Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Online Learning

Jordan Boyd-Graber
University of Colorado Boulder
LECTURE 21

Slides adapted from Mohri

Motivation

- PAC learning: distribution fixed over time (training and test), IID assumption.
- On-line learning:
  - no distributional assumption.
  - worst-case analysis (adversarial).
  - mixed training and test.
  - Performance measure: mistake model, regret.

General Online Setting

- For $t = 1$ to T:
  - Get instance $x_t \in X$
  - Predict $\hat{y}_t \in Y$
  - Get true label $y_t \in Y$
  - Incur loss $L(\hat{y}_t, y_t)$
- Classification: $Y = \{0, 1\}$, $L(y, y') = |y' - y|$
- Regression: $Y \subset \mathbb{R}, L(y, y') = (y' - y)^2$

General Online Setting

- For $t = 1$ to T:
  - Get instance $x_t \in X$
  - Predict $\hat{y}_t \in Y$
  - Get true label $y_t \in Y$
  - Incur loss $L(\hat{y}_t, y_t)$
- Classification: $Y = \{0, 1\}$, $L(y, y') = |y' - y|$
- Regression: $Y \subset \mathbb{R}, L(y, y') = (y' - y)^2$
- **Objective**: Minimize total loss $\sum_t L(\hat{y}_t, y_t)$

Plan

Experts

Perceptron Algorithm

Online Perceptron for Structure Learning

Prediction with Expert Advice

- For $t = 1$ to T:
  - Get instance $x_t \in X$ and advice $a_{t,i} \in Y, i \in [1, N]$
  - Predict $\hat{y}_t \in Y$
  - Get true label $y_t \in Y$
  - Incur loss $L(\hat{y}_t, y_t)$

Prediction with Expert Advice

- For $t = 1$ to T:
  - Get instance $x_t \in X$ and advice $a_t, i \in Y, i \in [1, N]$
  - Predict $\hat{y}_t \in Y$
  - Get true label $y_t \in Y$
  - Incur loss $L(\hat{y}_t, y_t)$
- **Objective**: Minimize regret, i.e., difference of total loss vs. best expert

$$\text{Regret}(T) = \sum_t L(\hat{y}_t, y_t) - \min_i \sum_t L(a_{t,i}, y_t) \tag{1}$$

Mistake Bound Model

- Define the maximum number of mistakes a learning algorithm L makes to learn a concept c over any set of examples (until it's perfect).

$$M_L(c) = \max_{x_1,\dots,x_T} |\text{mistakes}(L, c)| \tag{2}$$

- For any concept class C, this is the max over concepts c.

$$M_L(C) = \max_{c \in C} M_L(c) \tag{3}$$

Mistake Bound Model

- Define the maximum number of mistakes a learning algorithm L makes to learn a concept c over any set of examples (until it's perfect).

$$M_L(c) = \max_{x_1,\dots,x_T} |\text{mistakes}(L, c)| \tag{2}$$

- For any concept class C, this is the max over concepts c.

$$M_L(C) = \max_{c \in C} M_L(c) \tag{3}$$

- In the expert advice case, assumes some expert matches the concept (realizable)

Halving Algorithm

$H_1 \leftarrow H$;
for $t \leftarrow 1 \dots T$ **do**
  Receive x_t;
  $\hat{y}_t \leftarrow$ Majority$(H_t, \vec{a}_t, x_t)$;
  Receive y_t;
  if $\hat{y}_t \neq y_t$ **then**
    $H_{t+1} \leftarrow \{a \in H_t : a(x_t) = y_t\}$;
return H_{T+1}

Algorithm 1: The Halving Algorithm (Mitchell, 1997)

Halving Algorithm Bound (Littlestone, 1998)

- For a finite hypothesis set

$$M_{\text{Halving}(H)} \leq \lg |H| \tag{4}$$

- After each mistake, the hypothesis set is reduced by at least by half

Halving Algorithm Bound (Littlestone, 1998)

- For a finite hypothesis set

$$M_{\text{Halving}(H)} \leq \lg |H| \tag{4}$$

- After each mistake, the hypothesis set is reduced by at least by half
- Consider the optimal mistake bound opt(H). Then

$$\text{VC}(H) \leq \text{opt}(H) \leq M_{\text{Halving}(H)} \leq \lg |H| \tag{5}$$

- For a fully shattered set, form a binary tree of mistakes with height VC(H)

Halving Algorithm Bound (Littlestone, 1998)

- For a finite hypothesis set

$$M_{\text{Halving}(H)} \leq \lg |H| \tag{4}$$

- After each mistake, the hypothesis set is reduced by at least by half
- Consider the optimal mistake bound opt(H). Then

$$\text{VC}(H) \leq \text{opt}(H) \leq M_{\text{Halving}(H)} \leq \lg |H| \tag{5}$$

- For a fully shattered set, form a binary tree of mistakes with height VC(H)
- What about non-realizable case?

