CS446 Introduction to Machine Learning (Fall 2013)
University of Illinois at Urbana-Champaign
http://courses.engr.illinois.edu/cs446

LECTURE 6: ONLINE LEARNING I

Prof. Julia Hockenmaier
juliahmr@illinois.edu

Tuesday's key concepts

Overfitting

kNN classifier

Inductive bias

Statistical bias and variance

Today's key concepts

More on linear classifiers

Two new, mistake-driven update rules for learning linear classifiers:

– Perceptron (additive updates)

– Winnow (multiplicative updates)

More on linear separability

Recap: Vector length

Vector length $\|\mathbf{x}\|$ (l_2 norm)

$$\|\mathbf{x}\| = \sqrt{\sum_i x_i^2}$$

Unit vector (vector of length 1):

$$\mathbf{w}' = \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

With a separate bias term w_0: $f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + w_0$

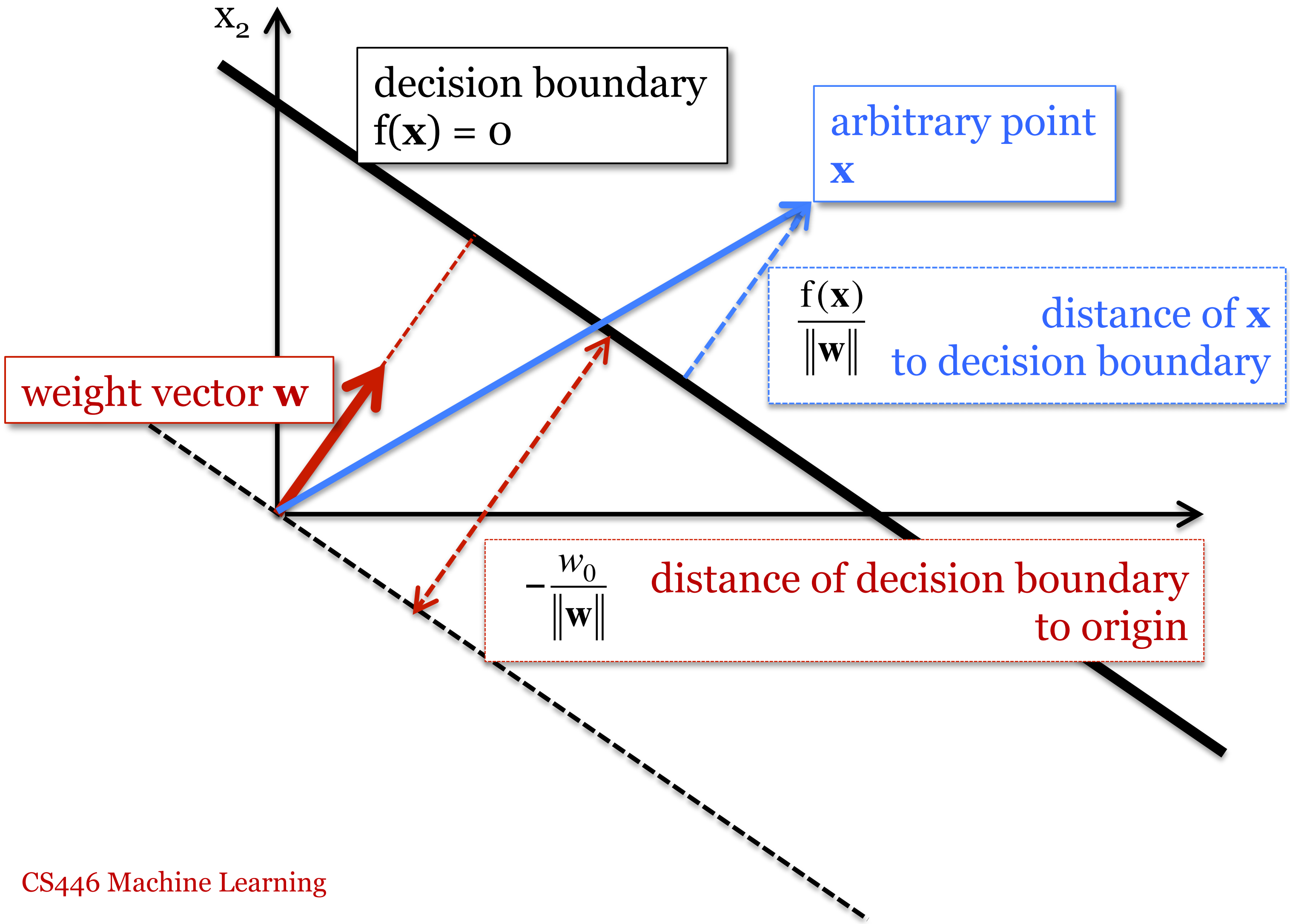
- The **instance space** $\mathcal{X}$ is a d-dimensional vector space (each $\mathbf{x} \in \mathcal{X}$ has d elements)
- The **decision boundary** $f(\mathbf{x}) = 0$ is a $(d-1)$-dimensional hyperplane in the instance space.
- The **weight vector w** is orthogonal (normal) to the decision boundary $f(\mathbf{x}) = 0$:

  For any two points $\mathbf{x}^A$ and $\mathbf{x}^B$ on the decision boundary $f(\mathbf{x}^A) = f(\mathbf{x}^B) = 0$

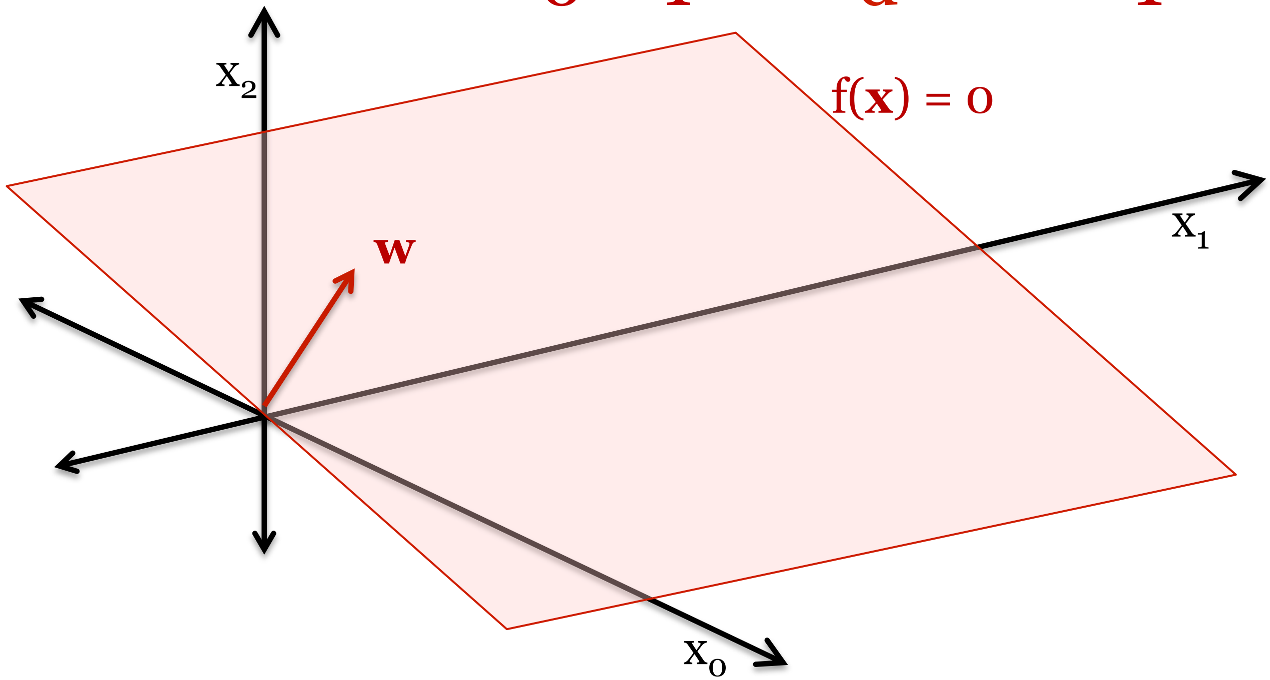
  For any vector $(\mathbf{x}^B - \mathbf{x}^A)$ on the decision boundary: $\mathbf{w}(\mathbf{x}^B - \mathbf{x}^A) = f(\mathbf{x}^B) - w_0 - f(\mathbf{x}^A) + w_0 = 0$
- The **bias term** w_0 determines the distance of the decision boundary from the origin:

  For $\mathbf{x}$ with $f(\mathbf{x}) = 0$, the distance to the origin is $\frac{\mathbf{w}\cdot\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|} = -\frac{w_0}{\sqrt{\sum_{i=1}^{d} w_i^2}}$

With a separate bias term w_0: $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + w_0$



In canonical form (with $x_0 = 1$)
$f(\mathbf{x}) = (w_0 w_1 \ldots w_d)\cdot(1\ x_1 \ldots x_d)$



- We now operate in (d+1)-dimensional space
- The **decision boundary** $f(\mathbf{x}) = 0$ is a d-dimensional hyperplane that goes through the origin.
- The **weight vector w** is still orthogonal to the decision boundary $f(\mathbf{x}) = 0$

Perceptron

Perceptron

Simple, mistake-driven algorithm
for learning linear classifiers

There are batch and online versions

We will analyze the online version

Also uses (stochastic) gradient descent,
but with a different loss function

Perceptron criterion

We would like a weight vector **w** such that

$f(\mathbf{x}_n) = \mathbf{w}\cdot\mathbf{x}_n > 0$ for $y_n = +1$

$f(\mathbf{x}_n) = \mathbf{w}\cdot\mathbf{x}_n < 0$ for $y_n = -1$

The perceptron tries to minimize the error

$$-\mathbf{w}\cdot\mathbf{x}_n\cdot y_n$$

for any misclassified example $(\mathbf{x}_n, y_n)$

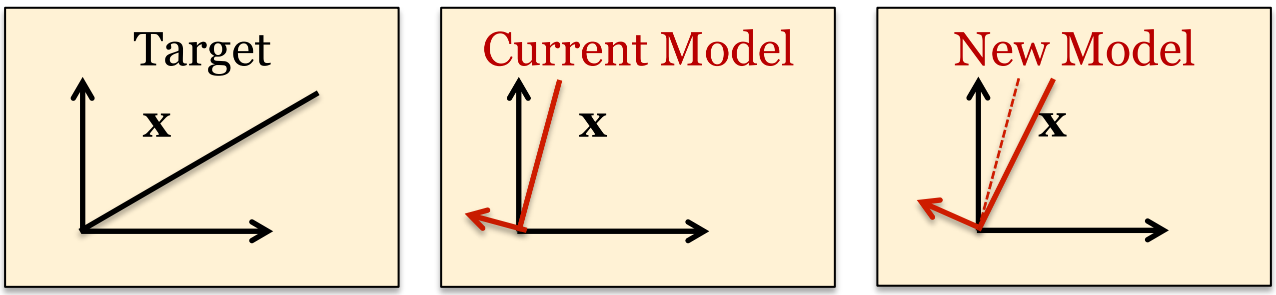
The overall training error of **w** depends on the misclassified items M:

$$E_{Perceptron}(\mathbf{w}) = -\sum_{n \in M} \mathbf{w}\cdot\mathbf{x}_n \cdot y_n$$

The Perceptron rule

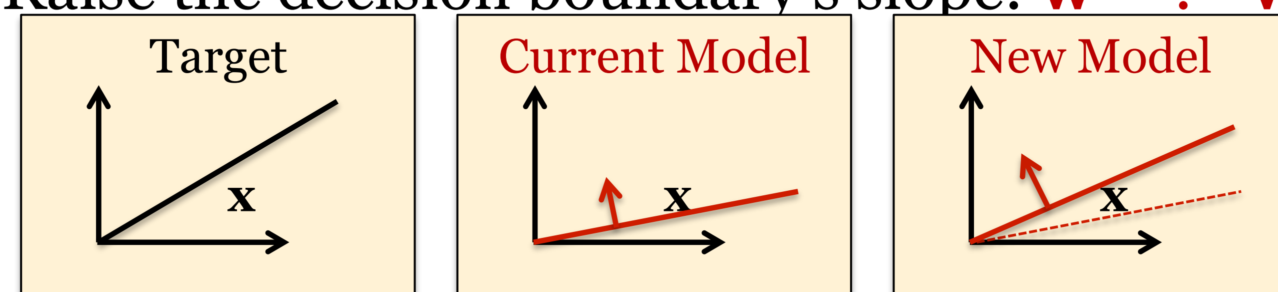
If target y = +1: **x** should be above the decision boundary

