Online Learning Perceptron Algorithm

Machine Learning – CSE546
Carlos Guestrin (taught by Sameer)
University of Washington
October 23, 2014

Challenge 1: Complexity of Computing Gradients

$$\underbrace{w_i^{(t+1)}}_{\text{each feature } K} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \underbrace{\sum_j}_{\text{over data } N} x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})] \right\}$$

For each t, $O(NK)$

large datasets → problem

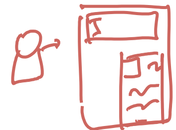
SGD

Challenge 2: Data is streaming

- Assumption thus far: **Batch data**

  All the data points are available

- But, e.g., in click prediction for ads is a streaming data task:
  - User enters query, and ad must be selected:
    - Observe $\mathbf{x}^j$, and must predict y^j



$\vec{x}^t = \langle query, user, ad \rangle$

$\hat{y} = $ click or not; $+1$, -1

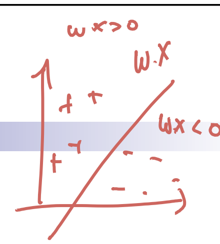
  - User either clicks or doesn't click on ad: $y^t = \{+1, -1\}$
    - Label y^j is revealed afterwards
      - Google gets a reward if user clicks on ad

      if $(y^t = \hat{y})$ Google becomes rich

      if $\hat{y} \neq y^t$
  - Weights must be updated for next time:

    $w^{t+1} \leftarrow w^t + \Delta_t$

Online Learning Problem



- At each time step t:
  - Observe features of data point:
    - Note: many assumptions are possible, e.g., data is iid, data is adversarially chosen… details beyond scope of course

    $X_t = \langle query, user, ad \rangle$ $\langle 1010111000\ldots \rangle$ $x_0 = 1$
  - Make a prediction:
    - Note: many models are possible, we focus on linear models
    - *For simplicity, use vector notation*

    w^t

    $\hat{y} = \begin{cases} +1 & w^t \cdot x^t \geq 0 \\ -1 & w^t \cdot x^t < 0 \end{cases}$

    $w^t \cdot x^t = \sum_{j=0}^{k} w_k^t x_k^t = w_0 + \sum_{j=1}^{k} w_k^t x_k^t$
  - Observe true label:
    - Note: other observation models are possible, e.g., we don't observe the label directly, but only a noisy version… Details beyond scope of course

    Observe $y^t = \begin{cases} +1 \\ -1 \end{cases}$ Mistake if $\hat{y} \neq y^t$
  - Update model:

    $w^{t+1} \leftarrow w^t + \Delta_t$

Rosenblatt 1957

The Perceptron Algorithm [Rosenblatt '58, '62]

- Classification setting: y in {-1,+1}
- Linear model
  - Prediction: $\hat{y} = \text{sign}(w \cdot x)$
- Training:
  - Initialize weight vector: $w^0 = 0$ or random
  - At each time step:
    - Observe features: x^t
    - Make prediction: $\hat{y} = \text{sign}(w \cdot x)$
    - Observe true class: $y^t \leftarrow +1$ or -1
    - Update model:
      - If prediction is not equal to truth

if $\hat{y} = y^t$ $\quad w^{t+1} \leftarrow w^t$

else $\quad w^{t+1} \leftarrow w^t + y^t x^t$

$\text{sign}(w \cdot x) \neq y^t$

if $y^t = 1$ but $\hat{y} = -1$

$w^t \cdot x^t < 0$

$\rightarrow y^t w^t \cdot x^t < 0$

if $y^t = -1$ but $\hat{y} = 1$

$w^t \cdot x^t \geq 0$

$\rightarrow y^t w^t \cdot x^t \leq 0$

Mistake when

$y^t w^t \cdot x^t \leq 0$

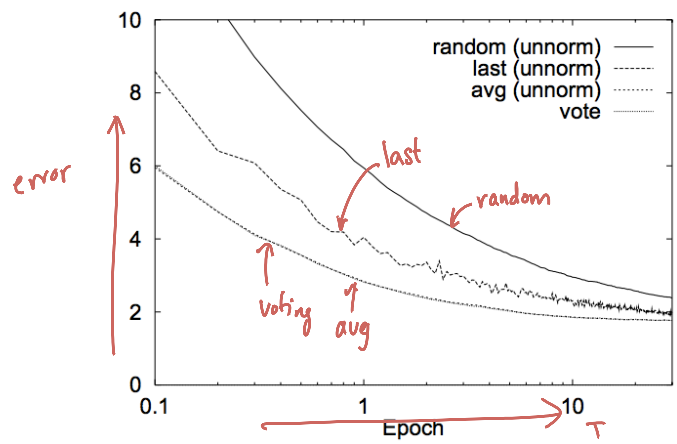
Fundamental Practical Problem for All Online Learning Methods: **Which weight vector to report?**

- Perceptron prediction: $\hat{y} = \text{sign}(w \cdot x)$
- Suppose you run online learning method and want to sell your learned weight vector… Which one do you sell???

