# 6.036 Recitation Notes (2/17/17)

Helen Zhou, Liang Zhou

## Agenda:

- linear classification
- perceptron
  - ↳ algorithm, example
  - ↳ comments: convergence, easy vs. hard problems
- loss functions
- support vector machines (SVM)
- gradient descent
- Pegasos

## Background

- supervised learning: start with labeled data, extract features to learn a classifier

example: tweet sentiment classification

(Data) "I love how warm it is outside!", (Feature Vector) $x^{(i)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{matrix} \leftarrow \text{"love"} \\ \leftarrow \text{"hate"} \end{matrix}$, (Label) $y^{(i)} = +1$

(Note: Sometimes $\phi(x)$ is used to refer to feature vector for data $x$)

$x^{(i)} \rightarrow \boxed{h(x;\theta)} \rightarrow$ prediction

↑ compare with $y^{(i)}$

# Linear Classification

- We're dealing w/ two-class classification today
- Goal: separate input space into two half-spaces using a linear decision boundary / hyperplane

$$\left.\begin{aligned} h(x;\theta) &= \text{sign}(\theta_1 x_1 + \dots + \theta_d x_d + \theta_0) \\ &= \text{sign}(\theta \cdot x + \theta_0) \\ &= \begin{cases} +1, & \theta\cdot x + \theta_0 > 0 \\ -1, & \theta \cdot x + \theta_0 \le 0 \end{cases} \end{aligned}\right\}$$

defines the set of linear classifiers parameterized by $\theta$ and $\theta_0$

decision boundary: $\theta \cdot x + \theta_0 = 0$

↳ $d=2 \Rightarrow$ 2D, line

$d=3 \Rightarrow$ 3D, plane

# Perceptron

- mistake-driven online learning algorithm for linear classification

## Algorithm:

Input: Training examples $S_n = \{(x^{(i)}, y^{(i)})\}$, $i=1,\dots,n$,

Number of epochs $T$

↑ times you iterate through all of the training examples

Output: $\theta, \theta_0$

Procedure: initialize $\theta, \theta_0 = 0$ (vector; constant term/"bias")

for $t = 1, \dots, T$: ← (or can run until convergence, when all points have been correctly classified)

  for $i = 1, \dots, n$:

    if $y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \le 0$:

      $\theta \leftarrow \theta + y^{(i)} x^{(i)}$

      $\theta_0 \leftarrow \theta_0 + y^{(i)}$

return $\theta, \theta_0$

## Perceptron Alg. Example:

Consider the set of linear classifiers without offset ($\theta_0 = 0$).
Use perceptron to find $\theta$ that linearly separates the data $S_n$.

$S_n$:

| $i$ | $x^{(i)}$ | $y^{(i)}$ |
|---|---|---|
| 1 | (5, 1) | +1 |
| 2 | (1, 3) | −1 |
| 3 | (−1, 1) | −1 |

1) initialize $\theta = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

2) iterate through $S_n$:

$\underline{i=1}$: $y^{(1)}(\theta \cdot x^{(1)}) = 0 \leq 0$

$\Rightarrow \theta \leftarrow \theta + \binom{5}{1} = \binom{5}{1}$

$\underline{i=2}$: $y^{(2)}(\theta \cdot x^{(2)}) = (-1)\left(\binom{5}{1} \cdot \binom{1}{3}\right) \leq 0$

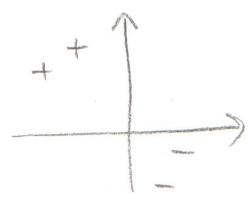
$\Rightarrow \theta \leftarrow \theta + (-1)\binom{1}{3} = \binom{4}{-2}$

$\underline{i=3}$: $y^{(3)}(\theta \cdot x^{(3)}) = (-1)\left(\binom{4}{-2} \cdot \binom{-1}{1}\right) = 6 \not\leq 0$

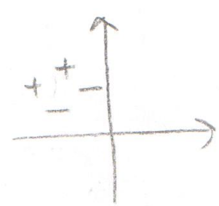
... iterate through all the points again to see it converged

## Notes:

- if data linearly separable, perceptron finds a boundary ← if not, averaging can give a reasonable estimate
- intuition:



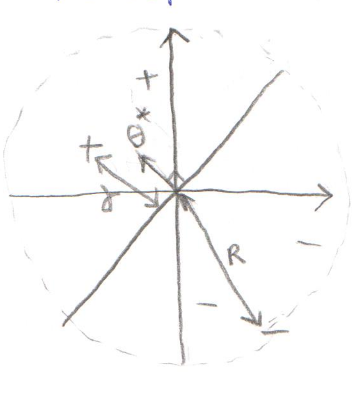

easier




harder

- larger separation
  $\Rightarrow$ more "wiggle room"
  $\Rightarrow$ "easier"

## Perceptron Convergence:




For linearly separable data through the origin, $\exists\, \theta^*$ that separates this data.