Lower the decision boundary's slope: $\mathbf{w}^{i+1} := \mathbf{w}^{i} + \mathbf{x}$



If target y = -1: **x** should be below the decision boundary

Raise the decision boundary's slope: $\mathbf{w}^{i+1} := \mathbf{w}^{i} - \mathbf{x}$



Online perceptron

Assumptions: class labels $y \in \{+1, -1\}$;
learning rate $\alpha > 0$

Initial weight vector $\mathbf{w}^0 := (0,\ldots,0)$

$i = 0$

for $m = 0 \ldots M$:

if $y_m \cdot f(\mathbf{x_m}) = y_m \cdot \mathbf{w}^i \cdot \mathbf{x_m} < 0$:

Perceptron rule

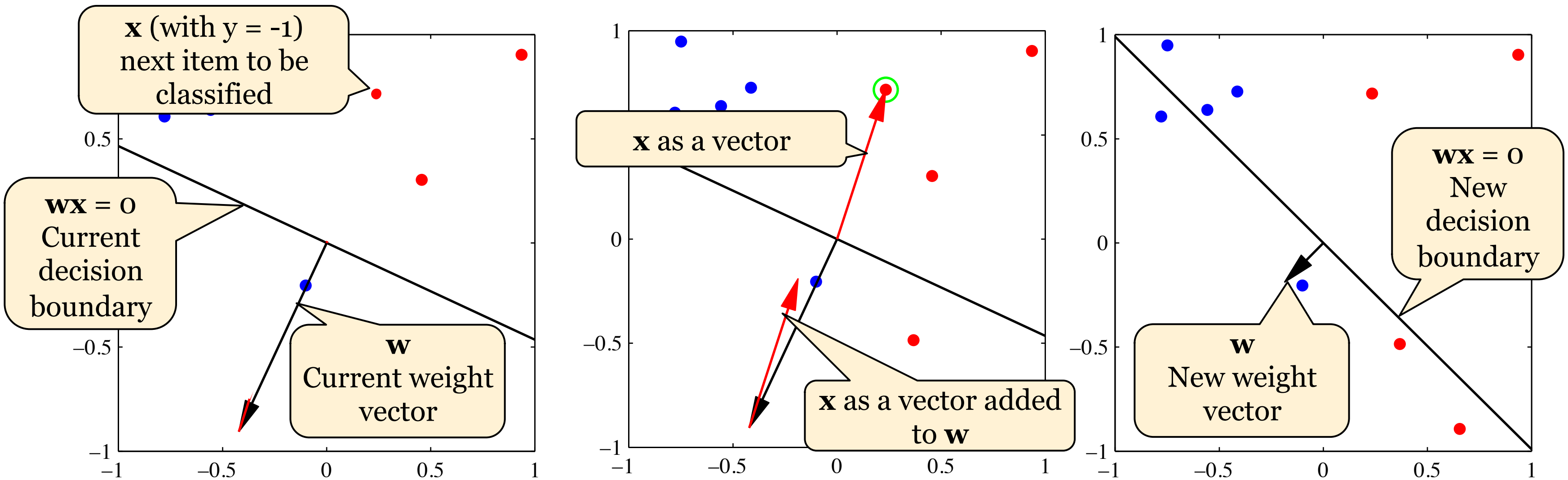
($\mathbf{x_m}$ is misclassified – add $\alpha \cdot y_m \cdot \mathbf{x_m}$ to $\mathbf{w}$!)

