Online Learning Perceptron Algorithm

CSE 446: Machine Learning
Emily Fox
University of Washington
February 10, 2017

Online learning:
Fitting models from streaming data

NOTATION WARNING!!!!!!!!

- To save a lot of writing, here we assume linear features

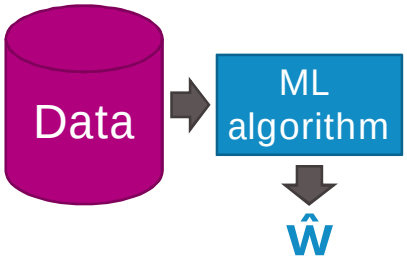
$$h_j(\mathbf{x}) = \mathbf{x}[j]$$

- Things follow straightforwardly for the more general case by replacing **x**[j] with $h_j(\mathbf{x})$

Batch vs online learning

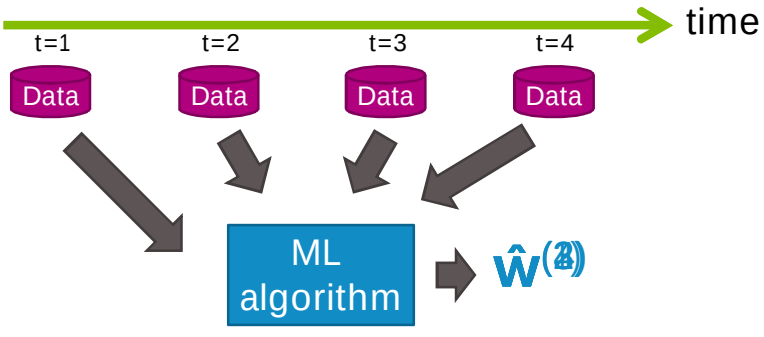
Batch learning

- All data is available at start of training time

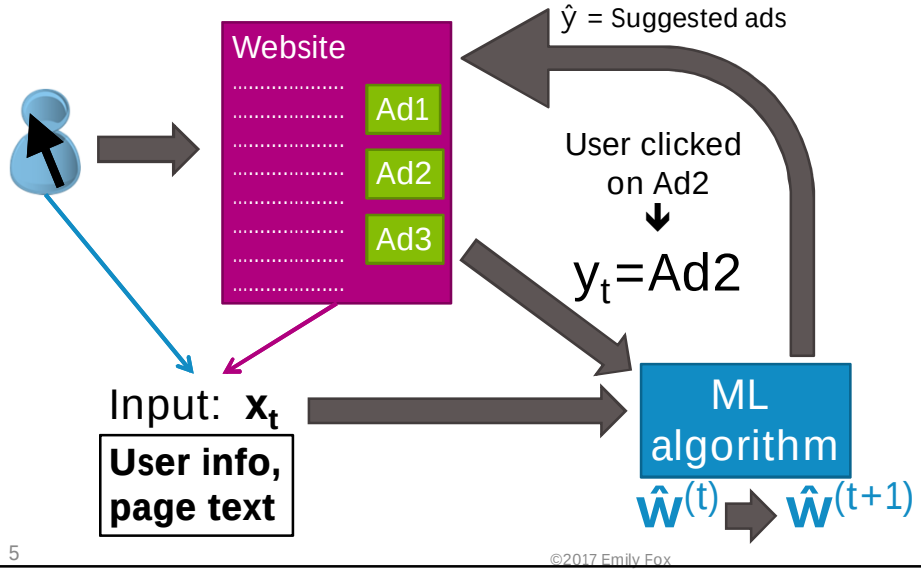


Online learning

- Data arrives (streams in) over time
  - Must train model as data arrives!



Online learning example: **Ad targeting**



Online learning problem

- Data arrives over each time step t:
  - Observe input x_t
    - Info of user, text of webpage
    - Note: many assumptions possible, e.g., data i.i.d., adversarially chosen…Details beyond scope of course
  - Make a prediction $\hat{y}_t$
    - Which ad to show
    - Note: many models possible, we focus on linear models
  - Observe true output y_t
    - Which ad user clicked on
    - 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

$\sum_j w_j^{(t)} x_t[j] \overset{?}{>} 0 \Rightarrow$ "click" $\quad w^{(t)} \cdot x_t \overset{?}{>} 0$

basically equal to $w^{(t)T} x_t$

$y_t \rightarrow$ clicked / not clicked

Need ML algorithm to update coefficients each time step!

$w^{(t+1)} \leftarrow w^{(t)} + \Delta^{(t)}$ ← something

Stochastic gradient ascent can be used for online learning!!!

