Introduction to Machine Learning

CSE474/574: Lecture 7

Varun Chandola <chandola@buffalo.edu>

9 Feb 2015

Outline

Outline

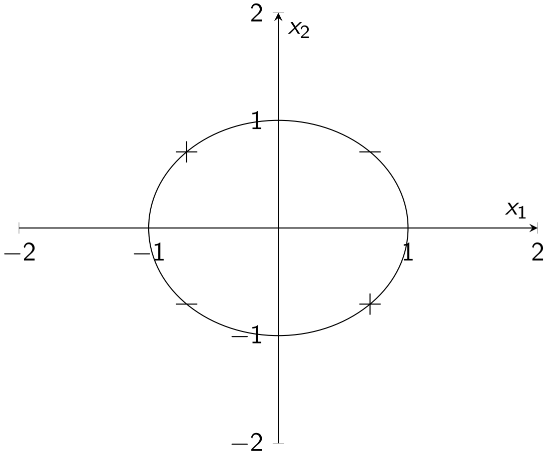
Questions about Winnow?

$\Theta \leftarrow \frac{d}{2}$
$w \leftarrow (1, 1, \ldots, 1)$
for $i = 1, 2, \ldots$ **do**
  if $w^\top x^{(i)} \geq \Theta$ **then**
    $c(x^{(i)}) = 1$
  else
    $c(x^{(i)}) = 0$
  end if
  if $c(x^{(i)}) \neq c_*(x^{(i)})$ **then**
    if $c_*(x^{(i)}) = 1$ **then**
      $\forall j : x_j^{(i)} = 1, w_j \leftarrow \alpha w_j$
    else
      $\forall j : x_j^{(i)} = 1, w_j \leftarrow 0$
    end if
  end if
end for

- Learns k-disjunctive concepts
- When do promotions happen?
  - What happens in a promotion?
- When do eliminations happen?
  - What happens in an elimination?

Outline

Does Perceptron Training Converge?



Convergence Assumptions

1. Linearly separable examples
2. No errors
3. $|\mathbf{x}| = 1$
4. A positive δ gap exists that "contains" the target concept (hyperplane)
   - $(\exists\delta)(\exists\mathbf{v})$ such that $(\forall\mathbf{x})\mathbf{v}^\top\mathbf{x} > c_*(\mathbf{x})\delta$.

Perceptron Convergence Theorem

Theorem

For a set of unit length and linearly separable examples, the perceptron learning algorithm will converge after a finite number of mistakes (at most $\frac{1}{\delta^2}$).

Proof discussed in Minsky's book [2].

Recap

Hypothesis Space, $\mathcal{H}$

- Conjunctive
- Disjunctive
  - Disjunctions of k attributes
- Linear hyperplanes
- $\mathbf{c}_* \in \mathcal{H}$
- $\mathbf{c}_* \notin \mathcal{H}$

Input Space, $\mathbf{x}$

- $\mathbf{x} \in \{0,1\}^d$
- $\mathbf{x} \in \Re^d$

Input Space, y

- $y \in \{0,1\}$
- $y \in \{-1,+1\}$
- $y \in \Re$

Recap

Hypothesis Space, $\mathcal{H}$

- Conjunctive
- Disjunctive
  - Disjunctions of k attributes
- Linear hyperplanes
- $\mathbf{c}_* \in \mathcal{H}$
- $\mathbf{c}_* \notin \mathcal{H}$

