Multiplicative Updates
&
the Winnow Algorithm

Lecture 7

Machine Learning
Fall 2015

THE UNIVERSITY OF UTAH

Where are we?

- Still looking at linear classifiers
- Still looking at mistake-bound learning

- We have seen the Perceptron update rule

> - Receive an input $(\boldsymbol{x}_i, y_i)$
> - if $\mathrm{sgn}(\mathbf{w}_t^\top \boldsymbol{x}_i) \neq y_i$:
>
>   Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i\, \boldsymbol{x}_i$

- The Perceptron update is an example of an *additive weight update*

This lecture

- The Winnow Algorithm
- Winnow mistake bound
- Generalizations

This lecture

- The Winnow Algorithm
- Winnow mistake bound
- Generalizations

The setting

- Recall linear threshold units
  - Prediction = +1 if $w^T x \geq \theta$
  - Prediction = -1 if $w^T x < \theta$

- The Perceptron mistake bound is $(R/\gamma)^2$
  - For Boolean functions with n attributes, $R^2 = n$, so basically O(n)

- *Motivating question*:

  Suppose we know that even though the number of attributes is n, the number of relevant attributes is k, which is much less than n

  Can we improve the mistake bound?

Learning when irrelevant attributes abound

Example

- Suppose we know that the true concept is a disjunction of only a small number of features
  - Say only x_1 and x_2 are relevant

- The elimination algorithm will work:
  - Start with h(x) = $x_1 \vee x_2 \vee \cdots \vee x_{1024}$
  - Mistake on a negative example: Eliminate all attributes in the example from your hypothesis function h
    - Suppose we have an example x_{100} = 1, x_{301} = 1, label = -1
    - Simple update: just eliminate these two variables from the function
  - Will never make mistakes on a positive example. Why?

- Makes O(n) updates

- But we know that our function is a k-disjunction
  - And there are only C(n, k) · $2^k \approx n^k 2^k$ k-disjunctions
  - The Halving algorithm will make k log(n) mistakes
  - Can we realize this bound with an efficient algorithm?

Multiplicative updates

- Let's use linear classifiers with a different update rule
  - Perceptron will also make O(n) mistakes

- The idea: Weights should be promoted and demoted via multiplicative, rather than additive, updates

The Winnow algorithm

Littlestone 1988

Given a training set D = $\{(x, y)\}$, $x \in \Re^n$, $y \in \{-1,1\}$

1. Initialize: $\mathbf{w} = (1,1,1,1\ldots,1) \in \Re^n$, $\theta = n$

2. For each training example (x, y):
   – Predict $y' = \text{sgn}(\mathbf{w}^T x - \theta)$
   – If $y = +1$ and $y' = -1$ then:

Promotion
   • Update each weight $\mathbf{w}_i \leftarrow 2\mathbf{w}_i$ *only* for those features x_i that are 1

   Else if $y = -1$ and $y' = +1$ then:

Demotion
   • Update each weight $\mathbf{w}_i \leftarrow \mathbf{w}_i/2$ *only* for those features x_i that are 1

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize $\theta = 1024$, $\mathbf{w} = (1,1,1,1\ldots,1)$

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize θ = 1024, **w** = (1,1,1,1...,1)

Example	Prediction	Error?	Weights
x=(1,1,1,...,1), y=+1	$\mathbf{w}^\mathsf{T} x \geq \theta$	No	**w** = (1,1,1,1...,1)

Example run of the algorithm

$f = x_1 \lor x_2 \lor x_{1023} \lor x_{1024}$ Initialize $\theta = 1024$, $\mathbf{w} = (1,1,1,1\ldots,1)$

Example	Prediction	Error?	Weights
x=(1,1,1,…,1), y=+1	$\mathbf{w}^\mathsf{T} x \geq \theta$	No	$\mathbf{w} = (1,1,1,1\ldots,1)$
x=(0,0,0,…,0), y=−1	$\mathbf{w}^\mathsf{T} x < \theta$	No	$\mathbf{w} = (1,1,1,1\ldots,1)$

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize θ = 1024, $\mathbf{w}$ = (1,1,1,1…,1)

Example	Prediction	Error?	Weights
x=(1,1,1,…,1), y=+1	$\mathbf{w}^\mathsf{T}x \geq \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(0,0,0,…,0), y=−1	$\mathbf{w}^\mathsf{T}x < \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(0,0,1,1,1,…,0), y=−1	$\mathbf{w}^\mathsf{T}x < \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)

No changes until there are mistakes

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize θ = 1024, **w** = (1,1,1,1…,1)

