Machine Learning

CS 165B

Prof. Matthew Turk

Wednesday, May 4, 2016

Today

- Linear learning models (cont.)

Notes

- Homework averages:
  - HW1 – 83% (56.7/68)
  - HW2 – 85% (83.3/98)
- Grading
  - Syllabus: 50% HW, 20% midterm, 30% final exam
  - New option:
    - 50% HW, **15%** midterm, **35%** final exam
    - Will use whichever is better for you…
  - Final grade classifier: A > 90%, B > 80%, C > 70%, D > 60%, else F
    - With some (small) possibility of minor curving

Notes

- Covered:
  - Understanding how to formulate core ML problems
  - Key ML concepts and terms
  - Basic classification and regression methods
    - Binary, multiclass
    - Scoring and ranking classifiers
    - Linear methods
- Coming:
  - Perceptron and SVM methods
  - Kernel methods (non-linear)
  - Clustering techniques
  - Neural networks
  - ML experiments

Quiz questions

- For data with N features, what is the dimensionality of the linear regression function?
  - N (fit a line to 1D data, fit a plane to 2D data, etc.)

- For data with N features, what is the dimensionality of the linear classification boundary?
  - N-1 (a line separates 2D data, a plane separates 3D data, etc.)

- For data with N features, what is (nonhomogeneous) ***w***?
  - An N-dimensional vector

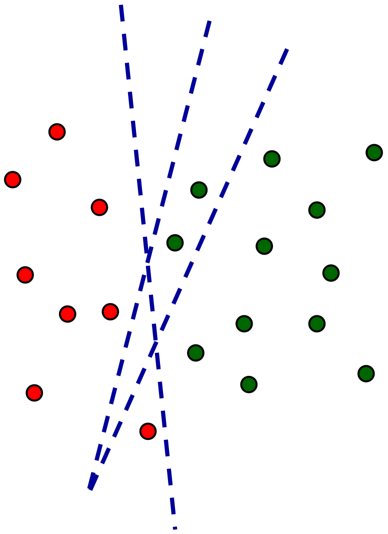
- What's the output/result of linear classifier training?
  - (***w***, *t*)

The perceptron

- The perceptron model is an iterative linear classifier that will achieve perfect separation on linearly separable data
- A perceptron iterates over the training data, updating $\boldsymbol{w}$ every time it encounters an incorrectly classified example
  - How to move the boundary for a misclassified example?
  - How much to move it?
- Update rule (homogeneous training data $\boldsymbol{x}_i \in \mathbb{R}^{k+1}$):

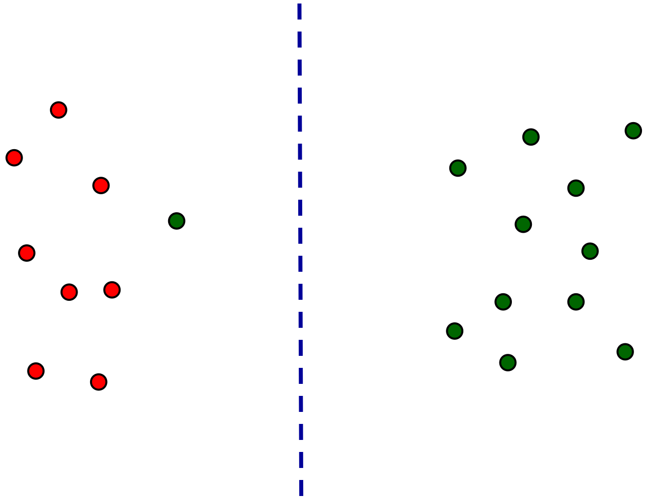
  $\boldsymbol{w}' = \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$ where η is the learning rate, $0 < \eta \leq 1$
- Iterate through the training examples (each pass over the data is called an epoch) until all examples in an epoch are correctly classified
- Guaranteed to converge if the training data is linearly separable – but won’t converge otherwise

