Perceptron

A Canonical Representation

- Given a training example: (<x_1, x_2, x_3, x_4>, y) $y \in \{-1, 1\}$
- Transform it to canonical representation

  (<1, x_1, x_2, x_3, x_4>, y)

- Learn a linear function g(**x**,**w**) $= \mathbf{w}^T\mathbf{x}$, where **w** = <w_0, w_1, w_2, w_3, w_4>

- Each **w** corresponds to one hypothesis

  $h(\mathbf{x}) = \text{sign}(g(\mathbf{x},\mathbf{w}))$

- A prediction is correct if $y\,\mathbf{w}^T\mathbf{x} > 0$
- Goal of learning is to find a good **w**
  - e.g., a **w** such that $h(\mathbf{x})$ makes few mis-predictions

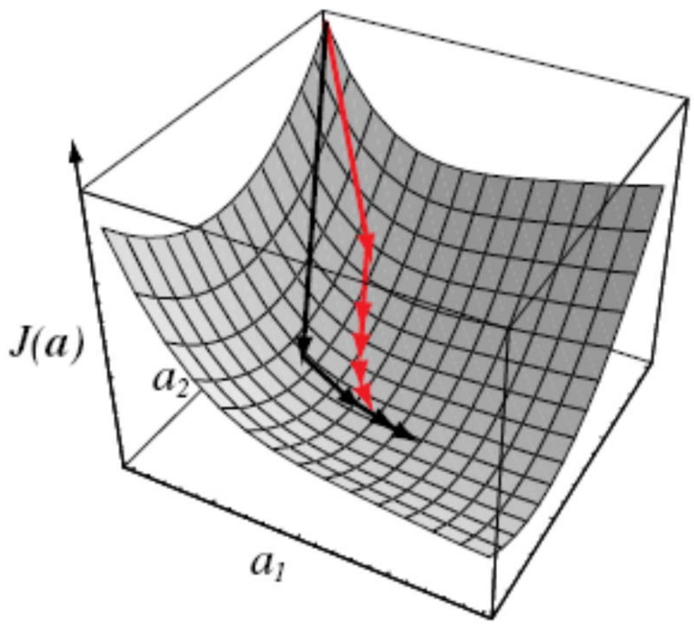
Learning w: An Optimization Problem

- Formulate learning problem as an optimization problems
  - Given:
    - A set of *N* training examples
      $\{(\mathbf{x}_1,y_1), (\mathbf{x}_2,y_2), \ldots, (\mathbf{x}_N,y_N)\}$
    - A loss function *L*
  - Find the weight vector **w** that minimizes the objective function - the expected/average loss on training data

$$J(w) = \frac{1}{N}\sum_{i=1}^{N} L(w \cdot x_i, y_i)$$

- Many machine learning algorithms apply some optimization algorithm to find a good hypothesis.

Gradient Descent Search



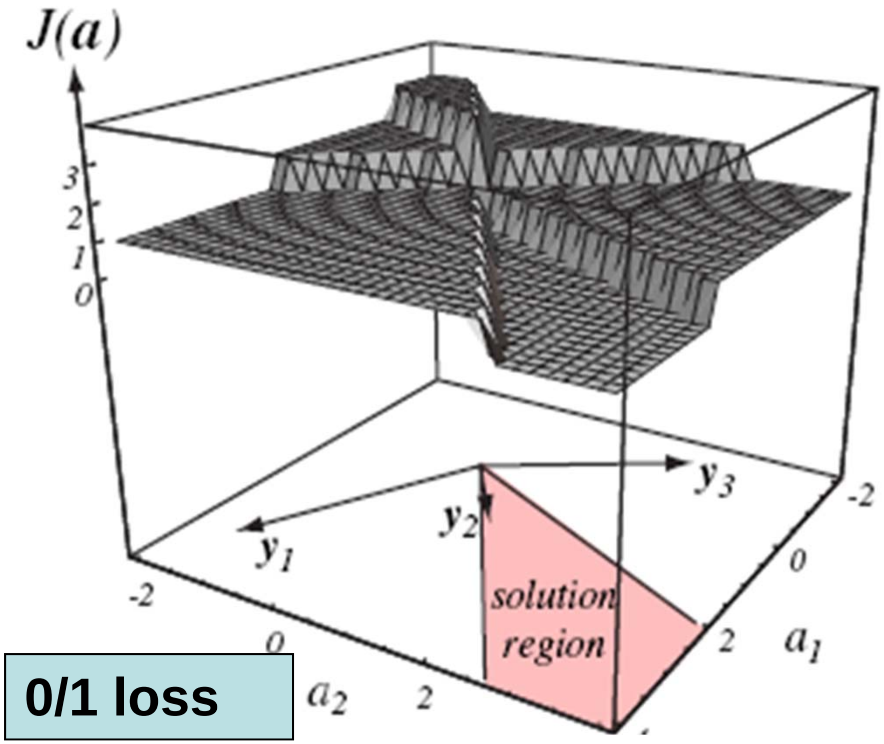
- Start with initial **w** = (w_0,..., w_n)
- Compute gradient $$\nabla J(\mathbf{w}_0) = \left(\frac{\partial J(\mathbf{w})}{\partial w_0}, \frac{\partial J(\mathbf{w})}{\partial w_1}, \cdots, \frac{\partial J(\mathbf{w})}{\partial w_n}\right)_{\mathbf{w}_0}$$
- $w_{t+1} = w_t - \eta \nabla J(w_t)$, Where η is the "step size" parameter
- Repeat until convergence