Weighted Majority (Littlestone and Warmuth, 1998)

for $t \leftarrow 1 \dots N$ **do**
  $w_{1,i} \leftarrow 1$;
for $t \leftarrow 1 \dots T$ **do**
  Receive x_t;
  $\hat{y}_t \leftarrow \mathbb{1}\left[\sum_{y_{t,i}=1} w_t \geq \sum_{y_{t,i}=0} w_t\right]$;
  Receive y_t;
  if $\hat{y}_t \neq y_t$ **then**
    for $t \leftarrow 1 \dots N$ **do**
      if $\hat{y}_t \neq y_t$ **then**
        $w_{t+1,i} \leftarrow \beta w_{t,i}$;
      else
        $w_{t+1,i} \leftarrow w_{t,i}$
return w_{T+1}

- Weights for every expert
- Classifications in favor of side with higher total weight ($y \in \{0, 1\}$)
- Experts that are wrong get their weights decreased ($\beta \in [0, 1]$)
- If you're right, you stay unchanged

Weighted Majority (Littlestone and Warmuth, 1998)

for $t \leftarrow 1 \dots N$ **do**
    $w_{1,i} \leftarrow 1$;
for $t \leftarrow 1 \dots T$ **do**
    Receive x_t;
    $\hat{y}_t \leftarrow \mathbb{1}\left[\sum_{y_{t,i}=1} w_t \geq \sum_{y_{t,i}=0} w_t\right]$;
    Receive y_t;
    if $\hat{y}_t \neq y_t$ **then**
        for $t \leftarrow 1 \dots N$ **do**
            if $\hat{y}_t \neq y_t$ **then**
                $w_{t+1,i} \leftarrow \beta w_{t,i}$;
            else
                $w_{t+1,i} \leftarrow w_{t,i}$
return w_{T+1}

- Weights for every expert
- Classifications in favor of side with higher total weight ($y \in \{0, 1\}$)
- Experts that are wrong get their weights decreased ($\beta \in [0, 1]$)
- If you're right, you stay unchanged

Weighted Majority (Littlestone and Warmuth, 1998)

for $t \leftarrow 1 \dots N$ **do**
  $w_{1,i} \leftarrow 1$;
for $t \leftarrow 1 \dots T$ **do**
  Receive x_t;
  $\hat{y}_t \leftarrow \mathbb{1}\left[\sum_{y_{t,i}=1} w_t \geq \sum_{y_{t,i}=0} w_t\right]$;
  Receive y_t;
  if $\hat{y}_t \neq y_t$ **then**
    for $t \leftarrow 1 \dots N$ **do**
      if $\hat{y}_t \neq y_t$ **then**
        $w_{t+1,i} \leftarrow \beta w_{t,i}$;
      else
        $w_{t+1,i} \leftarrow w_{t,i}$
return w_{T+1}

- Weights for every expert
- Classifications in favor of side with higher total weight ($y \in \{0, 1\}$)
- Experts that are wrong get their weights decreased ($\beta \in [0, 1]$)
- If you're right, you stay unchanged

Weighted Majority (Littlestone and Warmuth, 1998)

for $t \leftarrow 1 \dots N$ **do**
  $w_{1,i} \leftarrow 1$;
for $t \leftarrow 1 \dots T$ **do**
  Receive x_t;
  $\hat{y}_t \leftarrow \mathbb{1}\left[\sum_{y_{t,i}=1} w_t \geq \sum_{y_{t,i}=0} w_t\right]$;
  Receive y_t;
  if $\hat{y}_t \neq y_t$ **then**
    for $t \leftarrow 1 \dots N$ **do**
      if $\hat{y}_t \neq y_t$ **then**
        $w_{t+1,i} \leftarrow \beta w_{t,i}$;
      else
        $w_{t+1,i} \leftarrow w_{t,i}$
return w_{T+1}

- Weights for every expert
- Classifications in favor of side with higher total weight ($y \in \{0, 1\}$)
- Experts that are wrong get their weights decreased ($\beta \in [0, 1]$)
- If you're right, you stay unchanged

Weighted Majority

- Let m_t be the number of mistakes made by WM until time t
- Let m_t^* be the best expert's mistakes until time t

$$m_t \leq \frac{\log N + m_t^* \log \frac{1}{\beta}}{\log \frac{2}{1+\beta}} \tag{6}$$