Example	Prediction	Error?	Weights
x=(1,1,1,…,1), y=+1	$\mathbf{w}^{\mathsf{T}}x \geq \theta$	No	**w** = (1,1,1,1…,1)
x=(0,0,0,…,0), y=−1	$\mathbf{w}^{\mathsf{T}}x < \theta$	No	**w** = (1,1,1,1…,1)
x=(0,0,1,1,1,…,0), y=−1	$\mathbf{w}^{\mathsf{T}}x < \theta$	No	**w** = (1,1,1,1…,1)
x=(1,0,0,…,0), y=+1	$\mathbf{w}^{\mathsf{T}}x < \theta$	Yes	**w** = (**2**,1,1,1…,1)

Promote x_1

Example run of the algorithm

$f = x_1 \lor x_2 \lor x_{1023} \lor x_{1024}$ Initialize θ = 1024, $\mathbf{w}$ = (1,1,1,1…,1)

Example	**Prediction**	**Error?**	**Weights**
x=(1,1,1,…,1), y=+1	$\mathbf{w}^\mathsf{T} x \geq \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(0,0,0,…,0), y=−1	$\mathbf{w}^\mathsf{T} x < \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(0,0,1,1,1,…,0), y=−1	$\mathbf{w}^\mathsf{T} x < \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(1,0,0,…,0), y=+1	$\mathbf{w}^\mathsf{T} x < \theta$	Yes	$\mathbf{w}$ = (**2**,1,1,1…,1)
x=(0,1,0,…,0), y=+1	$\mathbf{w}^\mathsf{T} x < \theta$	Yes	$\mathbf{w}$ = (2,**2**,1,1…,1)

Promote x_2

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize θ = 1024, $\mathbf{w}$ = (1,1,1,1…,1)

Example	Prediction	Error?	Weights
x=(1,1,1,…,1), y=+1	$\mathbf{w}^T x \geq \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(0,0,0,…,0), y=−1	$\mathbf{w}^T x < \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(0,0,1,1,1,…,0), y=−1	$\mathbf{w}^T x < \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(1,0,0,…,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	$\mathbf{w}$ = (**2**,1,1,1…,1)
x=(0,1,0,…,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	$\mathbf{w}$ = (2,**2**,1,1…,1)
x=(1,1,1,…,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	$\mathbf{w}$ = (**4**,**4**,**2**,1…,1)

Promote x_1, x_2 and x_3

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize θ = 1024, **w** = (1,1,1,1…,1)

Example	Prediction	Error?	Weights
x=(1,1,1,…,1), y=+1	$\mathbf{w}^T x \geq \theta$	No	**w** = (1,1,1,1…,1)
x=(0,0,0,…,0), y=−1	$\mathbf{w}^T x < \theta$	No	**w** = (1,1,1,1…,1)
x=(0,0,1,1,1,…,0), y=−1	$\mathbf{w}^T x < \theta$	No	**w** = (1,1,1,1…,1)
x=(1,0,0,…,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	**w** = (**2**,1,1,1…,1)
x=(0,1,0,…,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	**w** = (2,**2**,1,1…,1)
x=(1,1,1,…,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	**w** = (**4**,**4**,**2**,1…,1)
x=(1,0,0,…,1), y=+1	$\mathbf{w}^T x < \theta$	Yes	**w** = (**8**,4,2,1…,**2**)
…	…	…	…

Suppose after many steps, **w** = (**512**,**256**,512,512…,**512**)

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize $\theta = 1024$, $\mathbf{w} = (1,1,1,1...,1)$

Example	Prediction	Error?	Weights
x=(1,1,1,...,1), y=+1	$\mathbf{w}^T x \geq \theta$	No	$\mathbf{w}$ = (1,1,1,1...,1)
x=(0,0,0,...,0), y=−1	$\mathbf{w}^T x < \theta$	No	$\mathbf{w}$ = (1,1,1,1...,1)
x=(0,0,1,1,1,...,0), y=−1	$\mathbf{w}^T x < \theta$	No	$\mathbf{w}$ = (1,1,1,1...,1)
x=(1,0,0,...,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	$\mathbf{w}$ = (**2**,1,1,1...,1)
x=(0,1,0,...,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	$\mathbf{w}$ = (2,**2**,1,1...,1)
x=(1,1,1,...,0), y=+1	$\mathbf{w}^T x < \theta$	Yes	$\mathbf{w}$ = (**4**,**4**,**2**,1...,1)
x=(1,0,0,...,1), y=+1	$\mathbf{w}^T x < \theta$	Yes	$\mathbf{w}$ = (**8**,4,2,1...,**2**)
...	...	...	...
			$\mathbf{w}$ = (**512**,**256**,512,512...,**512**)
x=(0,0,1,1,...,0), y=−1	$\mathbf{w}^T x \geq \theta$	Yes	$\mathbf{w}$ = (512,256,**256**,**256**...,512)