T epochs

- Last one? $\hat{w} \leftarrow w^T$ ← too noisy
- Random One - $\hat{w} \leftarrow w^t$ ← too noisy
- Average Weight $\hat{w} = \frac{1}{T+1} \sum_{t=0}^{T} w^t$ ← Good!
- Voting ← we won't cover

Choice can make a huge difference!!



[Freund & Schapire '99]

Mistake Bounds

- Algorithm "pays" every time it makes a mistake:

Google

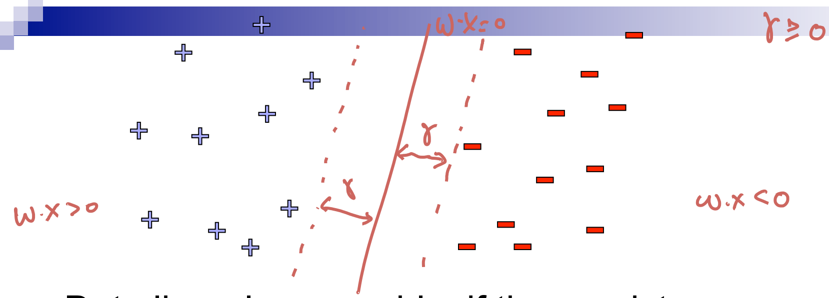
loss function $\boxed{\#\text{ mistakes}}$

- How many mistakes is it going to make?

Mistake Bound

Linear Separability: More formally, Using Margin

$\gamma \geq 0$



- Data linearly separable, if there exists
  - a vector $\exists\, w^*$ $\quad \|w^*\| = 1$
  - a margin $\gamma > 0$
- Such that all points are at least γ away from $w^* x$

$\forall t$ if $y^t = +1$ $\quad w^* x^t \geq \gamma$

$y^t = -1$ $\quad w^* x^t \leq -\gamma$

Linear Sep: $\forall t \quad y^t w^* x^t \geq \gamma$

Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples: $[(x^1, y^1) \ldots (x^T, y^T)]$
  - Each feature vector has bounded norm: $\forall t \ \|x^t\| \le R$
  - If dataset is linearly separable:

    $\exists w^*, \ \|w^*\|=1 \quad (y^t w^* x^t \ge \gamma)$ for $\gamma \ge 0$
- Then the number of mistakes made by the online perceptron on any such sequence is bounded by

  $\left(\frac{R}{\gamma}\right)^2$

  Doesn't depend on T

  Constant number of mistakes

  Independent of data size

Perceptron Proof for Linearly Separable case

- Every time we make a mistake, we get gamma closer to w^*:
  - Mistake at time t: $w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$
  - Taking dot product with w^*: $w^* \cdot w^{t+1} = w^* \cdot (w^t + y^t x^t) = w^* \cdot w^t + \underbrace{y^t w^* \cdot x^t}_{\ge \gamma}$

    $\boxed{w^* w^{t+1} - w^* w^t \ge \gamma}$
  - Thus after m mistakes:

    $w^* w^{t+1} - (w^* w^0) \ge m\gamma$ $\quad \|w^0\|=0$
- Similarly, norm of $w^{(t+1)}$ doesn't grow too fast:
  - $$||\mathbf{w}^{(t+1)}||^2 = ||\mathbf{w}^{(t)}||^2 + \underbrace{2y^{(t)}(\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)})}_{\text{mistake} \le 0} + \underbrace{||\mathbf{x}^{(t)}||^2}_{\le R^2}$$

    $\|w^{t+1}\|^2 - \|w^t\|^2 \le R^2$
  - Thus, after m mistakes:

    $(\|w^{t+1}\|^2 - \|w^0\|^2 \le mR^2)$ $\quad \|w^0\|=0$
- Putting all together:

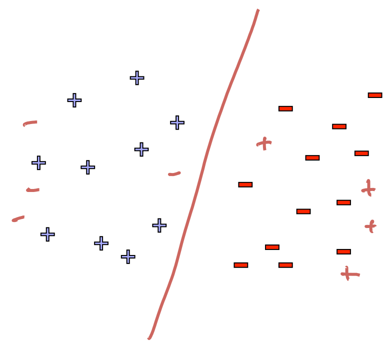
  $m\gamma \le w^* w^{t+1} \le \|w^*\| \|w^{t+1}\| \le \sqrt{m} R$ $\quad \|w^*\| = 1$

  $m\gamma \le \sqrt{m} R \qquad m \le \left(\frac{R}{\gamma}\right)^2$

Beyond Linearly Separable Case

- Perceptron algorithm is super cool!
  - No assumption about data distribution!
    - Could be generated by an oblivious adversary, no need to be iid
  - Makes a fixed number of mistakes, and it's done for ever!
    - Even if you see infinite data
- However, real world not linearly separable
  - Can't expect never to make mistakes again
  - Analysis extends to non-linearly separable case
  - Very similar bound, see Freund & Schapire
  - Converges, but ultimately may not give good accuracy (make many many many mistakes)

(degree of non-linearity



What you need to know

- Notion of online learning → Loss: # mistakes
- Perceptron algorithm
- Mistake bounds and proof
- In online learning, report averaged weights at the end

What's the Perceptron Optimizing?

Machine Learning – CSE546
Carlos Guestrin
University of Washington

October 23, 2013

What is the Perceptron Doing???

- When we discussed logistic regression:
  - Started from maximizing conditional log-likelihood

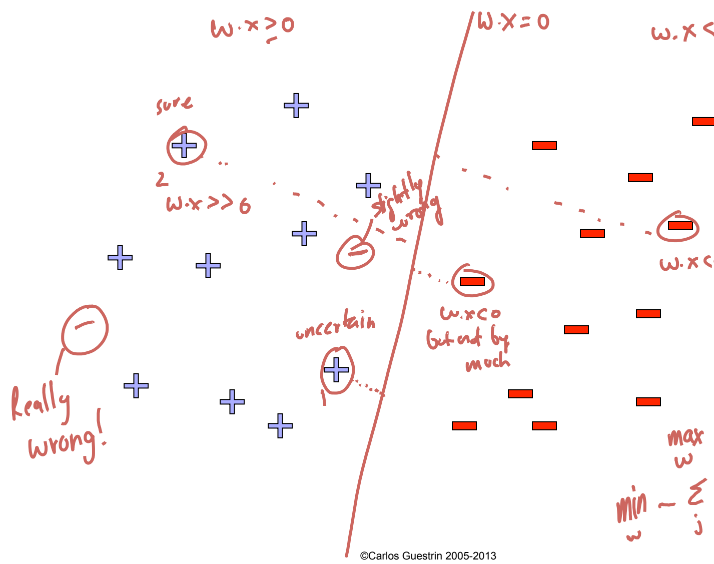
$$\max_w \ell(Y, X, w) = \max_w \log \prod_{i=1}^{N} P(y^i \mid x^i, w) \rightarrow \text{Gradient}$$
$$\hookrightarrow \text{LR algorithm}$$

- When we discussed the Perceptron:
  - Started from description of an algorithm

- What is the Perceptron optimizing????

$$\min_w \sum_t loss(y^t, x^t, w)$$

Perceptron Prediction: Margin of Confidence



Mistake: $y^t w \cdot x^t \le 0$

$\max_w \sum_j y^j w \cdot x^j$

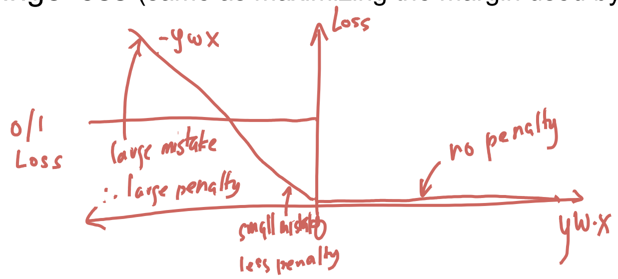
$\min_w - \sum_j y^j w x^j$

Hinge Loss

- Perceptron prediction: $\text{sign}(w \cdot x)$
- Makes a mistake when: $y w \cdot x \le 0$

$$l = \begin{cases} 0 & \text{if } y w \cdot x > 0 \quad \text{0/1 Loss} \\ -y w \cdot x & \text{if } y w \cdot x \le 0 \quad \text{Hinge Loss} \end{cases}$$