- Thus, mistake bound is $O(\log N)$ plus the best expert
- Halving algorithm $\beta = 0$

Proof: Potential Function

- Potential function is the sum of all weights

$$\Phi_t \equiv \sum_i w_{t,i} \tag{7}$$

- We'll create sandwich of upper and lower bounds

Proof: Potential Function

- Potential function is the sum of all weights

$$\Phi_t \equiv \sum_i w_{t,i} \tag{7}$$

- We'll create sandwich of upper and lower bounds
- For any expert i, we have lower bound

$$\Phi_t \geq w_{t,i} = \beta^{m_{t,i}} \tag{8}$$

Proof: Potential Function

- Potential function is the sum of all weights

$$\Phi_t \equiv \sum_i w_{t,i} \tag{7}$$

- We'll create sandwich of upper and lower bounds
- For any expert i, we have lower bound

$$\Phi_t \geq w_{t,i} = \beta^{m_{t,i}} \tag{8}$$

Weights are nonnegative, so $\sum_i w_{t,i} \geq w_{t,i}$

Proof: Potential Function

- Potential function is the sum of all weights

$$\Phi_t \equiv \sum_i w_{t,i} \tag{7}$$

- We'll create sandwich of upper and lower bounds
- For any expert i, we have lower bound

$$\Phi_t \geq w_{t,i} = \beta^{m_{t,i}} \tag{8}$$

Each error multiplicatively reduces weight by β

Proof: Potential Function (Upper Bound)

- If an algorithm makes an error at round t

$$\Phi_{t+1} \leq \frac{\Phi_t}{2} + \frac{\beta \Phi_t}{2} \tag{9}$$

Proof: Potential Function (Upper Bound)

- If an algorithm makes an error at round t

$$\Phi_{t+1} \leq \frac{\Phi_t}{2} + \frac{\beta \Phi_t}{2} \tag{9}$$

Half (at most) of the experts by weight were right

Proof: Potential Function (Upper Bound)

- If an algorithm makes an error at round t

$$\Phi_{t+1} \leq \frac{\Phi_t}{2} + \frac{\beta \Phi_t}{2} \tag{9}$$

Half (at least) of the experts by weight were wrong

Proof: Potential Function (Upper Bound)

- If an algorithm makes an error at round t

$$\Phi_{t+1} \leq \frac{\Phi_t}{2} + \frac{\beta \Phi_t}{2} = \left[\frac{1+\beta}{2}\right] \Phi_t \tag{9}$$

Proof: Potential Function (Upper Bound)

- If an algorithm makes an error at round t

$$\Phi_{t+1} \leq \frac{\Phi_t}{2} + \frac{\beta\Phi_t}{2} = \left[\frac{1+\beta}{2}\right]\Phi_t \tag{9}$$

- Initially potential function sums all weights, which start at 1

$$\Phi_1 = N \tag{10}$$

Proof: Potential Function (Upper Bound)

- If an algorithm makes an error at round t

$$\Phi_{t+1} \leq \frac{\Phi_t}{2} + \frac{\beta\Phi_t}{2} = \left[\frac{1+\beta}{2}\right]\Phi_t \tag{9}$$

- Initially potential function sums all weights, which start at 1

$$\Phi_1 = N \tag{10}$$

- After m_T mistakes after T rounds

$$\Phi_T \leq \left[\frac{1+\beta}{2}\right]^{m_T} N \tag{11}$$

Weighted Majority Proof

- Put the two inequalities together, using the best expert

$$\beta^{m_T^*} \leq \Phi_T \leq \left[\frac{1+\beta}{2}\right]^{m_T} N \tag{12}$$

Weighted Majority Proof

- Put the two inequalities together, using the best expert

$$\beta^{m_T^*} \leq \Phi_T \leq \left[\frac{1+\beta}{2}\right]^{m_T} N \tag{12}$$

- Take the log of both sides

$$m_T^* \log \beta \leq \log N + m_T \log \left[\frac{1+\beta}{2}\right] \tag{13}$$

Weighted Majority Proof

- Put the two inequalities together, using the best expert

$$\beta^{m_T^*} \leq \Phi_T \leq \left[\frac{1+\beta}{2}\right]^{m_T} N \tag{12}$$

- Take the log of both sides

$$m_T^* \log \beta \leq \log N + m_T \log \left[\frac{1+\beta}{2}\right] \tag{13}$$

- Solve for m_T

$$m_T \leq \frac{\log N + m_T^* \log \frac{1}{\beta}}{\log \left[\frac{2}{1+\beta}\right]} \tag{14}$$

Weighted Majority Recap

- Simple algorithm
- No harsh assumptions (non-realizable)
- Depends on best learner
- Downside: Takes a long time to do well in worst case (but okay in practice)
- Solution: Randomization

Plan

Perceptron Algorithm

- Online algorithm for classification
- Very similar to logistic regression (but 0/1 loss)
- But what can we prove?