- init $\mathbf{w}^{(1)}$=0, t=1
- Each time step t:
  - Observe input x_t
  - Make a prediction $\hat{y}_t$
  - Observe true output y_t
  - Update coefficients:

for j=0,…,D

$$w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \frac{\partial \ell_t(\mathbf{w})}{\partial \mathbf{w}_j}$$

7

The perceptron algorithm

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: e.g. $w^{(0)} = 0$ (or smarter)
  - At each time step:
    - Observe features: x_t
    - Make prediction: $\hat{y} = \text{sign}(w^{(t)} \cdot x_t)$
    - Observe true class: y_t
    - Update model:
      - If prediction is not equal to truth

If $\hat{y} = y_t$, $w^{(t+1)} \leftarrow w^{(t)}$

else $w^{(t+1)} \leftarrow w^{(t)} + y_t x_t$

Intuition

If $\hat{y} = y_t$,

$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)}$

else

$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + y_t \mathbf{x}_t$

$$\hat{y} = \text{sign}(\mathbf{w}^{(t)} \cdot \mathbf{x}_t)$$

Why is this a reasonable update rule?

If mistake, e.g. $y_t = +1$ but $w^{(t)} \cdot x_t < 0$ ... wanted $w^{(t)} \cdot x_t > 0$

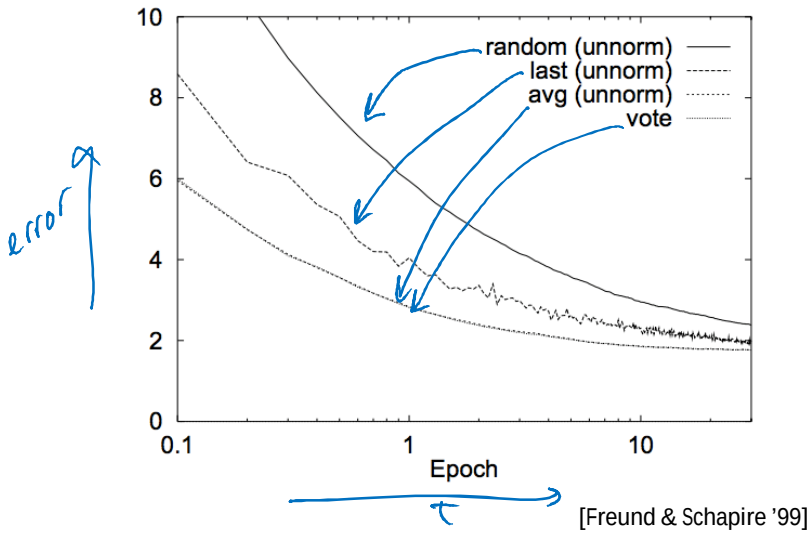
Q: what u $\max_u u \cdot x_t$ $\rightarrow$ $u^* = x_t$

So, by adding x_t to $w^{(t)}$, we increase $w^{(t)} \cdot x_t$ the most (similarly for $y_t = -1$)

Which weight vector to report?

- Practical problem for all online learning methods
- Suppose you run online learning method and want to sell your learned weight vector… Which one do you sell???
- Last one? $w^{(T)}$? no… very. Influenced by last mistake
- Random $w^{(\tau)}$ no
- Average $\hat{w} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$ easy maintain
- Voting + more advanced how long has this param been around?

Choice can make a huge difference!!



[Freund & Schapire '99]

How many mistakes can a perceptron make?

Mistake bounds

Algorithm "pays" every time it makes a mistake:

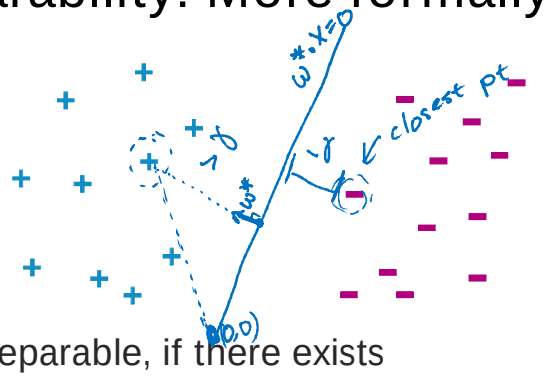
Loss fcn in online setting:
mistakes up to time T

⇒ Google "pays" for its mistakes

How many mistakes is it going to make?

