Binary Classification / Perceptron

Nicholas Ruozzi
University of Texas at Dallas

Slides adapted from David Sontag and Vibhav Gogate
Supervised Learning

- **Input:** $(x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})$
  - $x^{(m)}$ is the $m^{th}$ data item and $y^{(m)}$ is the $m^{th}$ label

- **Goal:** find a function $f$ such that $f(x^{(m)})$ is a “good approximation” to $y^{(m)}$
  - Can use it to predict $y$ values for previously unseen $x$ values
Supervised Learning

- **Hypothesis space**: set of allowable functions $f : X \rightarrow Y$

- **Goal**: find the “best” element of the hypothesis space

  - How do we measure the quality of $f$?
Examples of Supervised Learning

- Spam email detection
- Handwritten digit recognition
- Stock market prediction
- More?
Regression

Hypothesis class: linear functions $f(x) = ax + b$

How do we measure the quality of the approximation?
Linear Regression

• In typical regression applications, measure the fit using a squared loss function

\[ L(f) = \frac{1}{M} \sum_{m} (f(x^{(m)}) - y^{(m)})^2 \]

• Want to minimize the average loss on the training data

• For 2-D linear regression, the learning problem is then

\[ \min_{a,b} \frac{1}{M} \sum_{m} (ax^{(m)} + b - y^{(m)})^2 \]

• For an unseen data point, \( x \), the learning algorithm predicts \( f(x) \)
Supervised Learning

• **Select a hypothesis space** (elements of the space are represented by a collection of parameters)

• **Choose a loss function** (evaluates quality of the hypothesis as a function of its parameters)

• **Minimize loss function using gradient descent** (minimization over the parameters)

• **Evaluate quality of the learned model using test data** – that is, data on which the model was not trained
Binary Classification

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

• An example with \(n = 2\)
Binary Classification

- Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

- We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

- An example with \(n = 2\)

What is a good hypothesis space for this problem?
Binary Classification

- Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

- We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

- An example with \(n = 2\)

What is a good hypothesis space for this problem?
Binary Classification

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

• An example with \(n = 2\)

In this case, we say that the observations are linearly separable.
Linear Separators

• In $n$ dimensions, a hyperplane is a solution to the equation

$$w^T x + b = 0$$

with $w \in \mathbb{R}^n, b \in \mathbb{R}$

• Hyperplanes divide $\mathbb{R}^n$ into two distinct sets of points (called open halfspaces)

$$w^T x + b > 0$$

$$w^T x + b < 0$$
Binary Classification

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

• An example with \(n = 2\)

In this case, we say that the observations are linearly separable
The Linearly Separable Case

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• Hypothesis space: separating hyperplanes
  \[ f(x) = \text{sign} (w^T x + b) \]

• How should we choose the loss function?
The Linearly Separable Case

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• Hypothesis space: separating hyperplanes

\[
f(x) = \text{sign} (w^T x + b)
\]

• How should we choose the loss function?

• Count the number of misclassifications

\[
\text{loss} = \sum_m |y^{(m)} - \text{sign}(w^T x^{(m)} + b)|
\]

• Tough to optimize, gradient contains no information
The Linearly Separable Case

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• Hypothesis space: separating hyperplanes
  \[ f(x) = \text{sign} \ (w^T x + b) \]

• How should we choose the loss function?

  • Penalize each misclassification by the size of the violation

  \[ \text{perceptron loss} = \sum_{m} \max\{0, -y^{(m)}(w^T x^{(m)} + b)\} \]

  • Modified hinge loss (this loss is convex, but not differentiable)
The Perceptron Algorithm

- Try to minimize the perceptron loss using (sub)gradient descent
Subgradients

• For a convex function $g(x)$, a subgradient at a point $x^0$ is any tangent line/plane through the point $x^0$ that underestimates the function everywhere.
Subgradients

- For a convex function $g(x)$, a subgradient at a point $x^0$ is any tangent line/plane through the point $x^0$ that underestimates the function everywhere.
Subgradients

- For a convex function $g(x)$, a subgradient at a point $x^0$ is any tangent line/plane through the point $x^0$ that underestimates the function everywhere.
Subgradients

For a convex function $g(x)$, a subgradient at a point $x^0$ is any tangent line/plane through the point $x^0$ that underestimates the function everywhere.

