Linear Classifiers

Michael Collins

January 18, 2012

Today's Lecture

- Binary classification problems
- Linear classifiers
- The perceptron algorithm

Classification Problems: An Example

- Goal: build a system that automatically determines whether an image is a human face or not
- Each image is 100×100 pixels, where each pixel takes a grey-scale value in the set $\{0, 1, 2, \ldots, 255\}$
- We represent an image as a point $\underline{x} \in \mathbb{R}^d$, where $d = 100^2 = 10000$
- We have $n = 50$ training examples, where each training example is an **input** point $\underline{x} \in \mathbb{R}^{10000}$ paired with a **label** y where $y = +1$ if the training example contains a face, $y = -1$ otherwise

Binary Classification Problems

- Goal: Learn a function $f : \mathbb{R}^d \rightarrow \{-1, +1\}$
- We have n training examples

$$\{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \ldots, (\underline{x}_n, y_n)\}$$

- Each $\underline{x}_i$ is a point in $\mathbb{R}^d$
- Each y_i is either $+1$ or -1

Supervised Learning Problems

- Goal: Learn a function $f : \mathcal{X} \to \mathcal{Y}$
- We have n training examples

  $$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

  where each $x_i \in \mathcal{X}$, and each $y_i \in \mathcal{Y}$
- Often (not always) $\mathcal{X} = \mathbb{R}^d$ for some integer d
- Some possibilities for $\mathcal{Y}$:
  - $\mathcal{Y} = \{-1, +1\}$ (binary classification)
  - $\mathcal{Y} = \{1, 2, \ldots, k\}$ for some $k > 2$ (multi-class classification)
  - $\mathcal{Y} = \mathbb{R}$ (regression)

A Second Example: Spam Filtering

- Goal: build a system that predicts whether an email message is spam or not
- Training examples: $(\underline{x}_i, y_i)$ for $i = 1 \ldots n$
- Each y_i is $+1$ if a message is spam, -1 otherwise.
- Each $\underline{x}_i$ is a vector in $\mathbb{R}^d$ representing a document

What Kind of Solution would Suffice?

- Say we have $n = 50$ training examples. Each pixel can take 256 values. It's possible that some pixel, say pixel number 3, has a different value for every one of the 50 training examples
- Define $x_{t,3}$ for $t = 1 \dots n$ to be the value of pixel 3 on the t'th training example.
- A possible function $f(\underline{x}')$ learned from the training set:

  For $t = 1 \dots 50$:
      If $x'_3 = x_{t,3}$ then return y_t
  Return -1

- **Classifies the training examples perfectly, but does it generalize to new examples?**

Model Selection

- How can we find classifiers that generalize well?
- Key point: we must constrain the set of possible functions that we entertain
- If our set of possible functions is too large, we have a risk of finding a “trivial” function that works perfectly on the training data, but does not generalize well
- If our set of possible functions is too small, we may not even be able to find a function that works well on the training data
- Later in the course we’ll introduce formal (statistical) analysis relating the “size” of a set of functions to the generalization properties of a learning algorithm

Linear Classifiers through the Origin

- Model form:

$$f(\underline{x}; \underline{\theta}) = \mathsf{sign}(\theta_1 x_1 + \ldots + \theta_d x_d) = \mathsf{sign}(\underline{x} \cdot \underline{\theta})$$

- $\underline{\theta}$ is a vector of real-valued parameters
- The functions in our class are parameterized by $\underline{\theta} \in \mathbb{R}^d$
- $\mathsf{sign}(z) = +1$ if $z \geq 0$, and -1 otherwise

Linear Classifiers through the Origin: Geometric Intuition

- Each point $\underline{x}$ is in $\mathbb{R}^d$
- The parameters $\underline{\theta}$ specify a *hyperplane* (linear separator) that separates points into -1 vs. $+1$
- Specifically, the hyperplane is through the origin, with the vector $\underline{\theta}$ as its normal

Linear Classifiers (General Form)