Input Space, $\mathbf{x}$

- $\mathbf{x} \in \{0, 1\}^d$
- $\mathbf{x} \in \Re^d$

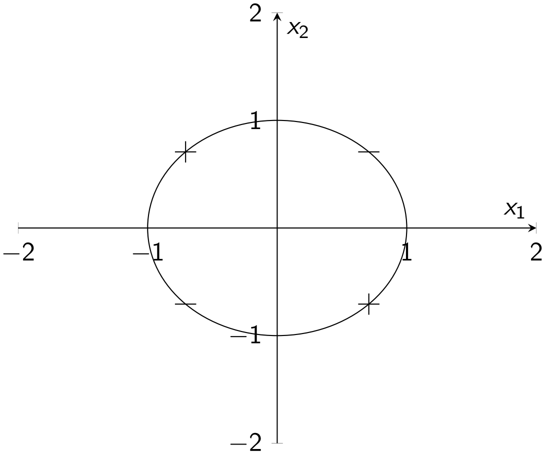
Input Space, y

- $y \in \{0, 1\}$
- $y \in \{-1, +1\}$
- $y \in \Re$

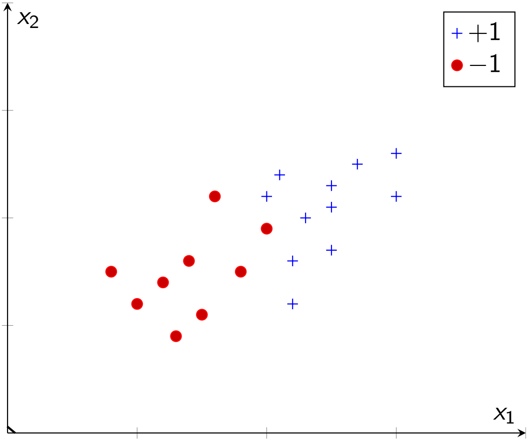
Outline

Target concept $c_* \notin \mathcal{H}$

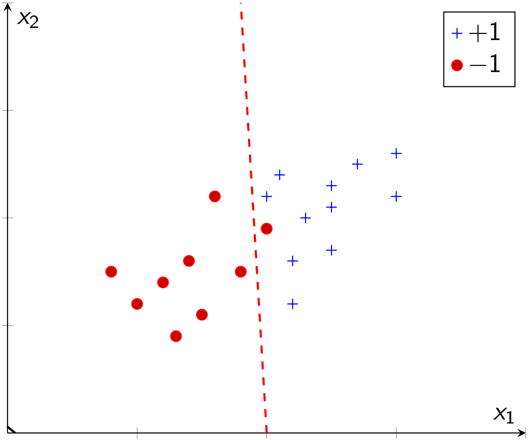
- Expand $\mathcal{H}$?
- Lower expectations
  - *Principle of good enough*



Perceptron Learning in Non-separable Case



Perceptron Learning in Non-separable Case



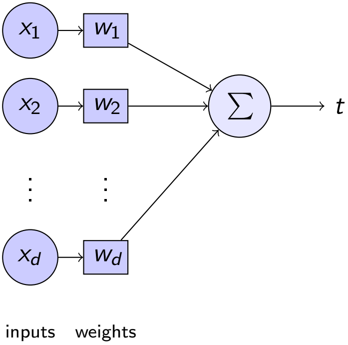
Outline

Gradient Descent and Delta Rule

- Which hyperplane to choose?
- Gives **best performance** on training data
  - Pose as an optimization problem
  - Objective function?
  - Optimization procedure?

Objective Function for Perceptron Learning

- An unthresholded perceptron (a linear unit)



- Training Examples: $\langle \vec{\mathbf{x}}_i, y_i \rangle$
- Weight: $\vec{\mathbf{w}}$

$$E(\vec{\mathbf{w}}) = \frac{1}{2}\sum_i (y_i - \vec{\mathbf{w}}^\top \vec{\mathbf{x}}_i)^2$$

Machine Learning as Optimization Problem[1]

- Learning is optimization
- Faster optimization methods for faster learning
- Let $w \in \Re^d$ and $S \subset \Re^d$ and $f_0(w), f_1(w), \ldots, f_m(w)$ be real-valued functions.
- Standard optimization formulation is:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & f_0(w) \\ \text{subject to} \quad & f_i(w) \leq 0, \; i = 1, \ldots, m. \end{aligned}$$

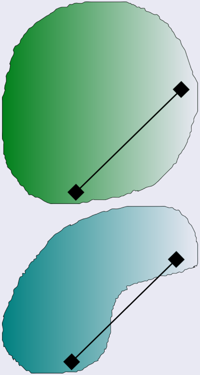
[1] Adapted from http://ttic.uchicago.edu/~gregory/courses/ml2012/lectures/tutorial_optimization.pdf. Also see, http://www.stanford.edu/~boyd/cvxbook/ and http://scipy-lectures.github.io/advanced/mathematical_optimization/.