$\mathbf{w}^{i+1} := \mathbf{w}^i + \alpha \cdot y_m \cdot \mathbf{x_m}$

$i := i+1$

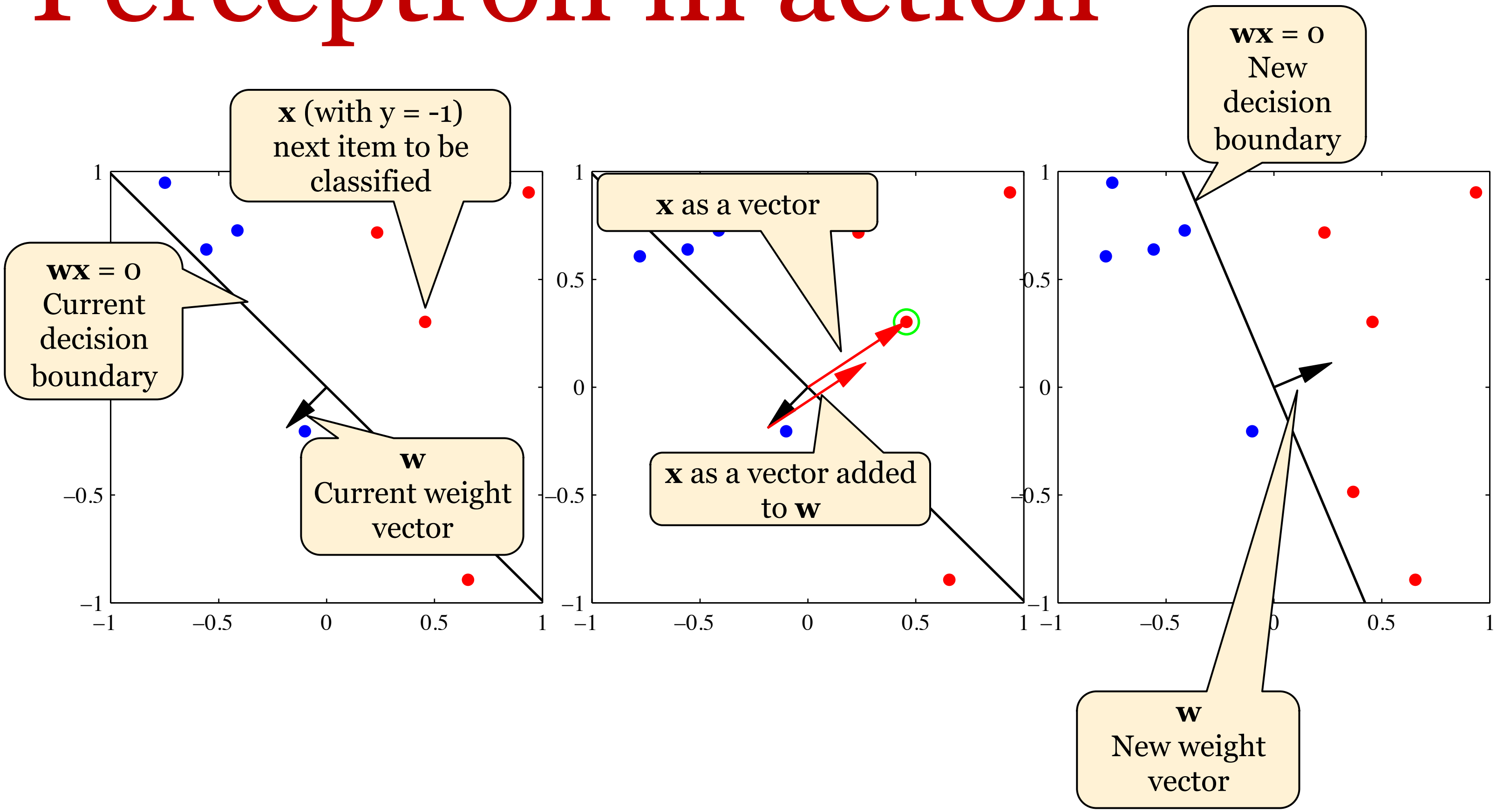
return $\mathbf{w}^{i+1}$ when all examples correctly classified

Perceptron in action



(Figures from Bishop 2006)

Perceptron in action



(Figures from Bishop 2006)