If $\overrightarrow{0}$ is a subgradient at $x^0$, then $x^0$ is a global minimum.
The Perceptron Algorithm

• Try to minimize the perceptron loss using (sub)gradient descent
The Perceptron Algorithm

• Try to minimize the perceptron loss using (sub)gradient descent

\[ \nabla_w (\text{perceptron loss}) = - \sum_{m=1}^{M} \left( y^{(m)} x^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right) \]

\[ \nabla_b (\text{perceptron loss}) = - \sum_{m=1}^{M} \left( y^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right) \]
• Try to minimize the perceptron loss using (sub)gradient descent

$$\nabla_w (\text{perceptron loss}) = - \sum_{m=1}^{M} \left( y^{(m)} x^{(m)} 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)$$

$$\nabla_b (\text{perceptron loss}) = - \sum_{m=1}^{M} \left( y^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)$$

Is equal to zero if the $m^{th}$ data point is correctly classified and one otherwise.
The Perceptron Algorithm

• Try to minimize the perceptron loss using (sub)gradient descent

\[
\begin{align*}
    w^{(t+1)} &= w^{(t)} + \gamma_t \sum_{m=1}^{M} \left( y^{(m)} x^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right) \\
    b^{(t+1)} &= b^{(t)} + \gamma_t \sum_{m=1}^{M} \left( y^{(m)} \cdot 1_{-y^{(m)} f_{w,b}(x^{(m)}) \geq 0} \right)
\end{align*}
\]

• With step size \( \gamma_t \) (also called the learning rate)
Stochastic Gradient Descent

• To make the training more practical, stochastic gradient descent is used instead of standard gradient descent.

• Approximate the gradient of a sum by sampling a few indices (as few as one) uniformly at random and averaging

\[
\nabla_x \left[ \sum_{m=1}^{M} g_m(x) \right] \approx \frac{1}{K} \sum_{k=1}^{K} \nabla_x g_{m_k}(x)
\]

Here, each \( m_k \) is sampled uniformly at random from \( \{1, \ldots, M\} \).

• Stochastic gradient descent converges under certain assumptions on the step size.
Stochastic Gradient Descent

- Setting $K = 1$, we can simply pick a random observation $m$ and perform the following update if the $m^{th}$ data point is misclassified

\[
\begin{align*}
    w^{(t+1)} &= w^{(t)} + \gamma_t y^{(m)} x^{(m)} \\
    b^{(t+1)} &= b^{(t)} + \gamma_t y^{(m)}
\end{align*}
\]

and

\[
\begin{align*}
    w^{(t+1)} &= w^{(t)} \\
    b^{(t+1)} &= b^{(t)}
\end{align*}
\]

otherwise

- Sometimes, you will see the perceptron algorithm specified with $\gamma_t = 1$ for all $t$
Applications of Perceptron

• Spam email classification
  • Represent emails as vectors of counts of certain words (e.g., sir, madam, Nigerian, prince, money, etc.)
  • Apply the perceptron algorithm to the resulting vectors
  • To predict the label of an unseen email
    • Construct its vector representation, $x'$
    • Check whether or not $w^T x' + b$ is positive or negative
Perceptron Learning Drawbacks

- No convergence guarantees if the observations are not linearly separable
- Can overfit
  - There can be a number of perfect classifiers, but the perceptron algorithm doesn’t have any mechanism for choosing between them
What If the Data Isn’t Separable?

- Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)
- We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)
- An example with \(n = 2\)

What is a good hypothesis space for this problem?
What If the Data Isn’t Separable?

• Input \((x^{(1)}, y^{(1)}), \ldots, (x^{(M)}, y^{(M)})\) with \(x^{(m)} \in \mathbb{R}^n\) and \(y^{(m)} \in \{-1, +1\}\)

• We can think of the observations as points in \(\mathbb{R}^n\) with an associated sign (either +/- corresponding to 0/1)

• An example with \(n = 2\)

What is a good hypothesis space for this problem?
Adding Features

- Perceptron algorithm only works for linearly separable data

Can add features to make the data linearly separable over a larger space!

Essentially the same as higher order polynomials for linear regression!
Adding Features

• The idea:

  • Given the observations $x^{(1)}, \ldots, x^{(M)}$, construct a feature vectors $\phi(x^{(1)}), \ldots, \phi(x^{(M)})$

  • Use $\phi(x^{(1)}), \ldots, \phi(x^{(M)})$ instead of $x^{(1)}, \ldots, x^{(M)}$ in the learning algorithm

  • Goal is to choose $\phi$ so that $\phi(x^{(1)}), \ldots, \phi(x^{(M)})$ are linearly separable

  • Learn linear separators of the form $w^T \phi(x)$ (instead of $w^T x$)

• **Warning**: more expressive features can lead to overfitting!
Adding Features: Examples

• \( \phi(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \)

  • This is just the input data, without modification

• \( \phi(x_1, x_2) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1^2 \\ x_2^2 \end{bmatrix} \)

  • This corresponds to a second degree polynomial separator, or equivalently, elliptical separators in the original space
Adding Features

\[(x_1 - 1)^2 + (x_2 - 1)^2 - 1 \leq 0\]
Adding Features

\[ 1x_1^2 + 1x_2^2 - 2x_1 - 2x_2 - 2 \leq 0 \]
Support Vector Machines

• How can we decide between two perfect classifiers?

• What is the practical difference between these two solutions?