Online Learning
Perceptron Algorithm

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

©Carlos Guestrin 2005-2013

Challenge 1: Complexity of Computing Gradients

\[ w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y_j - \hat{P}(Y = 1 | x_j, w)] \right\} \]

For each feature, \( O(NK) \)

large datasets \( \rightarrow \) 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 $x_j$, and must predict $y_j$
  - User either clicks or doesn't click on ad:
    - Label $y_j$ 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: $\hat{y} = \text{sign}(w \cdot x)$
- Training:
  - Initialize weight vector: $w^0 = 0$ or random
  - At each time step:
    - Observe features: $x^t$
    - Make prediction: $\hat{y}^t = \text{sign}(w^t \cdot x^t)$
    - Observe true class: $y^t$ is $+1$ or $-1$
    - Update model:
      - If prediction is not equal to truth
        - If $\hat{y}^t = y^t$ then:
          - $w^{t+1} = w^t + \gamma^t x^t$
        - Else:
          - $w^{t+1} = w^t - \gamma^t x^t$

  - If $y^t \cdot w^t \cdot x^t < 0$ then:
    - $\hat{y}^t = 1$ but $y^t = -1$
    - $w^t \cdot x^t < 0$
    - $\rightarrow y^t \cdot w^t \cdot x^t < 0$
  - If $y^t = -1$ but $\hat{y}^t = 1$ then:
    - $w^t \cdot x^t > 0$
    - $\rightarrow y^t \cdot w^t \cdot x^t \leq 0$

©Carlos Guestrin 2005-2013
Fundamental Practical Problem for All Online Learning Methods: \textbf{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???
- Last one?
- Random One \( \hat{w} \leq w^T \) too noisy
- Average Weight \( \hat{w} = \frac{1}{T+1} \sum_{i=0}^{T} w^T \) 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:

- 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
  - All points are at least \( \gamma \) away from \( w^*x \)
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: 
  \[ ||x|| \leq R \]
  - If dataset is linearly separable:
    \[ \exists w^*, ||w^*||=1 \quad \left( y^i w^* x^i \geq \gamma \right) \quad \text{for } Y \geq 0 \]
  - Then the number of mistakes made by the online perceptron on any such sequence is bounded by
    \[ \left( \frac{R^2}{\gamma} \right) \]
    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 \( t+1 \):
    \[ w^{t+1} = w^t + y^t x^t \]
  - Taking dot product with \( w^* \):
    \[ w^* \cdot w^{t+1} = w^* \cdot (w^t + y^t x^t) = w^* \cdot w^t + y^t (w^* \cdot x^t) \]
    \[ = \gamma \]
    \[ w^* \cdot w^{t+1} \geq \gamma \]

- Similarly, norm of \( w^{t+1} \) doesn’t grow too fast:
  - \[ ||w^{t+1}||^2 = ||w^t||^2 + 2y^t (w^t \cdot x^t) + ||x^t||^2 \]
  - \[ \leq R^2 \]

- Thus, after \( m \) mistakes:
  - \[ ||w^{t+1}||^2 \leq m R^2 \]
  - \[ ||w^t||^2 \leq m R^2 \]

- Putting all together:
  - \[ m \gamma \leq w^* \cdot w^{t+1} \leq (||w^t||)(||w^t||) \leq m R \]
  - \[ m \gamma \leq m R \]
  - \[ m \leq \left( \frac{R^2}{\gamma} \right) \]
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

What is the Perceptron Doing???

- When we discussed logistic regression:
  - Started from maximizing conditional log-likelihood
    \[
    \max_{\omega} L(y, x; \omega) = \max_{\omega} \log \prod_{i=1}^{m} p(y_i | x_i, \omega) \Rightarrow \text{Gradient} \Rightarrow \text{LR algorithm}
    \]

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

- What is the Perceptron optimizing???
  \[
  \min_{\omega} \sum_{(x_t, y_t)} \text{loss}(y_t, x_t; \omega)
  \]
Perceptron Prediction: Margin of Confidence

Perceptron prediction:

- Makes a mistake when:

Hinge Loss

- Perceptron prediction: $\text{sign}(w \cdot x)$
- Makes a mistake when:
  - $y \cdot w \cdot x \leq 0$
  - $l = \begin{cases} 0 & \text{if } y \cdot w \cdot x > 0 \\ y \cdot w \cdot x & \text{if } y \cdot w \cdot x \leq 0 \end{cases}$
- 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:
  \[
  \min_{\omega} \frac{1}{N} \sum_{i=1}^{N} \max \left( 0, -y_i \omega^T x_i \right)
  \]

- How do we compute the gradient?

Subgradients of Convex Functions

- Gradients lower bound convex functions:
  \[
  F(\omega') \geq F(\omega) + \langle \nabla F(\omega), \omega' - \omega \rangle
  \]

- Gradients are unique at \(\omega\) iff function differentiable at \(\omega\)

- Subgradients: Generalize gradients to non-differentiable points:
  - Any plane that lower bounds function:
    \[
    \forall \omega \in \{0\}^d, \quad v \in \partial F(\omega) \iff F(\omega) \geq F(\omega) + v^T (\omega' - \omega)
    \]
    \[
    \nabla F(\omega) \subseteq \partial F(\omega)
    \]
Subgradient of Hinge

- Hinge loss:

- Subgradient of hinge loss:
  - If \( y^{(i)}(w \cdot x^{(i)}) > 0 \): \( \nabla w t = 0 \)
  - If \( y^{(i)}(w \cdot x^{(i)}) < 0 \): \( \nabla w t = -y x \)
  - If \( y^{(i)}(w \cdot x^{(i)}) = 0 \): \( \nabla w t = [y x] \)
  - In one line:
    \[
    \nabla w \lambda(w, x, y) = \begin{cases} 
      0 & \text{if } y w x \leq 0 \\
      -y x & \text{if } y w x > 0 
    \end{cases} 
    \]

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:
  \[ \mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \mathbf{1} \cdot \begin{cases} y^{(t)} \mathbf{x}^{(t)} \\ y^{(t)} \mathbf{x}^{(t)} \end{cases} \cdot \mathbf{1} \cdot \begin{cases} 0 \\ y^{(t)} \mathbf{x}^{(t)} \end{cases} \]

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

- Difference?

What you need to know

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