Remaining question: what objective to use?

Loss Functions

- 0/1 Loss function: $J_{0/1}(w) = \frac{1}{N}\sum_{i=1}^{N} L(\text{sgn}(w \cdot x_i), y_i)$

  L(y',y) = 0 when y'=y, otherwise L(y',y)=1
- Does not produce useful gradient since the surface of J is flat



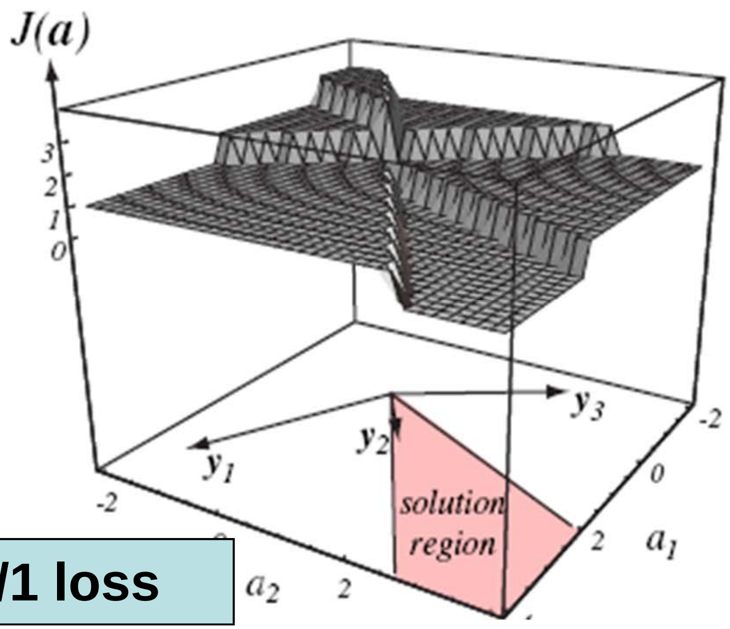
0/1 loss

Loss Functions

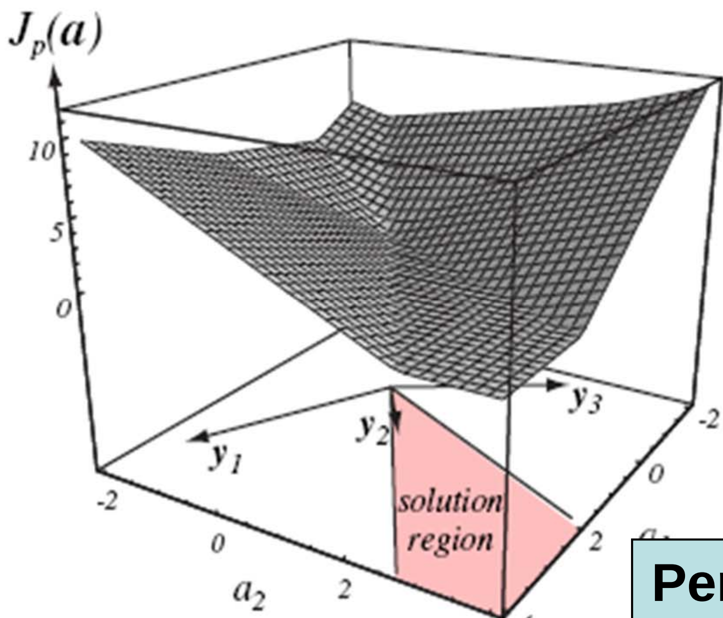
- Instead we will consider the "**perceptron criterion**" (a slightly modified version of hinge loss):

$$J_p(w) = \frac{1}{N}\sum_{i=1}^{N} \max(0, -y_i w \cdot x_i)$$

- The term $\max(0, -y_i w \cdot x_i)$ is 0 when y_i is predicted correctly otherwise it is equal to the "confidence" in the mis-prediction
- Has a nice gradient leading to the solution region



