Challenge 1: Complexity of Computing Gradients in LR

\[ w_i^{(t+1)} = w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_{j=1}^{N} x_i^j [y^j - \hat{P}(Y_i = 1 | x_i^j, w)] \right\} \]

- \text{if lots of data} \ldots \text{N is very large} \rightarrow \text{very slow.}

We talked about SGD instead:

- small change after each data point
Challenge 2: Data is streaming

- Assumption thus far: **Batch data**

- But, e.g., in click prediction for ads is a streaming data task:
  - User enters query, and ad must be selected:
    - Observe $x_j$, and must predict $y_j$ will the user click
    - Observe $x_j$, and must predict $y_j$ which ads have high click prob.
  - User either clicks or doesn’t click on ad:
    - Label $y_j$ is revealed afterwards
      - Google gets a reward if user clicks on ad
        - Reward is loss in classification
      - What’s $\Delta$?

- Weights must be updated for next time:

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
  - Make a prediction:
    - Note: many models are possible, we focus on linear models
      - For simplicity, use vector notation

- 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

- Update model:

©Carlos Guestrin 2005-2013
The Perceptron Algorithm

Classification setting: $y$ in $\{-1,+1\}$

Linear model
- Prediction:
  $y = \text{Sign}(w \cdot x)$

Training:
- Initialize weight vector:
- At each time step:
  - Observe features:
  - Make prediction:
  - Observe true class:
  - Update model:
    - If prediction is not equal to truth, if make a mistake:

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

Suppose you run online learning method and want to sell your learned weight vector... Which one do you sell???

- Last one?
- Random time step?
- Average:
  $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w(t)$
- Voting & more advanced methods: how long has this pattern been good
Choice can make a huge difference!!

Mistake Bounds

- Algorithm “pays” every time it makes a mistake:
  - Loss function for online setting: number of mistakes up to time $T$
  - $\Rightarrow$ Google pays for its mistake

- How many mistakes is it going to make?
Linear Separability: More formally, Using Margin

- Data linearly separable, if there exists
  - a vector $\exists w^*, \|w^*\| = 1$
  - a margin $\gamma > 0$
- Such that:
  - For all points $y = 1$:
    - $y(x) > \gamma$, if $w^* \cdot x > 0$
  - For all points $y = -1$:
    - $y(x) < -\gamma$, if $w^* \cdot x < 0$

Perceptron Analysis: Linearly Separable Case

- Theorem [Block, Novikoff]:
  - Given a sequence of labeled examples: $(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)$
  - Each feature vector has bounded norm: $\forall t, \|x(t)\| \leq 1$
  - If dataset is linearly separable:
    - $\exists w^*, \|w^*\| = 1$
    - $\forall t, y(t) w^* \cdot x(t) > \gamma$, for $y = 1$

- Then the number of mistakes made by the online perceptron on this sequence is bounded by
  $$\left(\frac{\gamma}{\delta}\right)^2$$
  - A constant, does not depend on $T$
  - Dimensionality of $x$!!
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 $w^*$:
    - Thus after $m$ mistakes:
  - Similarly, norm of $w^{(t+1)}$ doesn't grow too fast:
    - Thus, after $m$ mistakes:
  - Putting all together:

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)
What you need to know

- Notion of online learning
- Perceptron algorithm
- Mistake bounds and proof
- In online learning, report averaged weights at the end