Perceptrons and Neural networks

Who needs probabilities?

• Previously: model data with distributions
  • Joint: $P(X,Y)$
    – e.g. Naïve Bayes
  • Conditional: $P(Y|X)$
    – e.g. Logistic Regression
• But wait, why probabilities?
  • Lets try to be error-driven!
Generative vs. Discriminative

• Generative classifiers:
  – E.g. naïve Bayes
  – A joint probability model with evidence variables
  – Query model for causes given evidence

• Discriminative classifiers:
  – No generative model, no Bayes rule, often no probabilities at all!
  – Try to predict the label Y directly from X
  – Robust, accurate with varied features
  – Loosely: mistake driven rather than model driven

Linear Classifiers

• Inputs are feature values
• Each feature has a weight
• Sum is the activation

\[
activation_w(x) = \sum_{i} w_i \cdot f_i(x) = w \cdot f(x)
\]

• If the activation is:
  – Positive, output class 1
  – Negative, output class 2
Example: Spam

• Imagine 3 features (spam is “positive” class):
  – free (number of occurrences of “free”)
  – money (occurrences of “money”)
  – BIAS (intercept, always has value 1)

$$w \cdot f(x) = \sum_{i} w_{i} \cdot f_{i}(x)$$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIAS</td>
<td>1</td>
<td>BIAS : -3</td>
</tr>
<tr>
<td>free</td>
<td>1</td>
<td>free : 4</td>
</tr>
<tr>
<td>money</td>
<td>1</td>
<td>money : 2</td>
</tr>
</tbody>
</table>

...  

“free money”

$$w \cdot f(x) > 0 \Rightarrow \text{SPAM!!!}$$

Binary Decision Rule

• In the space of feature vectors
  – Examples are points
  – Any weight vector is a hyperplane
  – One side corresponds to $Y=+1$
  – Other corresponds to $Y=-1$

$$w \cdot x = 0$$

$w$
Perceptron

\[ o(x_1, \ldots, x_n) = \begin{cases} 
1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\
-1 & \text{otherwise.} 
\end{cases} \]

Sometimes we’ll use simpler vector notation:

\[ o(\vec{x}) = \begin{cases} 
1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\
-1 & \text{otherwise.} 
\end{cases} \]

Decision Surface of a Perceptron

(a) Represents some useful functions
- What weights represent \( g(x_1, x_2) = AND(x_1, x_2) \)?

(b) But some functions not representable
- All not linearly separable
- Therefore, we’ll want networks of these...
Perceptron Training Rule

\[ w_i \leftarrow w_i + \Delta w_i \]

where

\[ \Delta w_i = \eta (t - o)x_i \]

Where:

- \( t = c(\bar{x}) \) is target value
- \( o \) is perceptron output
- \( \eta \) is small constant (e.g., 0.1) called learning rate

Multiclass Decision Rule

- If we have more than two classes:
  - Have a weight vector for each class: \( w_y \)
  - Calculate an activation for each class

\[ \text{activation}_w(x, y) = w_y \cdot f(x) \]

- Highest activation wins

\[ y = \arg \max_y (\text{activation}_w(x, y)) \]
From Logistic Regression to the Perceptron: 2 easy steps!

- **Logistic Regression**: (in vector notation): \( y \) is \( \{0,1\} \)
  \[
  w = w + \eta \sum_j [y_j^* - p(y_j^* | x_j, w)] f(x_j)
  \]

- **Perceptron**: \( y \) is \( \{0,1\} \), \( y(x;w) \) is prediction given \( w \)
  \[
  w = w + [y^* - y(x;w)] f(x)
  \]

Differences?
- Drop the \( \Sigma_j \) over training examples: **online vs. batch learning**
- Drop the distribution: **probabilistic vs. error driven learning**

---

Properties of Perceptrons

- **Separability**: some parameters get the training set perfectly correct
  - Separable
  - Non-Separable

- **Convergence**: if the training is separable, perceptron will eventually converge (binary case)
Problems with the Perceptron

• Noise: if the data isn’t separable, weights might thrash
  – Averaging weight vectors over time can help (averaged perceptron)

• Mediocre generalization: finds a “barely” separating solution

• Overtraining: test / validation accuracy usually rises, then falls
  – Overtraining is a kind of overfitting

Linear Separators

• Which of these linear separators is optimal?
Support Vector Machines

- **Maximizing the margin:** good according to intuition, theory, practice

- SVMs find the separator with max margin

\[
\min_w \frac{1}{2}||w||^2 \\
\forall i, y \ w_y \cdot f(x_i) \geq w_y \cdot f(x_i) + 1
\]

Three Views of Classification

- Naïve Bayes:
  - Parameters from data statistics
  - Parameters: probabilistic interpretation
  - Training: one pass through the data

- Logistic Regression:
  - Parameters from gradient ascent
  - Parameters: linear, probabilistic model, and discriminative
  - Training: one pass through the data per gradient step; regularization essential

- The Perceptron:
  - Parameters from reactions to mistakes
  - Parameters: discriminative interpretation
  - Training: go through the data until accuracy on validation set maxes out
Example

“win the vote”
“win the election”
“win the game”

\[
\begin{array}{|c|c|c|}
\hline
\text{wSPORTS} & \text{wPOLITICS} & \text{wTECH} \\
\hline
\text{BIAS} : & \text{BIAS} : & \text{BIAS} : \\
\text{win} : & \text{win} : & \text{win} : \\
\text{game} : & \text{game} : & \text{game} : \\
\text{vote} : & \text{vote} : & \text{vote} : \\
\text{the} : & \text{the} : & \text{the} : \\
\text{...} & \text{...} & \text{...} \\
\hline
\end{array}
\]

Example

“win the vote”

\[
\begin{array}{|c|c|c|}
\hline
\text{wSPORTS} & \text{wPOLITICS} & \text{wTECH} \\
\hline
\text{BIAS} : & \text{BIAS} : & \text{BIAS} : \\
\text{win} : & \text{win} : & \text{win} : \\
\text{game} : & \text{game} : & \text{game} : \\
\text{vote} : & \text{vote} : & \text{vote} : \\
\text{the} : & \text{the} : & \text{the} : \\
\text{...} & \text{...} & \text{...} \\
\hline
\end{array}
\]
The Multi-class Perceptron Alg.

- Start with zero weights
- Iterate training examples
  - Classify with current weights
    \[ y = \arg \max_y w_y \cdot f(x) \]
    \[ = \arg \max_y \sum_i w_{y,i} \cdot f_i(x) \]
  - If correct, no change!
  - If wrong: lower score of wrong answer, raise score of right answer
    \[ w_y = w_y - f(x) \]
    \[ w_{y^*} = w_{y^*} + f(x) \]