Demote x_3 and x_4

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize θ = 1024, **w** = (1,1,1,1…,1)

Example	Prediction	Error?	Weights
x=(1,1,1,…,1), y=+1	$\mathbf{w}^{\mathsf{T}}x \geq \theta$	No	**w** = (1,1,1,1…,1)
x=(0,0,0,…,0), y=−1	$\mathbf{w}^{\mathsf{T}}x < \theta$	No	**w** = (1,1,1,1…,1)
x=(0,0,1,1,1,…,0), y=−1	$\mathbf{w}^{\mathsf{T}}x < \theta$	No	**w** = (1,1,1,1…,1)
x=(1,0,0,…,0), y=+1	$\mathbf{w}^{\mathsf{T}}x < \theta$	Yes	**w** = (**2**,1,1,1…,1)
x=(0,1,0,…,0), y=+1	$\mathbf{w}^{\mathsf{T}}x < \theta$	Yes	**w** = (2,**2**,1,1…,1)
x=(1,1,1,…,0), y=+1	$\mathbf{w}^{\mathsf{T}}x < \theta$	Yes	**w** = (**4**,**4**,**2**,1…,1)
x=(1,0,0,…,1), y=+1	$\mathbf{w}^{\mathsf{T}}x < \theta$	Yes	**w** = (**8**,4,2,1…,**2**)
…	…	…	…
			w = (**512**,**256**,512,512…,**512**)
x=(0,0,1,1,…,0), y=−1	$\mathbf{w}^{\mathsf{T}}x \geq \theta$	Yes	**w** = (512,256,**256**,**256**…,512)
x=(0,0,0,…,1), y=+1	$\mathbf{w}^{\mathsf{T}}x < \theta$	Yes	**w** = (512,256,256,256…,**1024**)

Example run of the algorithm

$f = x_1 \vee x_2 \vee x_{1023} \vee x_{1024}$ Initialize $\theta = 1024$, $\mathbf{w} = (1,1,1,1\ldots,1)$

Example	Prediction	Error?	Weights
x=(1,1,1,…,1), y=+1	$\mathbf{w}^\mathsf{T}x \geq \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(0,0,0,…,0), y=−1	$\mathbf{w}^\mathsf{T}x < \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(0,0,1,1,1,…,0), y=−1	$\mathbf{w}^\mathsf{T}x < \theta$	No	$\mathbf{w}$ = (1,1,1,1…,1)
x=(1,0,0,…,0), y=+1	$\mathbf{w}^\mathsf{T}x < \theta$	Yes	$\mathbf{w}$ = (**2**,1,1,1…,1)
x=(0,1,0,…,0), y=+1	$\mathbf{w}^\mathsf{T}x < \theta$	Yes	$\mathbf{w}$ = (2,**2**,1,1…,1)
x=(1,1,1,…,0), y=+1	$\mathbf{w}^\mathsf{T}x < \theta$	Yes	$\mathbf{w}$ = (**4**,**4**,**2**,1…,1)
x=(1,0,0,…,1), y=+1	$\mathbf{w}^\mathsf{T}x < \theta$	Yes	$\mathbf{w}$ = (**8**,4,2,1…,**2**)
…	…	…	…
			$\mathbf{w}$ = (**512**,**256**,512,512…,**512**)
x=(0,0,1,1,…,0), y=−1	$\mathbf{w}^\mathsf{T}x \geq \theta$	Yes	$\mathbf{w}$ = (512,256,**256**,**256**…,512)
x=(0,0,0,…,1), y=+1	$\mathbf{w}^\mathsf{T}x < \theta$	Yes	$\mathbf{w}$ = (512,256,256,256…,**1024**)

Eventually, the algorithm will converge to something like
$\mathbf{w}$ = (**1024**,**1024**,16,2…, **1024**,**1024**)

Detour: The multiplicative update

Widely used (and re-re-discovered) in various fields

- Winnow (and the Majority Weighted algorithm)
- We will see the AdaBoost algorithm
- Shows up in economics and game theory (from the 1950s)
- Computational Geometry
- Operations research
- Many more…

See: Sanjeev Arora, Elad Hazan and Satyen Kale, *The Multiplicative Weights Update Method: a Meta Algorithm and Applications,* for a survey

This lecture

- The Winnow Algorithm
- Winnow mistake bound
- Generalizations

Winnow mistake bound

We will analyze the simple case of k-disjunctions

Theorem

The Winnow algorithm learns the class of k-disjunctions with n Boolean variables in the Mistake bound model, making O(k log n) mistakes.