Perceptron and active features

Active features: The feature x_i is active in $\mathbf{x}$ if the i-th component of $\mathbf{x}$ is non-zero

The perceptron rule only updates the weights of active features:

$$\mathbf{w}^{i+1} = \mathbf{w}^{i} + \alpha \cdot y_m \cdot \mathbf{x}_m$$

$$\begin{pmatrix} w_1 + \alpha \cdot 1 \\ w_2 \\ w_3 - \alpha \cdot 1 \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} + \alpha \cdot 1 \cdot \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

Perceptron as (S)GD

Stochastic gradient descent
with a different loss function

Perceptron loss:

$L(y, f(\mathbf{x})) = -y \cdot f(\mathbf{x}) = -y \cdot \mathbf{w} \cdot \mathbf{x}$ if $\mathbf{x}$ misclassified

i.e. if $y \cdot f(\mathbf{x}) = y \cdot \mathbf{w} \cdot \mathbf{x} < 0$

$= 0$ if $\mathbf{x}$ correctly classified

i.e. if $y \cdot f(\mathbf{x}) = y \cdot \mathbf{w} \cdot \mathbf{x} \geq 0$

Perceptron as (S)GD

Perceptron loss when **x** is misclassified:

$$L(y, f(\mathbf{x})) = -y \cdot f(\mathbf{x}) = -y \cdot \mathbf{w} \cdot \mathbf{x}$$

Partial derivatives of perceptron loss on example (**x**, y): $\frac{\partial L}{\partial w_i} = -y \cdot x_i$

Gradient of the perceptron loss on (**x**, y):

$$\nabla L = -y \cdot \mathbf{x}$$

Effect of a single update

The update reduces the error contribution of the current, misclassified example $(\mathbf{x}_n, y_n)$:

Since $\| f(\mathbf{x}_n) y_n \|^2 > 0$

$$
\begin{aligned}
-\mathbf{w}^{i+1} \cdot f(\mathbf{x}_n) y_n &= -[\mathbf{w}^{i} - \alpha \cdot f(\mathbf{x}_n) y_n] \cdot f(\mathbf{x}_n) y_n \\
&= -\mathbf{w}^{i} \cdot f(\mathbf{x}_n) y_n - \alpha \cdot f(\mathbf{x}_n) y_n \cdot f(\mathbf{x}_n) y_n \\
&< -\mathbf{w}^{i} \cdot f(\mathbf{x}_n) y_n
\end{aligned}
$$

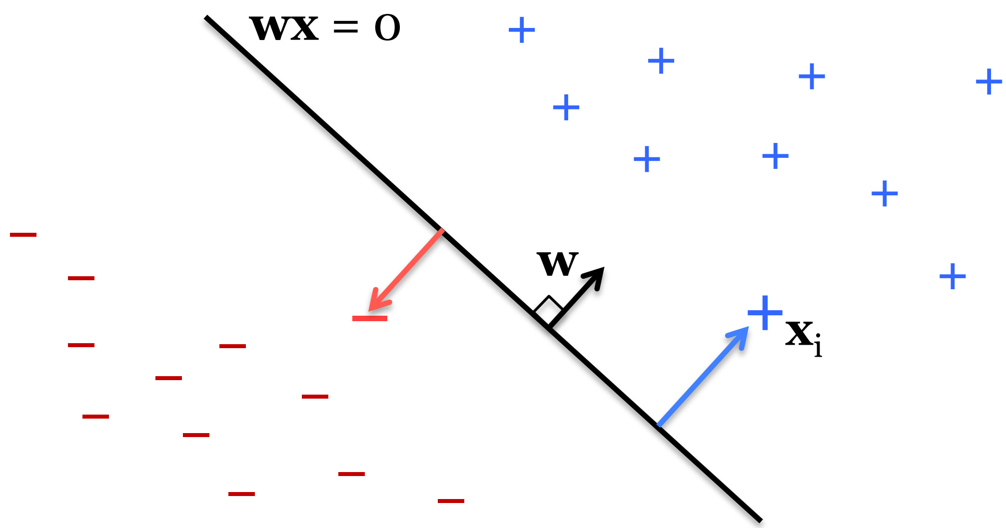
There is no guarantee that this update will reduce the error on any other examples.