0/1 loss



Perceptron criterion

Stochastic Gradient Descent

- The objective function consists of a sum over data points---we can update the parameter after observing each example
- This is referred to as Stochastic gradient descent approach

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \max(0, -y_i \mathbf{w}\cdot\mathbf{x}_i)$$

$$J_i(\mathbf{w}) = \max(0, -y_i \mathbf{w}\cdot\mathbf{x}_i)$$

$$\frac{\partial J_i}{\partial w_j} = \begin{cases} 0 & \text{if } y_i \mathbf{w}\cdot\mathbf{x}_i > 0 \\ -y_i x_{ij} & \text{otherwise} \end{cases}$$

$$\nabla J_i = \begin{cases} 0 & \text{if } y_i \mathbf{w}\cdot\mathbf{x}_i > 0 \\ -y_i \mathbf{x}_i & \text{otherwise} \end{cases}$$

After observing $(\mathbf{x}_i, y_i)$, if it is a mistake $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

Online Perceptron Algorithm

Let $\mathbf{w} \leftarrow (0,0,0,...,0)$

Repeat

Accept training example $i : (\mathbf{x}_i \ , y_i)$

$u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i$

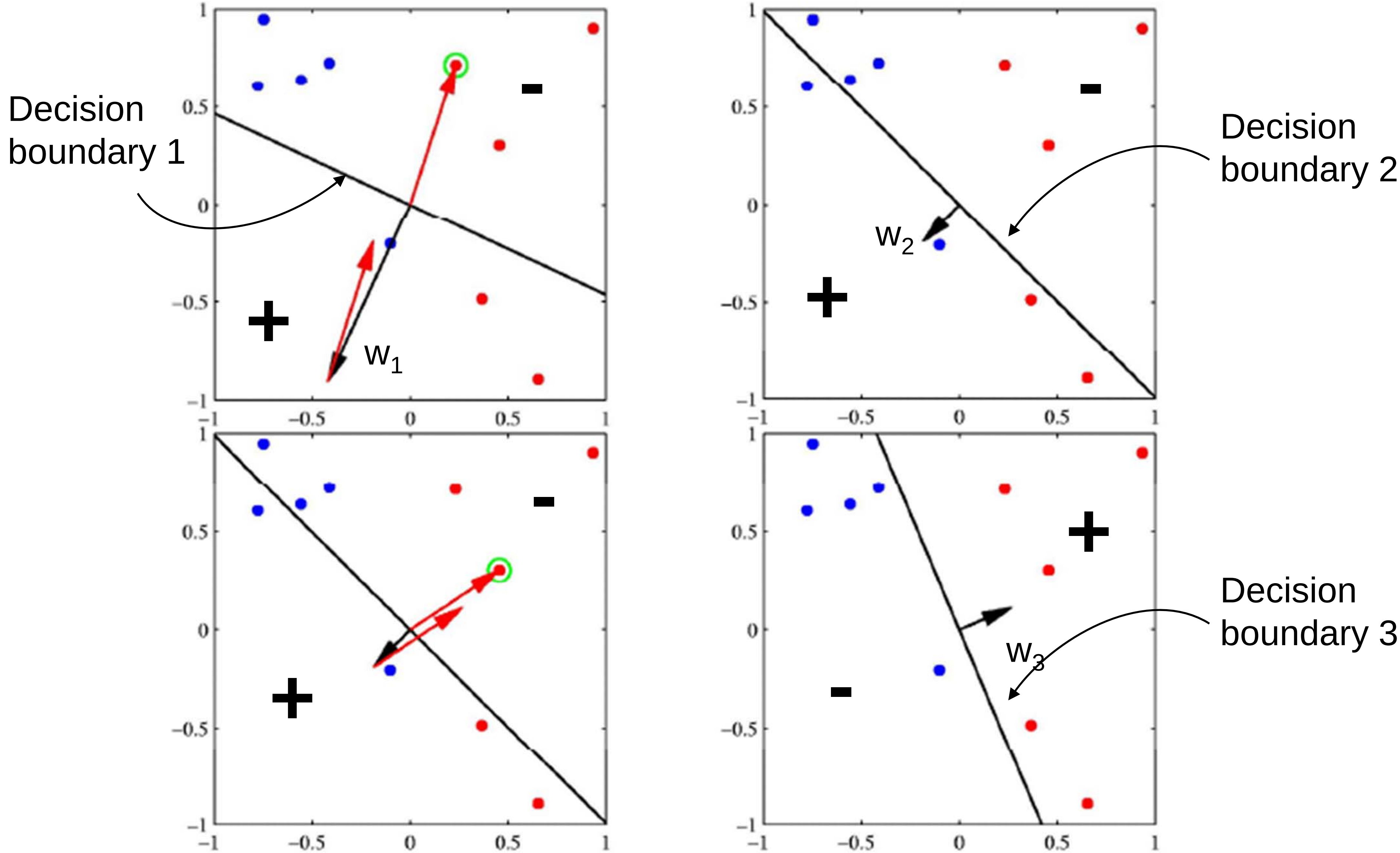
if $y_i \cdot u_i \ \leq \ 0$

$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$

Online learning refers to the learning mode in which the model update is performed each time a single observation is received.

Batch learning in contrast performs model update after observing the whole training set.

When an error is made, moves the weight in a direction that corrects the error



Red points belong to the positive class,
blue points belong to the negative class

Batch Perceptron Algorithm

$$
\begin{aligned}
&\text{Given: training examples } (\mathbf{x}_i\ , y_i),\ i = 1,\ldots, N \\
&\text{Let } \mathbf{w} \leftarrow (0,0,0,\ldots,0) \\
&\text{do} \\
&\qquad delta \leftarrow (0,0,0,\ldots,0) \\
&\qquad \text{for } i\ =\ 1 \text{ to } N \text{ do} \\
&\qquad\qquad u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i \\
