Challenge 1: Complexity of Computing Gradients

\[
\begin{align*}
    w_i^{(t+1)} & \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y^j - P(Y^j = 1 \mid x^j, w)] \right\}
\end{align*}
\]
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 $x_i$, and must predict $y_i$

- User either clicks or doesn't click on ad:
  - Label $y_i$ 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 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:
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](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
Carlos Guestrin
University of Washington
October 23, 2013
©Carlos Guestrin 2005-2013

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

Perceptron prediction:

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:
  \[ F(w') \geq F(w) + \nabla F(w)^T (w' - w) \]
- Gradients are unique at \( w \) iff function differentiable at \( w \)
- Subgradients: Generalize gradients to non-differentiable points:
  - Any plane that lower bounds function:
    \[ V \in \partial F(w) \quad \text{iff} \quad F(w') \geq F(w) + V(w' - w) \]
Subgradient of Hinge

- Hinge loss:

- Subgradient of hinge loss:
  - If $y^{(i)} (w \cdot x^{(i)}) > 0$:
  - If $y^{(i)} (w \cdot x^{(i)}) < 0$:
  - If $y^{(i)} (w \cdot x^{(i)}) = 0$:
  - 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:
  \[
  w^{(t+1)} \leftarrow w^{(t)} + \mathbb{1} \left[ y^{(t)}(w^{(t)} \cdot x^{(t)}) \leq 0 \right] y^{(t)}x^{(t)}
  \]

- Batch hinge minimization update:
  \[
  w^{(t+1)} \leftarrow w^{(t)} + \eta \frac{1}{N} \sum_{i=1}^{N} \left\{ \mathbb{1} \left[ y^{(i)}(w^{(t)} \cdot x^{(i)}) \leq 0 \right] y^{(i)}x^{(i)} \right\}
  \]

- Difference?

What you need to know

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