Implications:

1. Recall: The Perceptron mistake bound is O(n), "throwing lots of features at the problem" can hurt learning

2. Winnow is *attribute efficient* because it only has a log dependency on n. Only a small penalty for trying out lots of features

The Winnow algorithm

Littlestone 1988

Given a training set D = $\{(x, y)\}$, $x \in \Re^n$, $y \in \{-1,1\}$

1. Initialize: $\mathbf{w} = (1,1,1,1\ldots,1) \in \Re^n$, $\theta = n$

2. For each training example (x, y):
   - Predict $y' = \mathrm{sgn}(\mathbf{w}^T x - \theta)$
   - If $y = +1$ and $y' = -1$ then:
     - Update each weight $\mathbf{w}_i \leftarrow 2\mathbf{w}_i$ *only* for those features x_i that are 1

     Else if $y = -1$ and $y' = +1$ then:
     - Update each weight $\mathbf{w}_i \leftarrow \mathbf{w}_i/2$ *only* for those features x_i that are 1

Proof

Theorem: Winnow will make at most O(k log n) mistakes with k-disjunctions

Our target functions are k-disjunctions

Strategy

Total mistakes = mistakes on positive examples (m^+)

$+$

mistakes on negative examples (m^-)

Get mistake bound upper bounding each separately

1. Mistakes on positives

Theorem: Winnow will make at most O(k log n) mistakes with k-disjunctions

Our target functions are k-disjunctions

- A mistake on a positive example will double the weights for *at least* one of the relevant attributes. Why?

  Because a positive example will have at least one relevant attribute

- We initialized our weight vector with 1s and the threshold θ is always fixed to n

- How many times can a relevant attribute get promoted (i.e. doubled)?
  - 1 + log(n) times. After that, it will cross θ

m^+ = Number of mistakes on positive examples = k (1 + log(n))

2. Mistakes on negatives

Theorem: Winnow will make at most O(k log n) mistakes with k-disjunctions

Our target functions are k-disjunctions

Let TW_t = the sum of all weights at some step $t = \sum w_i$

a. Initially $TW_t = n$, and for all t, $TW_t > 0$

b. What happens to TW_t when there is a mistake on a positive example?

   $TW_{t+1} < TW_t + n$ (Why?)

   $\Rightarrow$ Total increase because of mistakes on positives $< m^+ n$

c. What happens to TW_t when there is a mistake on a negative example?

   $TW_{t+1} < TW_t - n/2$ (Why?)

   $\Rightarrow$ Total decrease because of mistakes on negatives $< m^- n/2$

Putting these together: $0 < TW_t < n + m^+ n - m^- n/2$

$\Rightarrow$ m^- = Number of mistakes on negative examples $< 2(1 + m^+)$

3. Mistake bound

Theorem: Winnow will make at most O(k log n) mistakes with k-disjunctions

Our target functions are k-disjunctions

- What we know:
  1. Mistakes on positive examples = $m^+ < k\,(1 + \log(n))$
  2. Mistakes on negative examples = $m^- < 2(1 + m^+)$

Total number of mistakes = $m^+ + m^-$

$$< m^+ + 2(1 + m^+)$$

$$= 2 + 3\,k\,(\,1 + \log(n))$$

Number of mistakes Winnow will make on k-disjunctions = O(k log n)

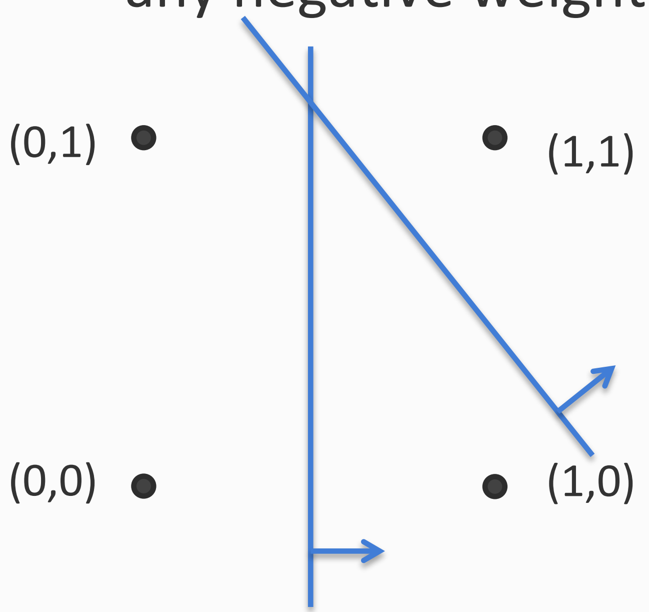
This lecture

- The Winnow Algorithm
- Winnow mistake bound
- Generalizations

What can Winnow represent?