&\qquad\qquad \text{if } \ y_i \cdot u_i \ \leq \ 0 \\
&\qquad\qquad\qquad delta \leftarrow delta - y_i \cdot x_i \\
&\qquad delta \leftarrow delta / N \\
&\qquad \mathbf{w} \leftarrow \mathbf{w} - \eta\, delta \\
&\text{until } \ |\, delta\, | < \varepsilon
\end{aligned}
$$

Simplest case: η = *1* and don’t normalize – ‘Fixed increment perceptron’

η – the step size

- Also referred as the learning rate
- In practice, recommend to decrease η as learning continues
- Some optimization approaches set step-size automatically, e.g., by line search, and converge faster
- If linearly separable, there is only one basin for the hinge loss, thus local minimum is the global minimum

Online VS. Batch Perceptron

- Batch learning learns with a batch of examples collectively
- Online learning learns with one example at a time
- Both learning mechanisms are useful in practice
- Online Perceptron is sensitive to the order that training examples are received
- In batch training, the correction incurred by each mistake is accumulated and applied at once at the end of the iteration
- In online training, each correction is applied immediately once a mistake is encountered, which will change the decision boundary, thus different mistakes maybe encountered for online and batch training
- Online training performs stochastic gradient descent, an approximation to real gradient descent, which is used by the batch training

Convergence Theorem

(Block, 1962, Novikoff, 1962)

Given training example sequence $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_N, y_N)$.

If $\forall i,\ \left\|\mathbf{x}_i\right\| \le D$, and $\exists\, \mathbf{u},\ \left\|\mathbf{u}\right\| = 1$ and $y_i \mathbf{u} \cdot \mathbf{x}_i \ge \gamma > 0$ for all i,

then the number of mistakes that the perceptron algorithm makes is at most $(D / \gamma)^2$.

Note that ||·|| is the Euclidean length of a vector.