Perceptron Algorithm

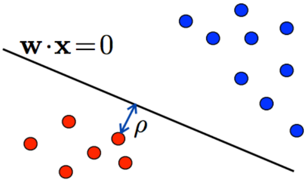
$\vec{w}_1 \leftarrow \vec{0}$;
for $t \leftarrow 1 \dots T$ **do**
  Receive x_t;
  $\hat{y}_t \leftarrow \mathrm{sgn}(\vec{w}_t \cdot \vec{x}_t)$;
  Receive y_t;
  if $\hat{y}_t \neq y_t$ **then**
    $\vec{w}_{t+1} \leftarrow \vec{w}_t + y_t \vec{x}_t$;
  else
    $\vec{w}_{t+1} \leftarrow w_t$;
return w_{T+1}

Algorithm 2: Perceptron Algorithm (Rosenblatt, 1958)

Objective Function

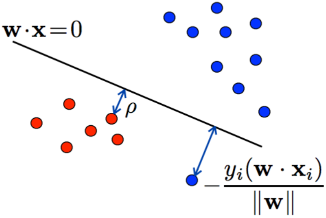
- Optimizes
$$\frac{1}{T}\sum_t \max\left(0, -y_t(\vec{w}\cdot x_t)\right) \tag{15}$$
- Convex but not differentiable

Margin and Errors



- If there's a good margin ρ, you'll converge quickly

Margin and Errors



- If there's a good margin ρ, you'll converge quickly
- Whenever you se an error, you move the classifier to get it right
- Convergence only possible if data are separable

How many errors does Perceptron make?

- If your data are in a R ball and there is a margin

$$\rho \leq \frac{y_t(\vec{v} \cdot \vec{x}_t)}{||v||} \tag{16}$$

for some $\vec{v}$, then the number of mistakes is bounded by R^2/ρ^2

- The places where you make an error are support vectors
- Convergence can be slow for small margins

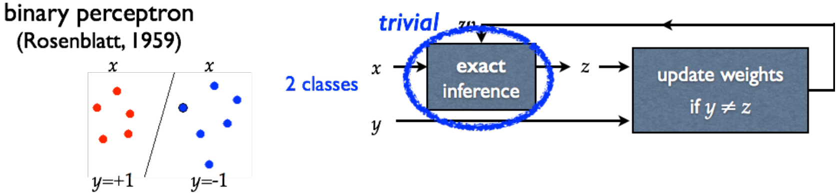
Plan

Experts

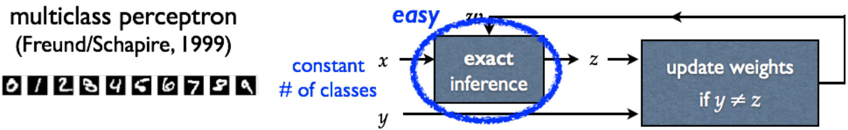
Perceptron Algorithm

Online Perceptron for Structure Learning

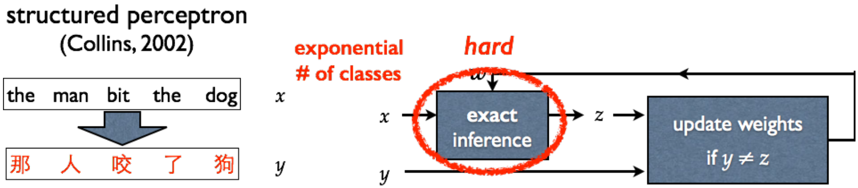
Binary to Structure



Binary to Structure



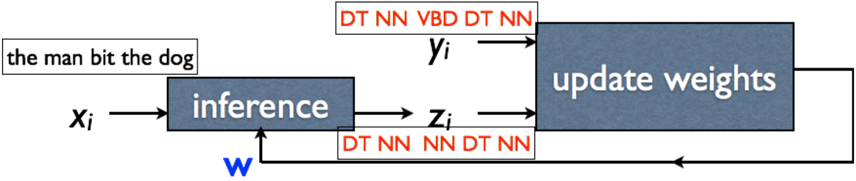
Binary to Structure



Generic Perceptron

- perceptron is the simplest machine learning algorithm
- online-learning: one example at a time
- learning by doing
  - find the best output under the current weights
  - update weights at mistakes