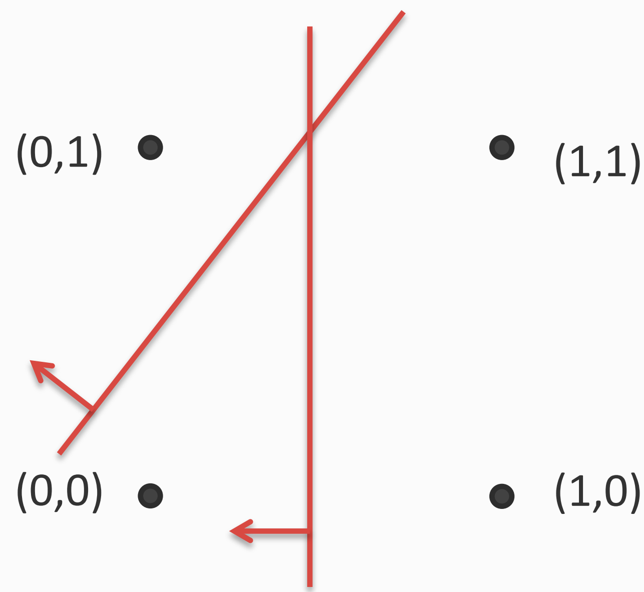
The version we saw can only learn monotone functions

- Why?
- Only multiplying and dividing the weights will never get us any negative weights



Will learn $x_1 \vee x_2$

x_2



Will not learn $\neg x_2$

$x_1 \vee \neg x_2$

Balanced Winnow

- Duplicate the variables
  - If x^+_i represents a Boolean variable, then, introduce a new variable x^-_i to denote its negation
  - That is, learn a monotone function over the 2n variables (w^+_i for each x^+_i and w^-_i for each x^-_i)
  - Effective weight vector is the difference of the two. That is, prediction is performed as:
    - Prediction = +1 if $(\mathbf{w}^+ - \mathbf{w}^-)^T\mathbf{x} \geq \theta$, else prediction = -1
  - Modify the update rule so that whenever w_i is promoted, w^-_i should be demoted and vice versa.
- Can learn any linear threshold unit
- Downsides of this approach?

Balanced Winnow

Given a training set D = $\{(x, y)\}$, $x \in \Re^n$, $y \in \{-1,1\}$

1. Initialize: $\mathbf{w}^+ = (1,1,1,1\ldots,1)$, $\mathbf{w}^- = (1,1,1,1\ldots,1) \in \Re^n$, $\theta = n$

2. For each training example (x, y):
   - Predict $y' = \text{sgn}((\mathbf{w}^+ - \mathbf{w}^-)^T x - \theta)$
   - If $y = +1$ and $y' = -1$ then:
     - Update weight $\mathbf{w}^+_i \leftarrow 2\mathbf{w}^+_i$ *only* for those features x_i that are 1
     - Update weight $\mathbf{w}^-_i \leftarrow \mathbf{w}^-_i/2$ *only* for those features x_i that are 1

     Else if $y = -1$ and $y' = +1$ then:
     - Update weight $\mathbf{w}^+_i \leftarrow \mathbf{w}^+_i/2$ *only* for those features x_i that are 1
     - Update weight $\mathbf{w}^-_i \leftarrow 2\mathbf{w}^-_i$ *only* for those features x_i that are 1

Perceptron and Winnow

- Both are:
  - Mistake bound algorithms
  - Learn linear threshold units
  - Are generally robust

- Which algorithm should you use??
  - Multiplicative algorithms: If you believe that the hidden target function is sparse
  - Additive algorithms: If you believe that your target function could be a dense vector
    - What if the target function is a dense vector but each example is sparse? (We will see additive algorithms that are designed for this regime)

Summary: What Winnow so far?

- A multiplicative update algorithm
  - Learns a linear classifier when very few attributes are relevant
  - Mistake bound only weakly (logarithmically) depends on the number of attributes

- Robust to both classification and attribute noise
  - In general, instead of multiplying and dividing by 2, we could do so by $(1 + \epsilon)$ for some small ϵ