"mistake bound"

Linear separability: More formally, using margin



Data linearly separable, if there exists

- a vector $\exists\ w^*$ $\|w^*\| = 1$
- a margin $\gamma > 0$

such that all pts are γ far away or more from $w^* \cdot x = 0$

$\forall t$ if $y_t = +1$, $w^* \cdot x_t \geq \gamma$
if $y_t = -1$, $w^* \cdot x_t \leq -\gamma$ $\Rightarrow$ $\boxed{y_t (w^* \cdot x_t) \geq \gamma}$

Perceptron analysis: Linearly separable case

Theorem [Block, Novikoff]:

- Given a sequence of labeled examples: $(x_1, y_1), \ldots (x_T, y_T)$
  examples need not be i.i.d. nor random
- Each feature vector has bounded norm:
  $\forall \quad \|x_t\| \leq R$
- If dataset is linearly separable:
  $\exists\ w^* \quad \|w^*\| = 1 \quad$ s.t. $\quad y_t (w^* \cdot x_t) \geq \gamma \quad$ for some $\gamma > 0$

Then the # mistakes made by the online perceptron on this sequence is bounded by

$\left(\frac{R}{\gamma}\right)^2$ Crazy!

← constant... doesn't depend on T or dim(x)

Perceptron proof for linearly separable case

- Every time we make a mistake, we get γ closer to $\mathbf{w}^*$:
  - Mistake at time t: $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + y_t \mathbf{x}_t$
  - Taking dot product with $\mathbf{w}^*$: $w^* \cdot w^{(t+1)} = w^* \cdot (w^{(t)} + y_t x_t) = w^* \cdot w^{(t)} + y_t (w^* \cdot x_t) \geq w^* \cdot w^{(t)} + \gamma$
  - Thus after m mistakes: by induction $w^* \cdot w^{(t+1)} \geq m\gamma$
- Similarly, norm of $\mathbf{w}^{(t+1)}$ doesn't grow too fast:

$$||\mathbf{w}^{(t+1)}||^2 = ||\mathbf{w}^{(t)}||^2 + 2y_t(\mathbf{w}^{(t)} \cdot \mathbf{x}_t) + ||\mathbf{x}_t||^2$$

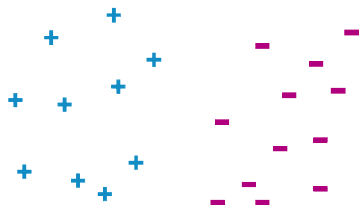
  - Thus, after m mistakes: $||w^{(t+1)}||^2 \leq mR^2$ (mistake < 0; $||x_t|| \leq R$)
- Putting all together:

$$m\gamma \leq w^* \cdot w^{(t+1)} \leq ||w^*||\,||w^{(t+1)}|| \leq \sqrt{m}\,R$$

$$\Rightarrow m\gamma \leq \sqrt{m}\,R \Rightarrow m \leq \left(\frac{R}{\gamma}\right)^2$$

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)



We need features that make data as linearly separable as possible

What is the perceptron optimizing?

What is the perceptron doing???

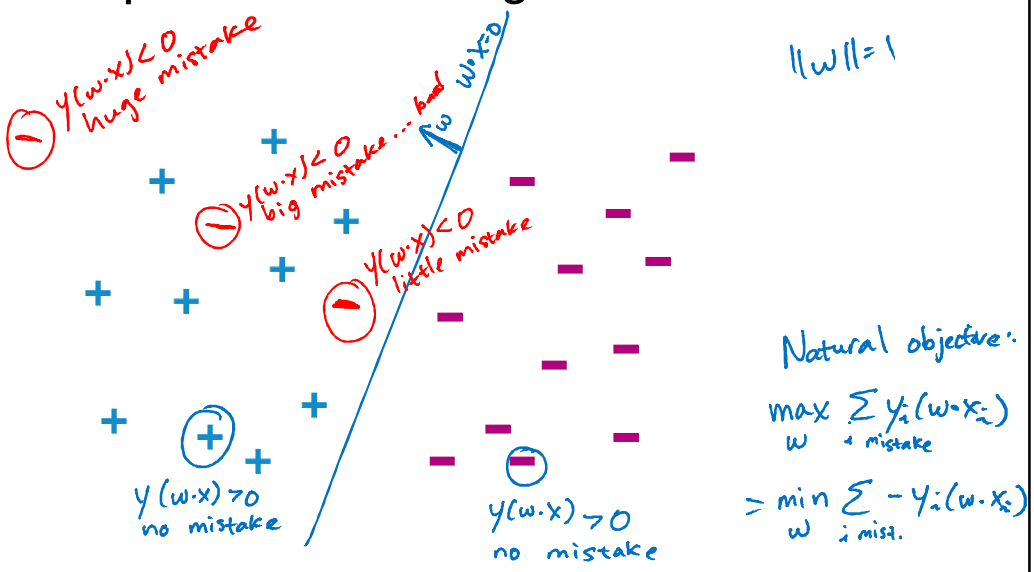
- When we discussed logistic regression:
  - Started from maximizing conditional log-likelihood