Will the perceptron converge?

If the data **is linearly separable**, yes:
It will converge (=classify all items correctly) in a finite number of learning steps.

If the data **is not linearly separable**, no:
The weights will cycle (= get back to the same weight vector over and over)

Margin of training data



The geometric margin γ (= distance) of a point **x** with label y (+1, -1) from the hyperplane defined by the normal vector **w** is $\gamma = y \frac{\mathbf{w}}{\|\mathbf{w}\|}\mathbf{x}$

The (geometric) margin of a set of points is the smallest margin of all points in this set

Perceptron Convergence Theorem
(Block & Novikoff)

Assumptions:

- the data are linearly separable with margin γ (by a unit norm hyperplane $\mathbf{u}$)
- the l_2-norm of the data is bounded by a constant R

We can show that the perceptron algorithm makes at most $k \leq R^2/\gamma^2$ mistakes during training.

Perceptron Convergence Theorem
(Block & Novikoff)

If $(\mathbf{x}_1; y_1),\ldots,(\mathbf{x}_t; y_t)$ is a sequence of labeled examples with $\|\mathbf{x}_i\| \leq R$ that are linearly separable with margin $\gamma > 0$ by a unit norm hyperplane $\mathbf{u}$ ($\mathbf{u} \in \mathbb{R}^N$; $\|\mathbf{u}\| = 1$ and $y_i\, \mathbf{u} \cdot \mathbf{x}_i \geq \gamma$ for all i),

then the perceptron algorithm converges after $k \leq R^2/\gamma^2$ mistakes during training.

That is, the speed of convergence depends on the size of the margin and the L2-norm of the data

Proof of convergence theorem (I)

Assumptions: The data are separable by a unit hyperplane $\mathbf{u}$ with non-zero margin γ: $y_i \cdot \mathbf{u} \cdot \mathbf{x}_i \geq \gamma > 0$ and $\| \mathbf{u} \| = 1$

Terminology: $\mathbf{v}_o = (0 \ldots 0)^T$ is the initial weight vector

$\mathbf{v}_k$ is the weight vector before the k-th mistake

Proof: The k-th mistake happens on $(\mathbf{x}_i, y_i)$, with $y_i \cdot \mathbf{v}_k \cdot \mathbf{x}_i \leq 0$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + y_i \cdot \mathbf{x}_i \qquad \text{(by definition)}$$

$$\mathbf{v}_{k+1} \cdot \mathbf{u} = \mathbf{v}_k \cdot \mathbf{u} + y_i \cdot \mathbf{x}_i \cdot \mathbf{u}$$

$$\geq \mathbf{v}_k \cdot \mathbf{u} + \gamma$$

$$\mathbf{v}_{k+1} \cdot \mathbf{u} \geq \mathbf{v}_{k-1} \cdot \mathbf{u} + 2\gamma \geq \ldots \geq \mathbf{v}_0 \cdot \mathbf{u} + k\gamma$$

Hence: $\mathbf{v}_{k+1} \cdot \mathbf{u} \geq k\gamma$

Proof of convergence theorem (II)

Assumptions: The data are separable by a unit hyperplane $\mathbf{u}$ with non-zero margin γ: $y_i \cdot \mathbf{u} \cdot \mathbf{x}_i \geq \gamma > 0$ and $\| \mathbf{u} \| = 1$

Terminology: $\mathbf{v}_o = (0 \ldots 0)^T$ is the initial weight vector

$\mathbf{v}_k$ is the weight vector before the k-th mistake

Proof: The k-th mistake happens on $(\mathbf{x}_i, y_i)$, with $y_i \cdot \mathbf{v}_k \cdot \mathbf{x}_i \leq 0$