Structured Perceptron



Perceptron Algorithm

Inputs: Training set (x_i, y_i) for $i = 1 \dots n$

Initialization: $\mathbf{W} = 0$

Define: $F(x) = \text{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

Algorithm: For $t = 1 \dots T, i = 1 \dots n$

$z_i = F(x_i)$

If $(z_i \neq y_i)$ $\quad \mathbf{W} \longleftarrow \mathbf{W} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$

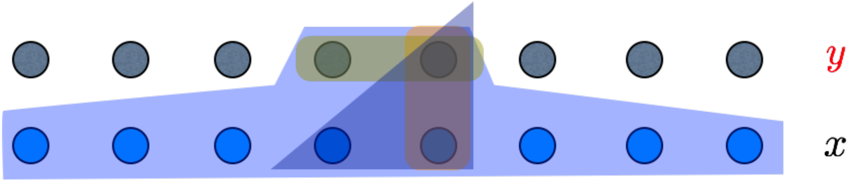
Output: Parameters $\mathbf{W}$

POS Example

- gold-standard: DT NN VBD DT NN y $\Phi(x, y)$
  - the man bit the dog x
- current output: DT NN NN DT NN z $\Phi(x, z)$
  - the man bit the dog x
- assume only two feature classes
  - tag bigrams t_{i-1} t_i
  - word/tag pairs w_i
- weights ++: (NN,VBD) (VBD, DT) (VBD→bit)
- weights −−: (NN, NN) (NN, DT) (NN→bit)

What must be true?

- Finding highest scoring structure must be really fast (you'll do it often)
- Requires some sort of dynamic programming algorithm
- For tagging: features must be local to y (but can be global to x)



Averaging is Good

Inputs: Training set (x_i, y_i) for $i = 1 \dots n$

Initialization: $\mathbf{W}_0 = 0$

Define: $F(x) = \text{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \mathbf{W}$

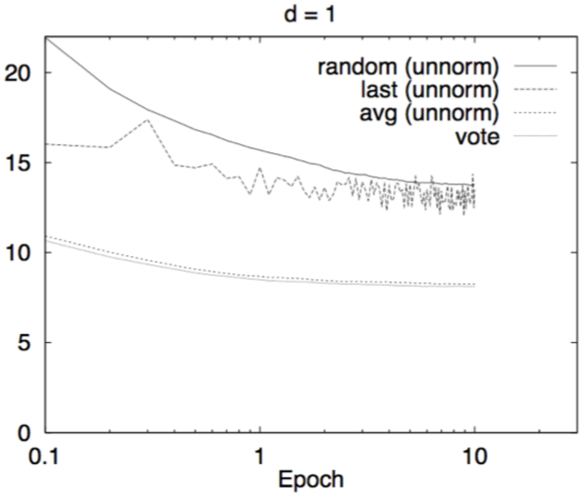
Algorithm: For $t = 1 \dots T, i = 1 \dots n$

$z_i = F(x_i)$

If $(z_i \neq y_i)$ $\mathbf{W}_{j+1} \leftarrow \mathbf{W}_j + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$

Output: Parameters $\mathbf{W} = \sum_j \mathbf{W}_j$

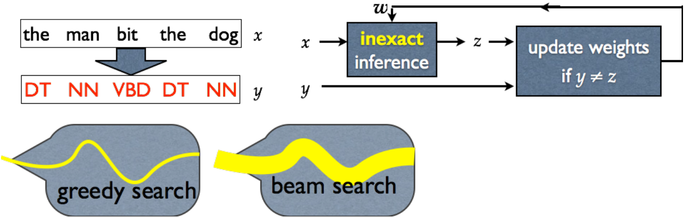
Averaging is Good



Smoothing

- Must include subset templates for features
- For example, if you have feature (t_0, w_0, w_{-1}), you must also have
  - (t_0, w_0); (t_0, w_{-1}); (w_0, w_{-1})

Inexact Search?



- Sometimes search is too hard
- So we use beam search instead
- How to create algorithms that respect this relaxation: track when right answer falls off the beam

Wrapup

- Structured prediction: when one label isn't enough
- Generative models can help with not a lot of data
- Discriminative models are state of the art

Wrapup

- Structured prediction: when one label isn't enough
- Generative models can help with not a lot of data
- Discriminative models are state of the art
- More in Natural Language Processing (at least when I teach it)