Challenge 1: Complexity of Computing Gradients

\[ w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid x^j, \theta)] \right\} \]
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 $x$, and must predict $y$
  - User either clicks or doesn’t click on ad:
    - Label $y$ is revealed afterwards
      - Google gets a reward if user clicks on ad
  - 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 i.i.d, 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:
The Perceptron Algorithm [Rosenblatt '58, '62]

- Classification setting: $y \in \{-1, +1\}$
- Linear model
  - Prediction:
    - Training:
      - Initialize weight vector:
      - At each time step:
        - Observe features:
        - Make prediction:
        - Observe true class:
      - Update model:
        - If prediction is not equal to truth
Fundamental Practical Problem for All Online Learning Methods: **Which weight vector to report?**

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

  - Last one?

Choice can make a huge difference!!

![Graph showing comparison between different weight vectors](Freund & Schapire '99)
Mistake Bounds

- Algorithm “pays” every time it makes a mistake:
  - How many mistakes is it going to make?

Linear Separability: More formally, Using Margin

- Data linearly separable, if there exists
  - a vector
  - a margin
  - Such that
Perceptron Analysis: Linearly Separable Case

Theorem [Block, Novikoff]:
- Given a sequence of labeled examples:
- Each feature vector has bounded norm:
- If dataset is linearly separable:
- Then the number of mistakes made by the online perceptron on any such sequence is bounded by

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:
- $||w^{(t+1)}||^2 = ||w^{(t)}||^2 + 2y^{(t)} (w^{(t)} \cdot x^{(t)}) + ||x^{(t)}||^2$
- 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
What’s the Perceptron Optimizing?

Machine Learning – CSE546
Sham Kakade
University of Washington
October 20, 2016

©Sham Kakade 2016

What is the Perceptron Doing???

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

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

- What is the Perceptron optimizing????
Perceptron Prediction: Margin of Confidence

Hinge Loss

- Perceptron prediction:

  - Makes a mistake when:

    - Hinge loss (same as maximizing the margin used by SVMs)
Minimizing hinge loss in Batch Setting

- Given a dataset:
  - Minimize average hinge loss:
  - How do we compute the gradient?

Subgradients of Convex Functions

- Gradients lower bound convex functions:

  - Gradients are unique at \( w \) iff function differentiable at \( w \)
  - Subgradients: Generalize gradients to non-differentiable points:
    - Any plane that lower bounds function:
Subgradient of Hinge

- Hinge loss:

- Subgradient of hinge loss:
  - If $y(t)(w \cdot x(t)) > 0$:
  - If $y(t)(w \cdot x(t)) < 0$:
  - If $y(t)(w \cdot x(t)) = 0$:
  - In one line:

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:
  \[ w(t+1) \leftarrow w(t) + \begin{cases} y(t) & \text{if} \quad y(t)(w(t) \cdot x(t)) \leq 0 \\ 0 & \text{otherwise} \end{cases} \]

- Batch hinge minimization update:
  \[ w(t+1) \leftarrow w(t) + \eta \frac{1}{N} \sum_{i=1}^{N} \begin{cases} y(i) & \text{if} \quad y(i)(w(t) \cdot x(i)) \leq 0 \\ 0 & \text{otherwise} \end{cases} \]

- Difference?

What you need to know

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