We have just established that $\mathbf{v}_{k+1} \cdot \mathbf{u} \geq k\gamma$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + y_i \cdot \mathbf{x}_i \quad \text{(by definition)}$$

$$\begin{aligned}
\| \mathbf{v}_{k+1} \|^2 &= \| \mathbf{v}_k \|^2 + 2 y_i \cdot \mathbf{v}_k \cdot \mathbf{x}_i + \| \mathbf{x}_i \|^2 \\
&\leq \| \mathbf{v}_k \|^2 + R^2 \qquad (\text{since } y_i \cdot \mathbf{v}_k \cdot \mathbf{x}_i \leq 0 \text{ and } \| \mathbf{x} \| \leq R) \\
&\leq \| \mathbf{v}_{k-1} \|^2 + 2R^2 \leq \ldots \leq \| \mathbf{v}_0 \|^2 + kR^2
\end{aligned}$$

Hence: $\| \mathbf{v}_{k+1} \|^2 \leq kR^2$

Since $\| \mathbf{u} \| = 1$: $\quad k\gamma \leq \mathbf{v}_{k+1} \cdot \mathbf{u} \leq \| \mathbf{v}_{k+1} \| \leq \sqrt{k} \cdot R$

Since $k\gamma \leq \sqrt{k} \cdot R$: $\quad \sqrt{k} \leq R/\gamma \Rightarrow k \leq R^2/\gamma^2$

Winnow

Winnow

Multiplicative, mistake-driven update rule

- Perceptron and LMS: additive update

Touches only weights of the active features in $\mathbf{x}$

- x_i is active in $\mathbf{x}$ if the i-th component in $\mathbf{x}$ is non-zero

Winnow update

if $\mathbf{w}^k$ misclassifies $\mathbf{x}^i$:

if $y^i = +1$:

($\mathbf{x}^i$ should be above the decision boundary, but is currently below it)

double the weights of the features that are active in $\mathbf{x}^i$

if $y^i = -1$:

($\mathbf{x}^i$ should be below the decision boundary, but is currently above it)

halve the weights of the features that are active in $\mathbf{x}^i$

(don't touch weights of inactive features)

Comparing perceptron, Winnow and LMS

Comparison: Winnow

if $\mathbf{w}^k$ misclassifies $\mathbf{x}^i$:

 if $y^i = +1$:

  double the weights of the features that are active in $\mathbf{x}^i$

 if $y^i = -1$:

  halve the weights of the features that are active in $\mathbf{x}^i$

Scales well when many features are irrelevant for the target concept (good when the target weight vector $\mathbf{w}$ is sparse)

Comparison: Perceptron

Perceptron

$\mathbf{w}^{i+1} = \mathbf{w}^{i} + y^{i}\mathbf{x}^{i}$ if $\mathbf{w}^{k}$ misclassifies $\mathbf{x}^{i}$,

$\mathbf{w}^{i+1} = \mathbf{w}^{i}$ otherwise

Converges after a finite number of mistakes if the data are linearly separable (otherwise, it cycles)

Rate of convergence depends on the number of active features in each item (good when the input $\mathbf{x}$ is sparse)

Comparison: LMS

LMS (aka Adaline, Widrow-Hoff):

$$\mathbf{w}^{i+1} = \mathbf{w}^{i} + \alpha(y^{i} - \mathbf{w}^{i}\mathbf{x}^{i}))\cdot\mathbf{x}^{i}$$

Converges asymptotically (in the limit) toward the minimum error hypothesis, even when data are not linearly separable

How many updates are required?

Mistake bounds: How many mistakes will the learner make before it has converged?

- Multiplicative algorithms (e.g. Winnow)

  Bounds depend on $||\mathbf{u}||_1$, the l_1-norm of the separating hyperplane

  Advantage with few relevant features in concept

- Additive algorithms (e.g. Perceptron)

  Bounds depend on $||\mathbf{x}||$ (Kivinen / Warmuth, '95)

  Advantage with few active features per example

Outlook: Dealing with data that is not linearly separable

Case 1: The target concept isn't linearly separable

Solution: use a non-linear classifier
or the kernel trick

Case 2: The data is noisy

(some feature values or target labels are incorrect)

Solution: use a linear classifier with margins

More on the kernel trick and margin-based classifiers later

Today's key concepts

More on linear classifiers

Two new, mistake-driven update rules for learning linear classifiers:

– Perceptron (additive updates)

– Winnow (multiplicative updates)