- Model form:
$$f(\underline{x}; \underline{\theta}, \theta_0) = \mathsf{sign}(\underline{x} \cdot \underline{\theta} + \theta_0)$$
- $\underline{\theta}$ is a vector of real-valued parameters, θ_0 is a "bias" parameter
- The functions in our class are parameterized by $\underline{\theta} \in \mathbb{R}^d$ and $\theta_0 \in \mathbb{R}$

Linear Classifiers (General Form): Geometric Intuition

- Each point $\underline{x}$ is in $\mathbb{R}^d$
- The parameters $\underline{\theta}, \theta_0$ specify a *hyperplane* (linear separator) that separates points into -1 vs. $+1$
- Specifically, the hyperplane has the vector $\underline{\theta}$ as its normal, and is at a distance $\theta_0/||\underline{\theta}||$ from the origin, where $||\underline{\theta}||$ is the norm (length) of $\underline{\theta}$.

A Learning Algorithm: The Perceptron

- We've chosen a function class (the class of linear separators through the origin)
- The estimation problem: choose a specific function in this class (i.e., a setting for the parameters $\underline{\theta}$) on the basis of the training set
- One suggestion: find a value for $\underline{\theta}$ that minimizes the number of training errors

$$\hat{E}(\underline{\theta}) = \frac{1}{n}\sum_{t=1}^{n}(1-\delta(y_t, f(\underline{x}_t;\underline{\theta}))) = \frac{1}{n}\sum_{t=1}^{n}\text{Loss}(y_t, f(\underline{x}_t;\underline{\theta}))$$

  where $\delta(y, y')$ is 1 if $y = y'$, 0 otherwise
- Other definitions of Loss are possible

The Perceptron Algorithm

- Initialization: $\underline{\theta} = \underline{0}$ (i.e., all parameters are set to 0)
- Repeat until convergence:
  - For $t = 1 \ldots n$
    1. $y' = \text{sign}(\underline{x}_t \cdot \underline{\theta})$
    2. If $y' \neq y_t$ Then $\underline{\theta} = \underline{\theta} + y_t \underline{x}_t$, Else leave $\underline{\theta}$ unchanged
- "Convergence" occurs when the parameter vector $\underline{\theta}$ remains unchanged for an entire pass over the training set. At that point, all training examples are classified correctly

More about the Perceptron

- Analysis: if there exists a parameter setting $\underline{\theta}$ that correctly classifies all training examples, the algorithm will converge. Otherwise, the algorithm will not converge.
- Intuition: Suppose we make a mistake on $\underline{x}_t$. We then do the update $\underline{\theta}' = \underline{\theta} + y_t \underline{x}_t$. From this:

$$
\begin{aligned}
y_t(\theta' \cdot \underline{x}_t) &= y_t(\underline{\theta} + y_t \underline{x}_t) \cdot \underline{x}_t \\
&= y_t(\underline{\theta} \cdot \underline{x}_t) + y_t^2(\underline{x}_t \cdot \underline{x}_t) \\
&= y_t(\underline{\theta} \cdot \underline{x}_t) + ||\underline{x}_t||^2
\end{aligned}
$$

- Hence $y_t(\theta \cdot \underline{x}_t)$ increases by $||\underline{x}_t||^2$

The Perceptron Convergence Theorem

- Assume their exists some parameter vector $\underline{\theta}^*$, and some $\gamma > 0$ such that for all $t = 1 \ldots n$,

$$y_t(\underline{x}_t \cdot \underline{\theta}^*) \geq \gamma$$

- Assume in addition that for all $t = 1 \ldots n$, $||\underline{x}_t|| \leq R$
- Then the perceptron algorithm makes at most

$$\frac{R^2||\underline{\theta}^*||^2}{\gamma^2}$$

updates before convergence

A Geometric Interpretation

- Assume their exists some parameter vector $\underline{\theta}^*$, and some $\gamma > 0$ such that for all $t = 1 \ldots n$,

$$y_t(\underline{x}_t \cdot \underline{\theta}^*) \geq \gamma$$

- The ratio $\gamma / ||\underline{\theta}^*||$ is the smallest distance of any point $\underline{x}_t$ to the hyperplane defined by $\underline{\theta}^*$