Linear smoother

\[ \hat{y} = \sum_{i=1}^{n} \ell_i(x) y_i \]

\[ \hat{y} = S\ y \]

where \( s_{ij} = s_{ij}(x) \)

e.g. \( s_{ij} = \text{diag}(l_i(x)) \)
Online Learning: LMS and Perceptrons

Partially adapted from slides by Ryan Gabbard and Mitch Marcus (and lots original slides by Lyle Ungar)

Note: supplemental material for today is supplemental; not required!
Why do online learning?

- Batch learning can be expensive for big datasets
  - How expensive is it to compute $(X^TX)^{-1}$ for $X$?
  - Tricky to parallelize

A) $n^3$
B) $p^3$
C) $np^2$
D) $n^2p$
Why do online learning?

- Batch learning can be expensive for big datasets
  - How hard is it to compute \((X^TX)^{-1}\)?
    - \(np^2\) to form \(X^TX\)
    - \(p^3\) to invert
  - Tricky to parallelize inversion
- Online methods are easy in a map-reduce environment
  - They are often clever versions of stochastic gradient descent

Have you seen map-reduce/hadoop?

A) Yes
B) No
Online linear regression

• **Minimize** \( \text{Err} = \sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \)
  - Using stochastic gradient descent
    
    Where we look at each observation \((\mathbf{x}_i, y_i)\) sequentially and decrease its error \(\text{Err}_i = (y_i - \mathbf{w}^T \mathbf{x}_i)^2\)

• **LMS (Least Mean Squares) algorithm**
  - \( \mathbf{w}_{i+1} = \mathbf{w}_i - \eta/2 \frac{d\text{Err}_i}{d\mathbf{w}_i} \)
  - \( \frac{d\text{Err}_i}{d\mathbf{w}_i} = -2 (y_i - \mathbf{w}_i^T \mathbf{x}_i) \mathbf{x}_i \)
    
    \( = -2 r_i \mathbf{x}_i \)

  \[ \mathbf{w}_{i+1} = \mathbf{w}_i + \eta r_i \mathbf{x}_i \]

Note that I is the index for both the iteration and the observation, since there is one update per observation

How do you pick the “learning rate” \(\eta\)?
Online linear regression

• LMS (Least Mean Squares) algorithm

\[ \mathbf{w}_{i+1} = \mathbf{w}_i + \eta \ r_i \mathbf{x}_i \]

• Converges for \( 0 < \eta < \lambda_{\text{max}} \)
  • Where \( \lambda_{\text{max}} \) is the largest eigenvalue of the covariance matrix \( \mathbf{X}^\top \mathbf{X} \)

• Convergence rate is inversely proportional to \( \lambda_{\text{max}}/\lambda_{\text{min}} \) (ratio of extreme eigenvalues of \( \mathbf{X}^\top \mathbf{X} \))
Online learning methods

• Least mean squares (LMS)
  • Online regression -- $L_2$ error

• Perceptron
  • Online SVM -- Hinge loss
Perceptron Learning Algorithm

**Input:** A list $T$ of training examples $\langle x_0, y_0 \rangle \ldots \langle x_n, y_n \rangle$ where $\forall i : y_i \in \{+1, -1\}$

**Output:** A classifying hyperplane $\vec{w}$

Randomly initialize $\vec{w}$;

**while** model $\vec{w}$ makes errors on the training data do

  **for** $\langle x_i, y_i \rangle$ in $T$ do

  Let $\hat{y} = \text{sign}(\vec{w} \cdot \vec{x}_i)$;

  **if** $\hat{y} \neq y_i$ **then**

  $\vec{w} = \vec{w} + y_i \vec{x}_i$;

  **end**

  **end**

**end**

What do we do if error is zero?

Of course, this only converges for linearly separable data
Perceptron Learning Algorithm

For each observation \((y_i, x_i)\)

\[
    w_{i+1} = w_i + \eta \ r_i \ x_i
\]

Where \(r_i = y_i - \text{sign}(w_i^T x_i)\)

and \(\eta = \frac{1}{2}\)

I.e., if we get it right: *no change*

if we got it wrong: \(w_{i+1} = w_i + y_i \ x_i\)
Perceptron Update

If the prediction at \( x_1 \) is wrong, what is the true label \( y_1 \)?