Proof

To show convergence, we just need to show that each update moves the weight vector closer to a solution vector by a lower bounded amount

Note that $\alpha\mathbf{u}$ is also a solution vector, given that $\mathbf{u}$ is a solution vector, where α is an arbitrary scaling factor

Let $\mathbf{x}_k$ be the kth mistake, we have $\mathbf{w}(k+1) = \mathbf{w}(k) + y_k\mathbf{x}_k$

$$
\begin{aligned}
& \left\|\mathbf{w}(k+1) - \alpha\mathbf{u}\right\|^2 \\
&= \left\|\mathbf{w}(k) + y_k\mathbf{x}_k - \alpha\mathbf{u}\right\|^2 = \left\|\left(\mathbf{w}(k) - \alpha\mathbf{u}\right) + y_k\mathbf{x}_k\right\|^2 \\
&= \left\|\mathbf{w}(k) - \alpha\mathbf{u}\right\|^2 + 2y_k[\mathbf{x}_k \cdot \left(\mathbf{w}(k) - \alpha\mathbf{u}\right)] + (y_k)^2\left\|\mathbf{x}_k\right\|^2 \\
&= \left\|\mathbf{w}(k) - \alpha\mathbf{u}\right\|^2 + 2y_k\mathbf{x}_k{}^T\mathbf{w}(k) - 2y_k\alpha\mathbf{u}^T\mathbf{x}_k + \left\|\mathbf{x}_k\right\|^2 \\
&\le \left\|\mathbf{w}(k) - \alpha\mathbf{u}\right\|^2 + 2y_k\mathbf{x}_k{}^T \cdot \mathbf{w}(k) - 2y_k\alpha\mathbf{u}^T\mathbf{x}_k + D^2 \quad \text{, because } \left\|\mathbf{x}_k\right\| \le D \\
&\le \left\|\mathbf{w}(k) - \alpha\mathbf{u}\right\|^2 - 2\alpha\mathbf{u}^T\mathbf{x}_k + \mathrm{D}^2 \quad \text{, because } y_k\mathbf{x}_k{}^T\mathbf{w}(k) \le 0 \\
&\le \left\|\mathbf{w}(k) - \alpha\mathbf{u}\right\|^2 - 2\alpha\gamma + D^2 \quad \text{, because } y_k\mathbf{u}^T\mathbf{x}_k \ge \gamma
\end{aligned}
$$

Because α is an arbitrary scaling factor, we can set $\alpha = \dfrac{D^2}{\gamma}$

$$\left\|\mathbf{w}(k+1) - \alpha\mathbf{u}\right\|^2 \le \left\|\mathbf{w}(k) - \alpha\mathbf{u}\right\|^2 - D^2$$

Proof (cont.)

By induction on k, we can show that

$$\left\|\mathbf{w}(k+1)-\alpha\mathbf{u}\right\|^2 \le \left\|\mathbf{w}(1)-\alpha\mathbf{u}\right\|^2 - kD^2 = \alpha^2\left\|\mathbf{u}\right\|^2 - kD^2 = \alpha^2 - kD^2$$

$$\Leftrightarrow \quad \alpha^2 - kD^2 \ge 0$$

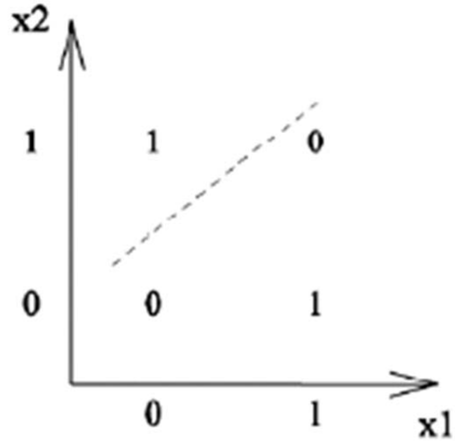
$$\Leftrightarrow \quad k \le \frac{\alpha^2}{D^2} \qquad (\alpha = \frac{D^2}{\gamma})$$

$$\Leftrightarrow \quad k \le (D/\gamma)^2$$

Margin

- γ is referred to as the margin
  - The minimum distance from data points to the decision boundary
  - The bigger the margin, the easier the classification problem is
  - The bigger the margin, the more confident we are about our prediction
- We will see later in the course this concept leads to one of the recent most exciting developments in the ML field – support vector machines

Not linearly separable case



- In such cases the algorithm will never stop! How to fix?
- Look for decision boundary that make as few mistakes as possible – NP-hard!