Solving Optimization Problems

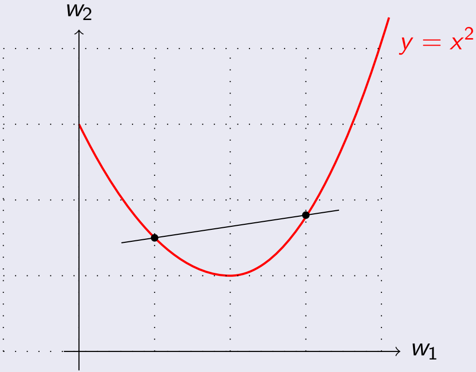
- Methods for **general optimization problems**
  - Simulated annealing, genetic algorithms
- Exploiting *structure* in the optimization problem
  - **Convexity**, Lipschitz continuity, smoothness

Convexity

Convex Sets



Convex Functions



Convex Optimization

- Optimality Criterion

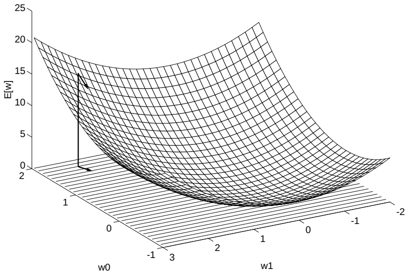
$$\begin{aligned} \underset{w}{\text{minimize}} \quad & f_0(w) \\ \text{subject to} \quad & f_i(w) \leq 0, \; i = 1, \ldots, m. \end{aligned}$$

  where all $f_i(w)$ are **convex functions**.
- w_0 is feasible if $w_0 \in Dom\ f_0$ and all constraints are satisfied
- A feasible w^* is optimal if $f_0(w^*) \leq f_0(w)$ for all w satisfying the constraints

Gradient of a Function

- Denotes the direction of steepest ascent

$$\nabla E(\vec{\mathbf{w}}) = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_d} \end{bmatrix}$$



Finding Extremes of a Single Variable Function

- Set derivative to 0
- Second derivative for minima or maxima

Finding Extremes of a Multiple Variable Function - Gradient Descent

1. Start from any point in variable space
2. Move along the direction of the steepest descent (or ascent)
   - By how much?
   - A learning rate (η)
   - What is the direction of steepest descent?
     - Gradient of E at $\vec{\mathbf{w}}$

Training Rule for Gradient Descent

$$\vec{\mathbf{w}} = \vec{\mathbf{w}} - \eta \nabla E(\vec{\mathbf{w}})$$

For each weight component:

$$w_i = w_i - \eta \frac{\partial E}{\partial w_i}$$

Convergence Guaranteed?

- Error surface contains only one global minimum
- Algorithm *will* converge
  - Examples need not be linearly separable
- η should be *small enough*
- Impact of too large η?
- Too small η?

Issues with Gradient Descent

- Slow convergence
- Stuck in local minima

Stochastic Gradient Descent [1]

- **Update weights after every training example.**
- For sufficiently small η, closely approximates Gradient Descent.

Gradient Descent	**Stochastic Gradient Descent**
Weights updated after summing error over all examples	Weights updated after examining each example
More computations per weight update step	Significantly lesser computations
Risk of local minima	Avoids local minima

References

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel.
Backpropagation applied to handwritten zip code recognition.
Neural Comput., 1(4):541–551, Dec. 1989.

M. L. Minsky and S. Papert.
Perceptrons: An Introduction to Computational Geometry.
MIT Press, 1969.