How do you update \( \mathbf{w} \)?
Perceptron Update Example II

\[ w = w + (-1)x \]
Properties of the Simple Perceptron

• You can prove that
  • If it’s possible to separate the data with a hyperplane (i.e. if it’s linearly separable), then the algorithm will converge to that hyperplane.
  • And it will converge such that the number of mistakes $M$ it makes is bounded by
    $$M < \frac{R^2}{\gamma}$$
    where
    $$R = \max_i |x_i|_2$$
    size of biggest $x$
    $$\gamma > y_i \cdot w^T x_i$$
    $> 0$ if separable
Properties of the Simple Perceptron

But what if it isn’t separable?

- Then perceptron is unstable and bounces around
Voted Perceptron

• Works just like a regular perceptron, except you keep track of all the intermediate models you created
• When you want to classify something, you let each of the many models vote on the answer and take the majority

Often implemented after a “burn-in” period
Properties of Voted Perceptron

• Simple!
• Much better generalization performance than regular perceptron
  • Almost as good as SVMs
  • Can use the ‘kernel trick’
• Training is as fast as regular perceptron
• But run-time is slower
  • Since we need n models
Averaged Perceptron

• Return as your final model the *average* of all your intermediate models
• Approximation to voted perceptron
• Again extremely simple!
  • And can use kernels
• Nearly as fast to train and exactly as fast to run as regular perceptron
Many possible Perceptrons

• If point $x_i$ is misclassified
  • $w_{i+1} = w_i + \eta y_i x_i$

• Different ways of picking learning rate $\eta$

• Standard perceptron: $\eta = 1$
  — Guaranteed to converge to the correct answer in a finite time if the points are separable (but oscillates otherwise)

• Pick $\eta$ to maximize the margin ($w_i^T x_i$) in some fashion
  — Can get bounds on error even for non-separable case
Can we do a better job of picking $\eta$?

- Perceptron:

  For each observation $(y_i, x_i)$

  $$w_{i+1} = w_i + \eta \cdot r_i \cdot x_i$$

  where $r_i = y_i - \text{sign}(w_i^T x_i)$

  and $\eta = \frac{1}{2}$

Let’s use the fact that we are actually trying to minimize a loss function.
Passive Aggressive Perceptron

- Minimize the hinge loss at each observation
  - $L(w_i; x_i, y_i) = 0$ if $y_i w_i^T x_i \geq 1$ (loss 0 if correct with margin > 1)
  - $1 - y_i w_i^T x_i$ else

- Pick $w_{i+1}$ to be as close as possible to $w_i$ while still setting the hinge loss to zero
  - If point $x_i$ is correctly classified with a margin of at least 1
    - no change
  - Otherwise
    - $w_{i+1} = w_i + \eta y_i x_i$
    - where $\eta = L(w_i; x_i, y_i)/||x_i||^2$

- Can prove bounds on the total hinge loss
Passive-Aggressive = MIRA

\[ w_{i+1} = w_i + \frac{y_i - w_i \cdot x_i}{\|x_i\|^2} x_i \]

easy to show:

\[ y_i (w_{i+1} \cdot x_i) = y_i (w_i + \frac{y_i - w_i \cdot x_i}{\|x_i\|^2} x_i) \cdot x_i = 1 \]
Margin-Infused Relaxed Algorithm (MIRA)

- **Multiclass**: each class has a prototype vector
  - Note that the prototype \( w \) is like a feature vector \( x \)
- Classify an instance by choosing the class whose prototype vector is *most similar* to the instance
  - *Has the greatest dot product with the instance*
- During training, when updating make the ‘smallest’ change to the prototype vectors which guarantees correct classification by a minimum margin
  - “passive aggressive”
What you should know

- LMS
  - Online regression

- Perceptrons
  - Online SVM
    - Large margin / hinge loss
  - Has nice mistake bounds (for separable case)
    - See wiki
  - In practice use averaged perceptrons
  - Passive Aggressive perceptrons and MIRA
    - Change \( w \) just enough to set it’s hinge loss to zero.

What we didn’t cover:
feature selection