$$\max_w \ln P(y|x,w)$$

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

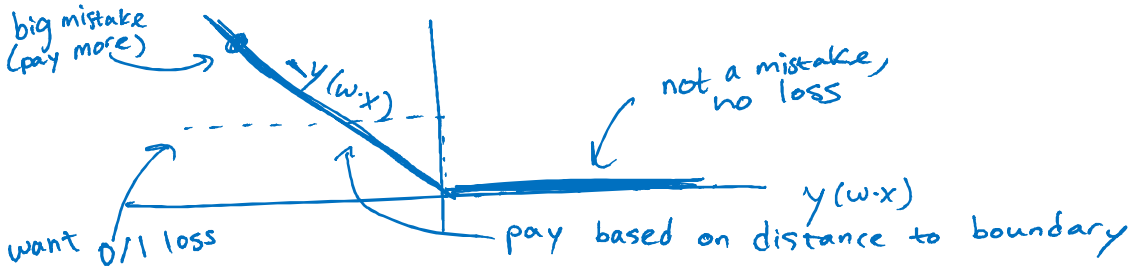
- What is the perceptron optimizing????

Perceptron prediction: Margin of confidence



Hinge loss

- Perceptron prediction: $\text{sign}(w \cdot x)$
- Makes a mistake when: $y(w \cdot x) < 0 \Rightarrow$ loss $L(w,x) = \begin{cases} 0 & y(w \cdot x) > 0 \\ -y(w \cdot x) & \text{o.w.} \end{cases} \Rightarrow (-y(w \cdot x))_+$
- Hinge loss (same as maximizing the margin used by SVMs)

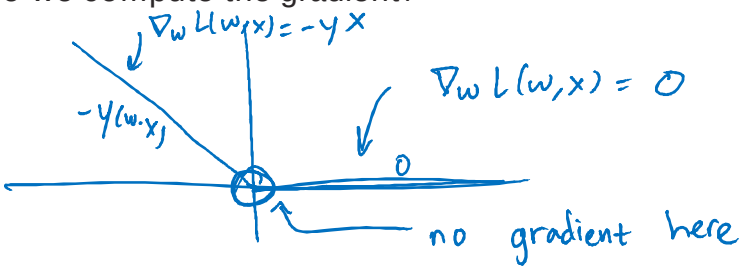


Minimizing hinge loss in batch setting

- Given a dataset: $(x_1, y_1), \ldots, (x_N, y_N)$
- Minimize average hinge loss:

$$\min_w \frac{1}{N} \sum_{i=1}^{N} L(w, x_i) \qquad L(w, x_i) = (-y_i (w \cdot x_i))_+$$

- How do we compute the gradient?



Subgradients of convex functions

- Gradients lower bound convex functions:
- Gradients are unique at **x** if function differentiable at **x**
- Subgradients: Generalize gradients to non-differentiable points:
  - Any plane that lower bounds function:

Subgradient of hinge

- Hinge loss:

- Subgradient of hinge loss:
  - If $y_t(\mathbf{w}.\mathbf{x}_t) > 0$:
  - If $y_t(\mathbf{w}.\mathbf{x}_t) < 0$:
  - If $y_t(\mathbf{w}.\mathbf{x}_t) = 0$:
  - In one line:

Subgradient descent for hinge minimization

- Given data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

- Want to minimize:

$$\frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{w}, \mathbf{x}_i) = \frac{1}{N}\sum_{i=1}^{N} (-y_i(\mathbf{w}\cdot\mathbf{x}_i))_+$$

- 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:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \mathbb{I}\left[y_t(\mathbf{w}^{(t)} \cdot \mathbf{x}_t) \leq 0\right] y_t \mathbf{x}_t$$

- Batch hinge minimization update:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta \frac{1}{N} \sum_{i=1}^{N} \left\{\mathbb{I}\left[y_i(\mathbf{w}^{(t)} \cdot \mathbf{x}_i) \leq 0\right] y_i \mathbf{x}_i\right\}$$

- Difference?

What you need to know

- Notion of online learning
- Perceptron algorithm
- Mistake bounds and proof
- In online learning, report averaged weights at the end
- Perceptron is optimizing hinge loss
- Subgradients and hinge loss
- (Sub)gradient decent for hinge objective