- Hinge loss (same as maximizing the margin used by SVMs)

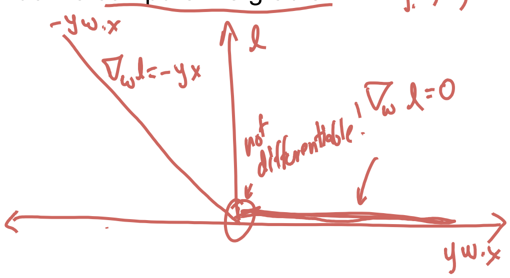


Minimizing hinge loss in Batch Setting

- Given a dataset: $(x^1, y^1) \ldots (x^N, y^N)$
- Minimize average hinge loss:

$$\min_w \frac{1}{N} \sum_{j=1}^{N} \ell(y^j, x^j, w) \quad \begin{cases} 0 & \text{if } y^j w x^j \geq 0 \\ -y^j w x^j & \text{if } y^j w x^j < 0 \end{cases}$$

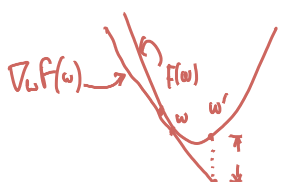
- How do we compute the gradient? $\ell(y^j, x^j, w) = (-y^j w x^j)_+ \qquad (a)_+ = \begin{cases} a & \text{if } a \geq 0 \\ 0 & \text{o.w.} \end{cases}$



"Sub-gradient"

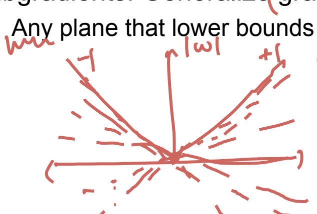
Subgradients of Convex Functions

- Gradients lower bound convex functions:



$$F(w') \geq F(w) + \underbrace{\nabla_w F(w)}^T (w' - w)$$

- Gradients are unique at **w** iff function differentiable at **w**
- Subgradients: Generalize gradients to non-differentiable points:
  - Any plane that lower bounds function:



For $|w| = 0$

$v \in [-1, 1]$

$v \in \partial_w F(w)$

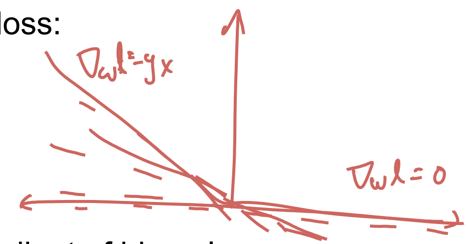
v is subgradient

iff $F(w') \geq F(w) + v^T (w' - w)$

$\nabla_w F(w) \subseteq \partial_w F(w)$

Subgradient of Hinge

- Hinge loss:



- Subgradient of hinge loss:
  - If $y^{(t)}(w.\mathbf{x}^{(t)}) > 0$: $\nabla_w l = 0$
  - If $y^{(t)}(w.\mathbf{x}^{(t)}) < 0$: $\nabla_w l = -yx$
  - If $y^{(t)}(w.\mathbf{x}^{(t)}) = 0$: $\partial_{w_i} l = [-y x_i, 0]$
  - In one line:

$$\partial_w l(w,x,y) = \underbrace{\mathbb{1}(y w.x \leq 0)}_{\text{mistake}} (-y.x)$$

Subgradient Descent for Hinge Minimization

- Given data:
- Want to minimize:
- Subgradient descent works the same as gradient descent:
  - But if there are multiple subgradients at a point, just pick (any) one:

Perceptron Revisited

- Perceptron update:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \mathbb{1}\left[\underbrace{y^{(t)}(\mathbf{w}^{(t)} \cdot \mathbf{x}^{(t)}) \leq 0}_{\text{mistake}}\right] \underbrace{y^{(t)}\mathbf{x}^{(t)}}_{\text{add } yx}$$

no step size (= 1)

- Batch hinge minimization update:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \underbrace{\eta}_{\text{all data}} \frac{1}{N} \sum_{i=1}^{N} \left\{ \mathbb{1}\left[\underbrace{y^{(i)}(\mathbf{w}^{(t)} \cdot \mathbf{x}^{(i)}) \leq 0}_{\text{mistake}}\right] y^{(i)}\mathbf{x}^{(i)} \right\}$$

- Difference? Perceptron is Hinge Loss minimization using SGD and $\eta = 1$

What you need to know

- Perceptron is optimizing hinge loss
- Subgradients and hinge loss
- (Sub)gradient decent for hinge objective