The perceptron



$$\boldsymbol{w}' = \boldsymbol{w} + \eta y_i \boldsymbol{x_i}$$

Iterate through the training examples (each pass over the data is called an epoch) until all examples are correctly classified



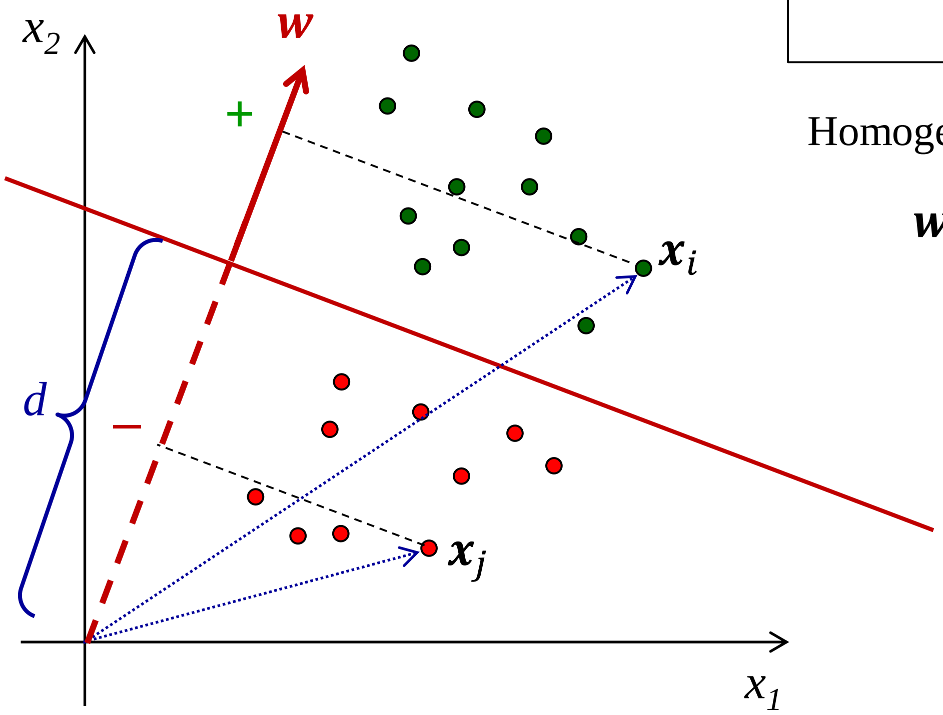
By the way…

- The book is sometimes unclear when they're using homogeneous notation and when they're not
- For example, $\boldsymbol{w}^T\boldsymbol{x}$ can mean either

$$\begin{tikzcd}
\text{Nonhomogeneous} & & \text{Homogeneous} \\
\begin{bmatrix} w_1 & w_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \arrow[d] & \text{or} & \begin{bmatrix} w_1 & w_2 & -t \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \arrow[d] \\
\boldsymbol{w}^T\boldsymbol{x} - t > 0 \arrow[dr] & & \boldsymbol{w}^T\boldsymbol{x} > 0 \arrow[dl] \\
& w_1x_1 + w_2x_2 - \mathrm{t} > 0 &
\end{tikzcd}$$

Interpret in context…

Classifier geometry – *w* and *t*



Non-homogeneous:

$$\boldsymbol{w}^T\boldsymbol{x} - t = \begin{bmatrix} w_1 & w_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - t > 0$$

Homogeneous:

$$\boldsymbol{w}^T\boldsymbol{x} = \begin{bmatrix} w_1 & w_2 & -t \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} > 0$$

Is $\boldsymbol{w}$ a unit vector?

Doesn't have to be

What's the relationship between $\boldsymbol{w}$ and t ?

