

- The Model:
  \[ \hat{y} = M(x) = \text{sgn}[\beta_0 + (\hat{\beta}_1, ..., \hat{\beta}_d)x^T] \]

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

- The decision boundary:
  \[ \hat{\beta}_0 + (\hat{\beta}_1, ..., \hat{\beta}_d)x^T = 0 \]

**Linear Separating Hyper-Planes**

- The model parameters are:
  \( \left( \hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_d \right) \)

- 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 \( \left( \hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_d \right) \)
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 \((\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_d)\)
  - 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

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

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

Rosenblatt’s Minimization Function

- This is classic Machine Learning!
- First define a cost function in model parameter space
  \[
  D(\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_d) = -\sum_{x \in M} \left[ y \cdot \left( \hat{\beta}_0 + \sum_{i=1}^{d} \hat{\beta}_i x_i \right) \right]
  \]
- Then find an algorithm that modifies \((\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_d)\) such that this cost function is minimized
- One such algorithm is Gradient Descent
The Gradient Descent Algorithm for the Perceptron

$$\frac{\partial D}{\partial \beta} = - \sum_{i=1}^{d} y_i \cdot x_i$$

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?

Linearity Separable Data

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

a) ![Dataset a](image1)

b) ![Dataset b](image2)

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:
  \[ \phi_1(x) = x_1^2 \quad \phi_2(x) = x_2^2 \quad \phi_3(x) = x_1 x_2 \quad \phi_4(x) = \sin(x_{11}) \]
Picking a Model Structure?

- How do you pick the basis functions?
  - The number and type?
- These are called **learning parameters**
  - 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?
- See Homework 1 (next week)…