Fixing the Perceptron

- Idea one: only go through the data once, or a fixed number of times

$$
\begin{array}{l}
\text{Let } \mathbf{w} \leftarrow (0,0,0,\ldots,0) \\
\text{Repeat for N times} \\
\qquad \text{Take a training example } i : (\mathbf{x}_i \ , y_i) \\
\qquad u_i \leftarrow \mathbf{w} \cdot \mathbf{x}_i \\
\qquad \text{if } \ y_i \cdot u_i \ \leq \ 0 \\
\qquad \qquad \mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i
\end{array}
$$

- At least this stops
- Problem: the final ***w*** might not be good e.g. right before it stops the algorithm might perform an update on a total outlier

Voted Perceptron

- Keep intermediate hypotheses and have them vote [Freund and Schapire 1998]

Let $w_0 = (0,0,0, \ldots, 0)$

$c_0 = 0$

repeat

  Take example $i : (x_i, y_i)$

  $u_i \leftarrow \mathbf{w}_n \cdot \mathbf{x}_i$

  if $y_i \cdot u_i <= 0$

    $\mathbf{w}_{n+1} \leftarrow \mathbf{w}_n + y_i \mathbf{x}_i$

    $c_{n+1} = 0$

    $n = n + 1$

  else

    $c_n = c_n + 1$

Store a collection of linear separators $\mathbf{w}_0$ $\mathbf{w}_1$ …, along with their survival time c_0, c_1 …

The c's can be good measures of the reliability of the **w**'s

For classification, take a weighted vote among all separators:

$$\operatorname{sgn}\{\sum_{n=0}^{N} c_n \operatorname{sgn}(\mathbf{w}_n^T \mathbf{x})\}$$

Summary of Perceptron

- Learns a Classifier $\hat{y} = f(\mathbf{x})$ directly
- Applies gradient descent search to optimize the hinge loss function
  - Online version performs stochastic gradient descent
- Guaranteed to converge in finite steps if linearly separable
  - There exists an upper bound on the number of corrections needed
  - Inversely proportional to the margin of the optimal decision boundary
- If not linearly separable, use voted perceptrons

Geometric Interpretation of Linear Discriminant Functions

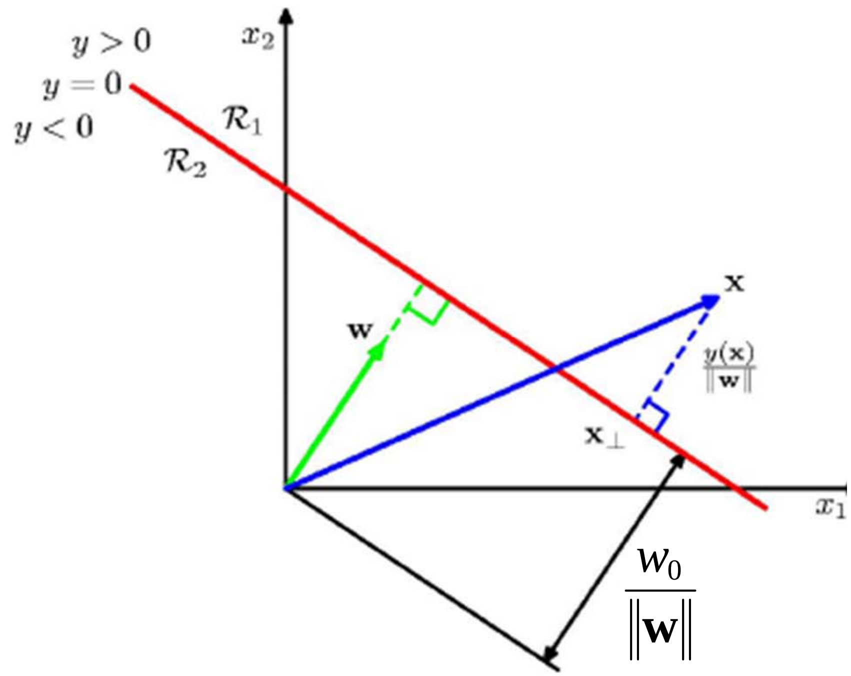
- Two classes

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

if $y(\boldsymbol{x}) \geq 0$, assign to C_1

otherwise, assign to C_2

- Decision boundary: $y(\boldsymbol{x})=0$
- Decision boundary is perpendicular to $\boldsymbol{w}$



The signed distance (positive if **x** is on the positive side, negative otherwise) from any point **x** to the decision boundary is: $\dfrac{y(\mathbf{x})}{\|\mathbf{w}\|}$

Note that in Perceptron, due to the adoption of the canonical representation, all training points will lie on the hyperplane x_0=0, and the decision boundary will always go through the origin.