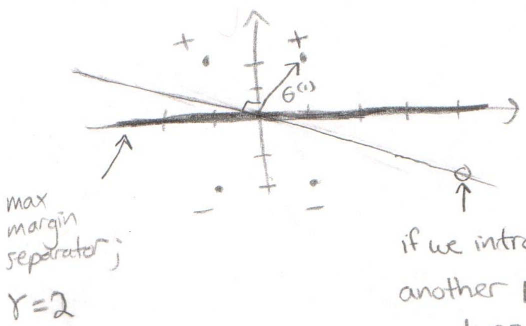
<u>Perceptron Convergence Thm:</u>

Perceptron converges in $\leq \left(\frac{R}{\gamma}\right)^2$ mistakes.

* To see what happens w/ linear classifier w/ offset, consider expanded definition of $\theta$ to include $\theta_0$, and $x$ with a constant bias.

# Perceptron convergence example:

| $i$ | $x^{(i)}$ | $y^{(i)}$ |
|---|---|---|
| 1 | (1,2) | +1 |
| 2 | (-1,2) | +1 |
| 3 | (-1,-2) | -1 |
| 4 | (1,-2) | -1 |




if we introduce another point here, R increases, as does the number of mistakes until convergence. (note $\gamma$ stays the same)

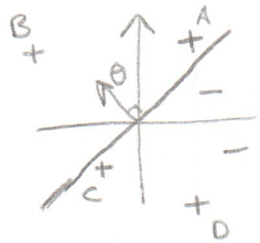
Before: $\left(\frac{R}{\gamma}\right)^2 = \left(\frac{\sqrt{5}}{2}\right)^2 = \frac{5}{4} \rightarrow 1$ step

After: $\left(\frac{R}{\gamma}\right)^2 = \left(\frac{\sqrt{20}}{2}\right)^2 = \frac{20}{4} = 5 \rightarrow >1$ step

# Loss

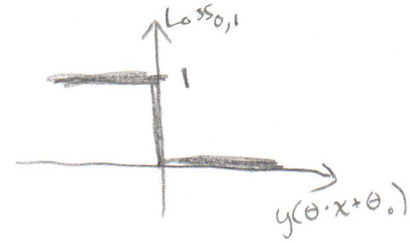
**Zero-one Loss:** $\text{Loss}_{0,1}(y(\theta\cdot x+\theta_0)) = [\![ y(\theta\cdot x+\theta_0) \le 0 ]\!]$

↳ not very sensitive (points near decision boundary have same loss as those further from the decision boundary)

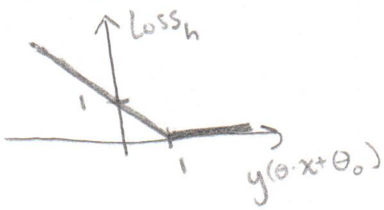



A & B both have loss = 0

C & D both have loss = 1




**Hinge Loss:** $\text{Loss}_h(y(\theta\cdot x+\theta_0)) = \max\{0,\ 1-y(\theta\cdot x+\theta_0)\}$




$$= \begin{cases} 1-y(\theta\cdot x+\theta_0) & \text{if } y(\theta\cdot x+\theta_0) \le 1 \\ 0 & \text{otherwise} \end{cases}$$

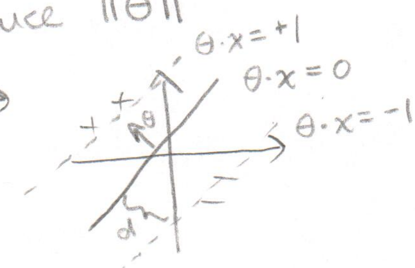
# Support Vector Machines (SVMs)

**Goals:** 1) minimize average Hinge loss on $S_n$

2) push margin boundaries apart

↳ reduce $\|\theta\|$

↳




distance from point to plane: (w/normal $\theta$)

$$\left|\frac{\theta\cdot x}{\|\theta\|}\right|$$

$\Rightarrow$ distance $d = \dfrac{1}{\|\theta\|}$

- note that these goals have opposing effects; have to balance

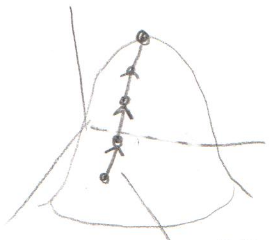
optimization objective:

$$\min \underbrace{\frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h(y^{(i)}(\theta\cdot x^{(i)}))}_{\Rightarrow \text{goal } 1} + \underbrace{\frac{\lambda}{2}\|\theta\|^2}_{\Rightarrow \text{goal } 2}$$

## Gradient Descent

- Stochastic vs. Batch : update per example vs. over sum of loss across batches (or all) of the examples

Basic idea:



size of these steps → learning rate $\eta$

$$\theta \leftarrow \theta \overset{\text{gradient DEscent}}{-} \underbrace{\eta}_{\text{learning rate}} \underbrace{\nabla_\theta}_{\text{gradient}} \overset{\text{objective function}}{f}$$

note: if not convex function, only guaranteed to find a local minimum

## Pegasos example:

$f$ = SVM objective

↪ calculate the update $\theta \leftarrow \theta - \eta \nabla_\theta f$