$(w, t) \equiv (kw, kt)$

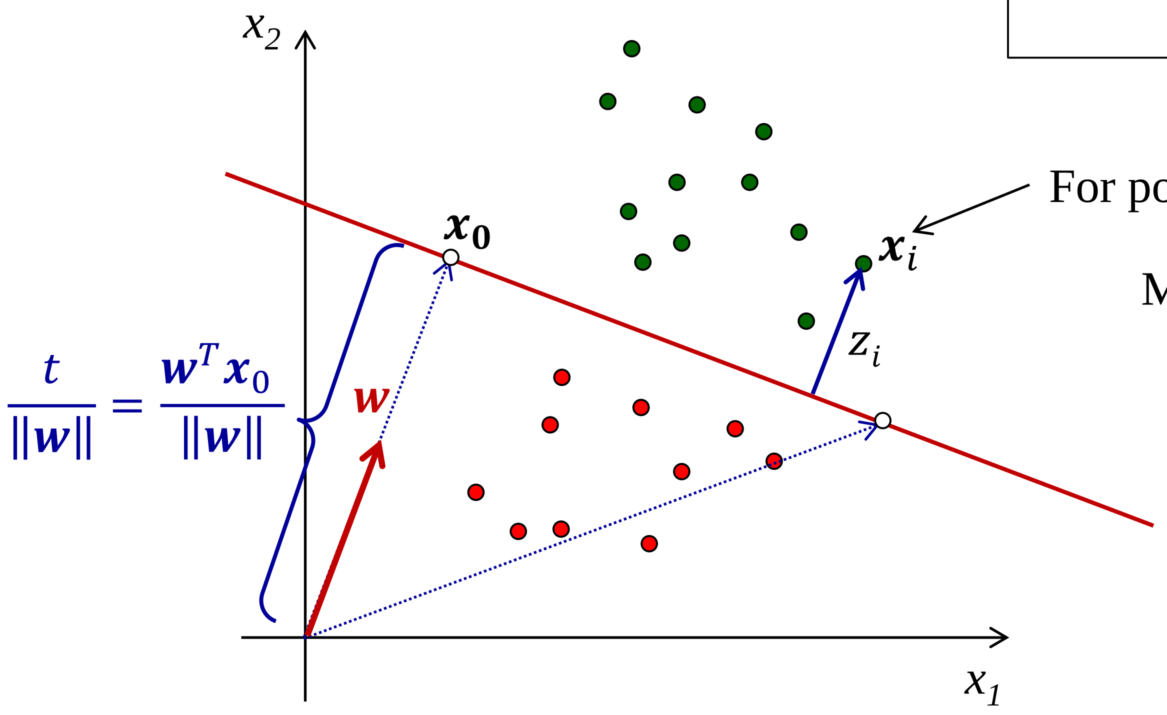
$$\begin{bmatrix} w_1 & w_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - t = 0 \qquad \begin{bmatrix} 2w_1 & 2w_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 2t = 0$$

These describe the same line

Classifier geometry – *w* and *t*

Non-homogeneous:

$$\boldsymbol{w}^T\boldsymbol{x} - t = \begin{bmatrix} w_1 & w_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - t > 0$$



$$\frac{t}{\|\boldsymbol{w}\|} = \frac{\boldsymbol{w}^T\boldsymbol{x}_0}{\|\boldsymbol{w}\|}$$

For point $\boldsymbol{x_i}$:

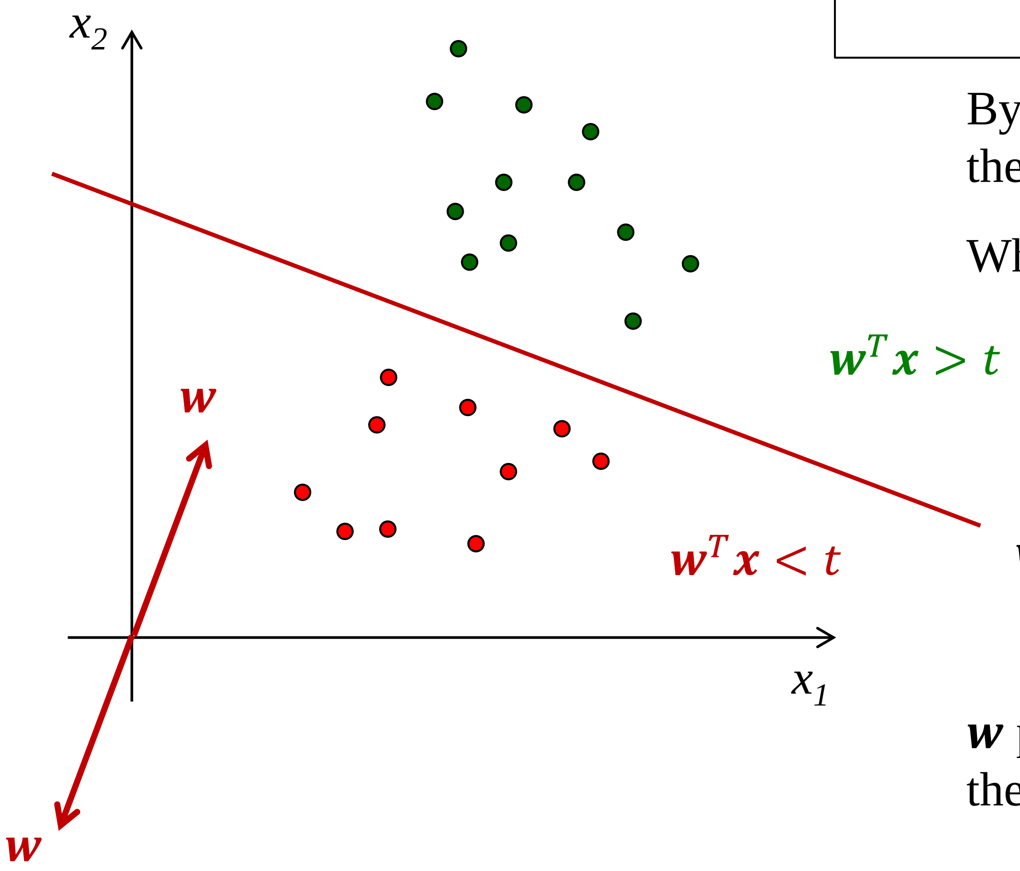
Margin: $z_i = \frac{y_i(\boldsymbol{w}^T\boldsymbol{x_i} - t)}{\|\boldsymbol{w}\|}$

$\boldsymbol{w}^T\boldsymbol{x} = t$

Classifier geometry – *w* and *t*

Non-homogeneous:

$$\boldsymbol{w}^T\boldsymbol{x} - t = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - t > 0$$



x_2
x_1
$\boldsymbol{w}$
$\boldsymbol{w}$
$\boldsymbol{w}^T\boldsymbol{x} > t$
$\boldsymbol{w}^T\boldsymbol{x} < t$
$\boldsymbol{w}^T\boldsymbol{x} = t$

By definition, the vector $\boldsymbol{w}$ points in the direction of the positive class.

