Machine Learning (CSE 446):
The Perceptron Algorithm

Sham M Kakade
© 2019

University of Washington
cse446-staff@cs.washington.edu
Announcements

▶ HW1 posted.
▶ Today: the perceptron algo
Is there a happy medium?

- Decision trees (that aren’t too deep): use relatively few features to classify.
- If we want to use ’all’ the features, then we need a deep (high complexity) decision tree (so overfitting will be more of a concern).
- Is there a ’low complexity’ method (something which is better at keeping the generalization error small) which is able to utilize all the features for prediction?
- One idea:
  A ’linear classifier’: use all features, but weight them.
Input signals come in through dendrites, output signal passes out through the axon.
A “parametric” Hypothesis

▶ Consider using a “linear classifier”:

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

where \( w \in \mathbb{R}^d \) and \( b \) is a scalar.

▶ Here \( \text{sign}(z) \) is the function which is 1 if \( z \geq 0 \) and \(-1\) otherwise.
(Let us say that \( \mathcal{Y} = \{-1, 1\} \).)

▶ Notation: \( x \in \mathbb{R}^d \); \( x[i] \) denotes \( i \)-th coordinate; remember that:

\[
w \cdot x = \sum_{j=1}^{d} w[j] \cdot x[j]
\]

▶ Learning requires us to set the weights \( w \) and the bias \( b \).
▶ (convenience) we can always append 1 to the vector \( x \) through the concatenation

\[
x \leftarrow (x, 1).
\]

This transformation allows us to absorb the bias \( b \) into the last component of the weight vector.
Geometrically...

- What does the decision boundary look like?
What would we like to do?

\[
\tilde{\epsilon}(w) = \frac{1}{N} \sum_{i=1}^{N} 1\{y_i \neq \text{sign}(w \cdot x_i)\}
\]

- **Optimization problem:** find a classifier which minimizes the classification loss on the training dataset \(D\):
  \[
  \min_w \tilde{\epsilon}(w)
  \]

- Problem: (in general) this is an NP-Hard problem.
- Let’s ignore this, and think of an algorithm.

This is the general approach of loss function minimization: find parameters which make our training error “small” (and which also generalizes)
The Perceptron Learning Algorithm (Rosenblatt, 1958)

Let’s think of an **online** algorithm, where we try to update $w$ as we examine each training point $(x_i, y_i)$, one at a time.

Consider the 'epoch based' algorithm:

1. For all $(x, y)$ in our training set:
2. Choose a point $(x, y)$ without replacement from $D$:
   - Let $\hat{y} = \text{sign}(w \cdot x)$
   - If $\hat{y} = y$, then do not update:
     \[ w_{t+1} = w_t \]
   - If $\hat{y} \neq y$,
     \[ w_{t+1} = w_t + yx \]

_return to the first step._

**When to stop?**
Parameters and Convergence

This is the first supervised algorithm we have seen with (no-trivial) real valued parameters, $w$.

- The perceptron learning algorithm’s sole hyperparameter is $E$, the number of epochs (passes over the training data). How should we tune $E$? Use a Dev set.
- Basic question: will the algorithm converge or not?
  - Can you say what has to occur for the algorithm to converge? (Novikoff, 1962)
  - Can we understand when it will *never* converge? (Minsky and Papert, 1969)
When does the perceptron not converge?
Linear Separability

A dataset \( D = \{(x_n, y_n)\}_{n=1}^{N} \) is **linearly separable** if there exists some linear classifier (defined by \( w, b \)) such that, for all \( n \), \( y_n = \text{sign}(w \cdot x_n + b) \).
The Perceptron Convergence

- Again taking $b = 0$ (absorbing it into $w$).
- Margin def: Suppose the data are linearly separable, and all data points are $\gamma$ away from the separating hyperplane. Precisely, there exists a $w_*$, which we can assume to be of unit norm (without loss of generality), such that for all $(x, y) \in D$.

$$y (w_* \cdot x) \geq \gamma$$

$\gamma$ is the margin.

**Theorem:** (Novikoff, 1962) Suppose the inputs bounded such that $\|x\| \leq R$. Assume our data $D$ is linearly separable with margin $\gamma$. Then the perceptron algorithm will make at most $\frac{R^2}{\gamma^2}$ mistakes.

(This implies that at most $O\left(\frac{N}{\gamma^2}\right)$ updates, after which time $w_t$ never changes.)
Proof of the “Mistake Lemma”

- Let $M_t$ be the number of mistakes at time $t$.
  - If we make a mistake using $w_t$ on $(x, y)$, then observe that $yw_t \cdot x \leq 0$.
  - Suppose we make a mistake at time $t$:
    \[ w_\ast \cdot w_t = w_\ast \cdot (w_{t-1} + yx) = w_\ast \cdot w_{t-1} + yw_\ast \cdot x \geq w_\ast \cdot w_{t-1} + \gamma. \]
    Since $w_0 = 0$ and $w_\ast \cdot w_t$ grows by $\gamma$ every time we make a mistake, this implies that $w_\ast \cdot w_t \geq \gamma M_t$.
  - Also, if we make a mistake at time $t$ (using that $yw_t \cdot x \leq 0$),
    \[ \|w_t\|^2 = \|w_{t-1}\|^2 + 2yw_{t-1} \cdot x + \|x\|^2 \leq \|w_{t-1}\|^2 + 0 + \|x\|^2 \leq \|w_{t-1}\|^2 + R^2. \]
    Since $\|w_t\|^2$ grows by $R^2$ on every mistake, this implies $\|w_t\|^2 \leq R^2 M_t$ and so $\|w_t\| \leq R \sqrt{M_t}$.
- Now we have that:
  \[ \gamma M_t \leq w_\ast \cdot w_t \leq \|w_\ast\|\|w_t\| \leq R \sqrt{M_t}. \]
  solving for $M_t$ completes the proof.