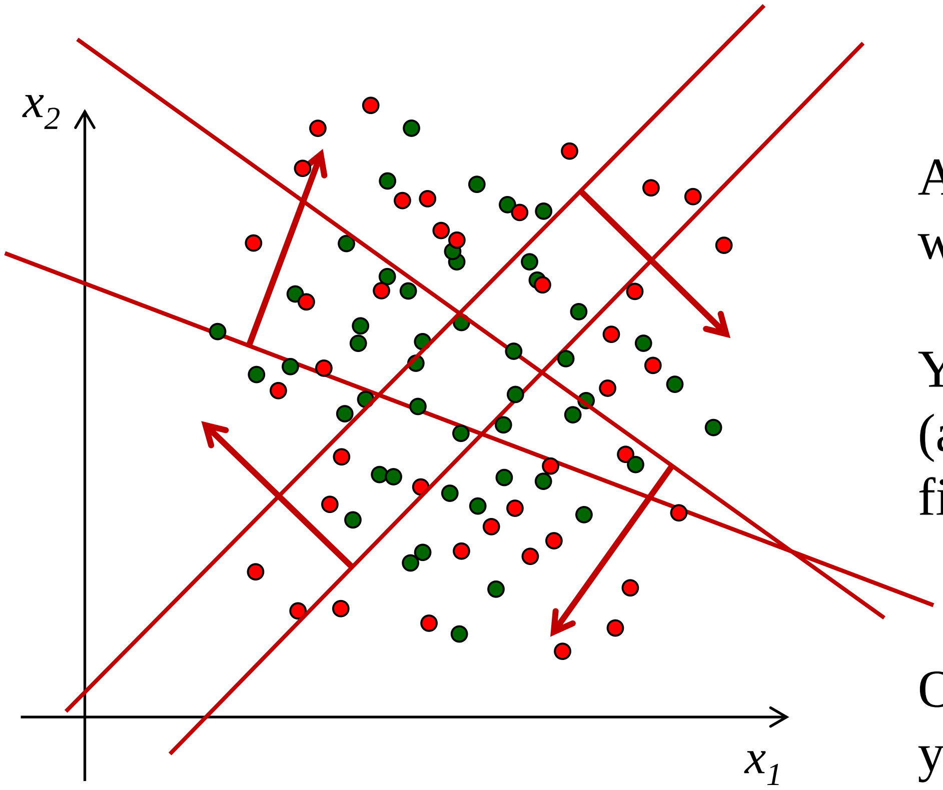
What if the classes here are swapped?

$$\boldsymbol{w} \leftarrow (-\boldsymbol{w})$$

$$t \leftarrow (-t)$$

$\boldsymbol{w}$ points in the (relative) direction of the positive class

Classifier geometry – ***w*** and *t*



An appropriate choices of $\boldsymbol{w}$ and t will achieve any decision line

You can start with the line equation (and direction of positive class) and figure out $\boldsymbol{w}$ and t

Or, alternative, if given $\boldsymbol{w}$ and t, you can figure out the classification line

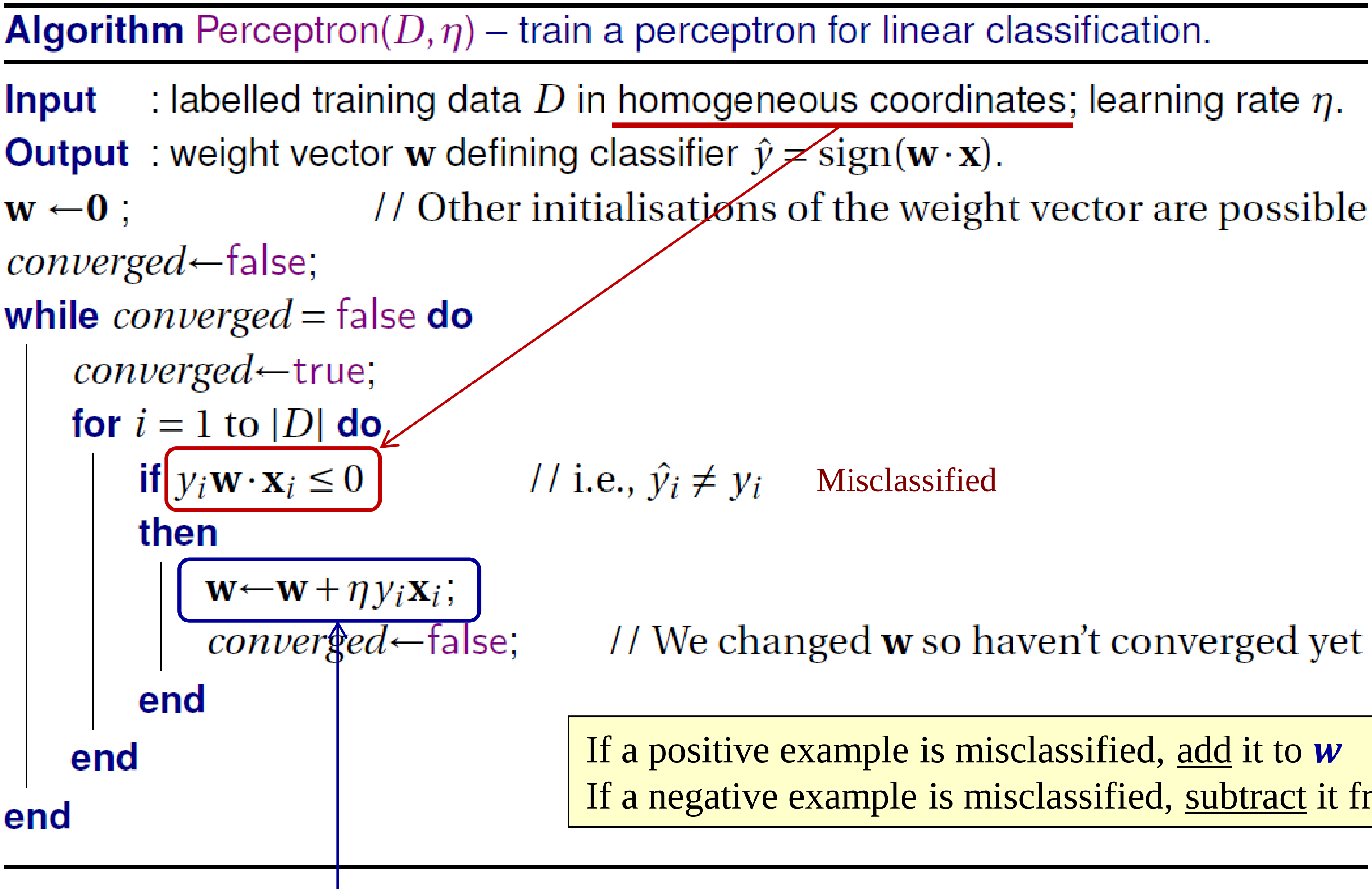
Non-homogeneous:

$$\boldsymbol{w}^T\boldsymbol{x} - t = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - t > 0$$

The perceptron training algorithm

When will this algorithm halt?

$D = \{ (\mathbf{x}_i, y_i) \}$



Algorithm Perceptron(D, η) – train a perceptron for linear classification.

```
Input  : labelled training data $D$ in homogeneous coordinates; learning rate $\eta$.
Output : weight vector $\mathbf{w}$ defining classifier $\hat{y} = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x})$.
$\mathbf{w} \leftarrow \mathbf{0}$ ;            // Other initialisations of the weight vector are possible
$converged \leftarrow$ false;
while $converged =$ false do
    $converged \leftarrow$ true;
    for $i = 1$ to $|D|$ do
        if $\boxed{y_i \mathbf{w} \cdot \mathbf{x}_i \leq 0}$            // i.e., $\hat{y}_i \neq y_i$    Misclassified
        then
            $\boxed{\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i;}$
            $converged \leftarrow$ false;       // We changed $\mathbf{w}$ so haven't converged yet
        end
    end
end
```

If a positive example is misclassified, add it to $\boldsymbol{w}$
If a negative example is misclassified, subtract it from $\boldsymbol{w}$

All components of homogeneous $\boldsymbol{w}$ are updated (including $\boldsymbol{w}_{k+1} = -t$)

Perceptron demo

Matlab example

Perceptron duality

- Every time a training example $\boldsymbol{x}_i$ is misclassified, the amount $\eta y_i \mathbf{x}_i$ is added to the weight vector $\boldsymbol{w}$
- After training is completed, each example $\boldsymbol{x}_i$ has been misclassified α_i times
- Thus the weight vector can be written as

$$\boldsymbol{w} = \eta \sum_{\boldsymbol{i}} \alpha_i y_i \boldsymbol{x}_i$$

Assuming the initial value of $\boldsymbol{w}$ was initialize to $\boldsymbol{0}$

So the weight vector is a linear combination of the training instances

- So, alternatively, we can view perceptron learning as learning the α_i coefficients and then, when finished, constructing $\boldsymbol{w}$
  - This perspective comes up again (soon) in support vector machines