Linear Classifiers: The Perceptron

Intro to ML 466/566
Winter 2021

Administrivia

- Very small change to question 5 on the homework (566 students only)

$$\ell(\boldsymbol{w}) = \begin{cases} \frac{1}{2}(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x}))^2 + \lambda \sum_{i=1}^{p} w_i^2, & \text{if } |\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \delta \\ \delta|\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})| - \frac{\delta^2}{2} + \lambda \sum_{i=1}^{p} w_i^2, & \text{otherwise} \end{cases}$$

The Perceptron

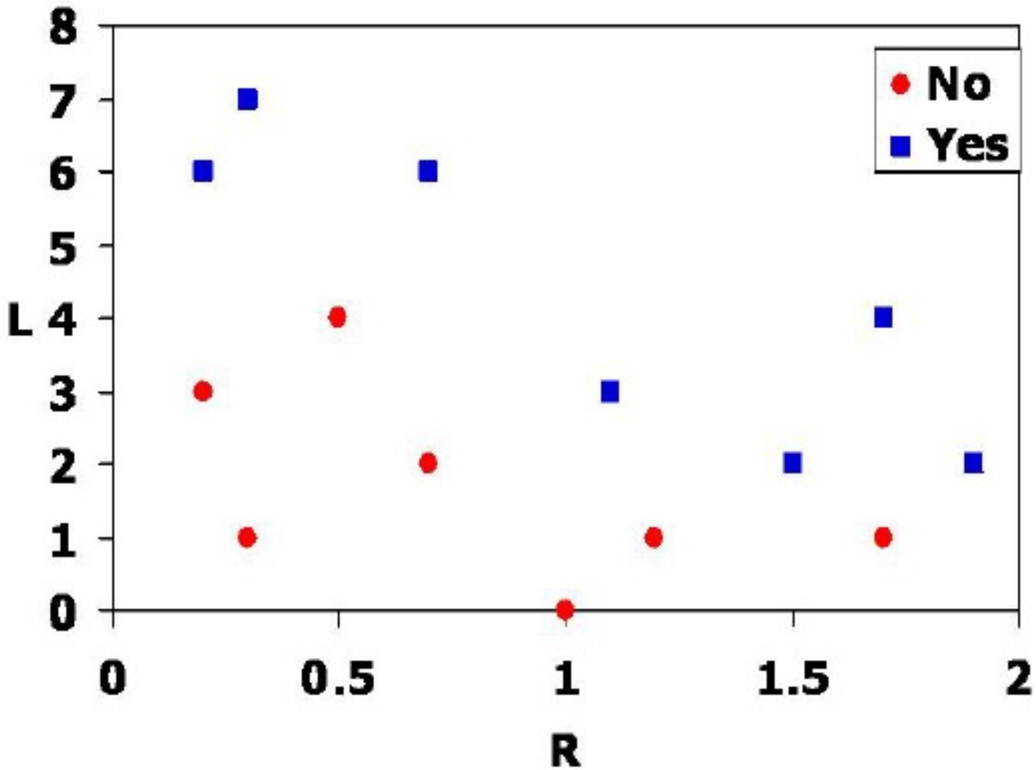
- Introduces the idea of a linear decision boundary
  - Decision trees are piecewise linear, but non-linear overall
- Introduces the idea of an iterative learning algorithm
  - Very similar to stochastic gradient descent, which is pervasive in ML

Bankruptcy example

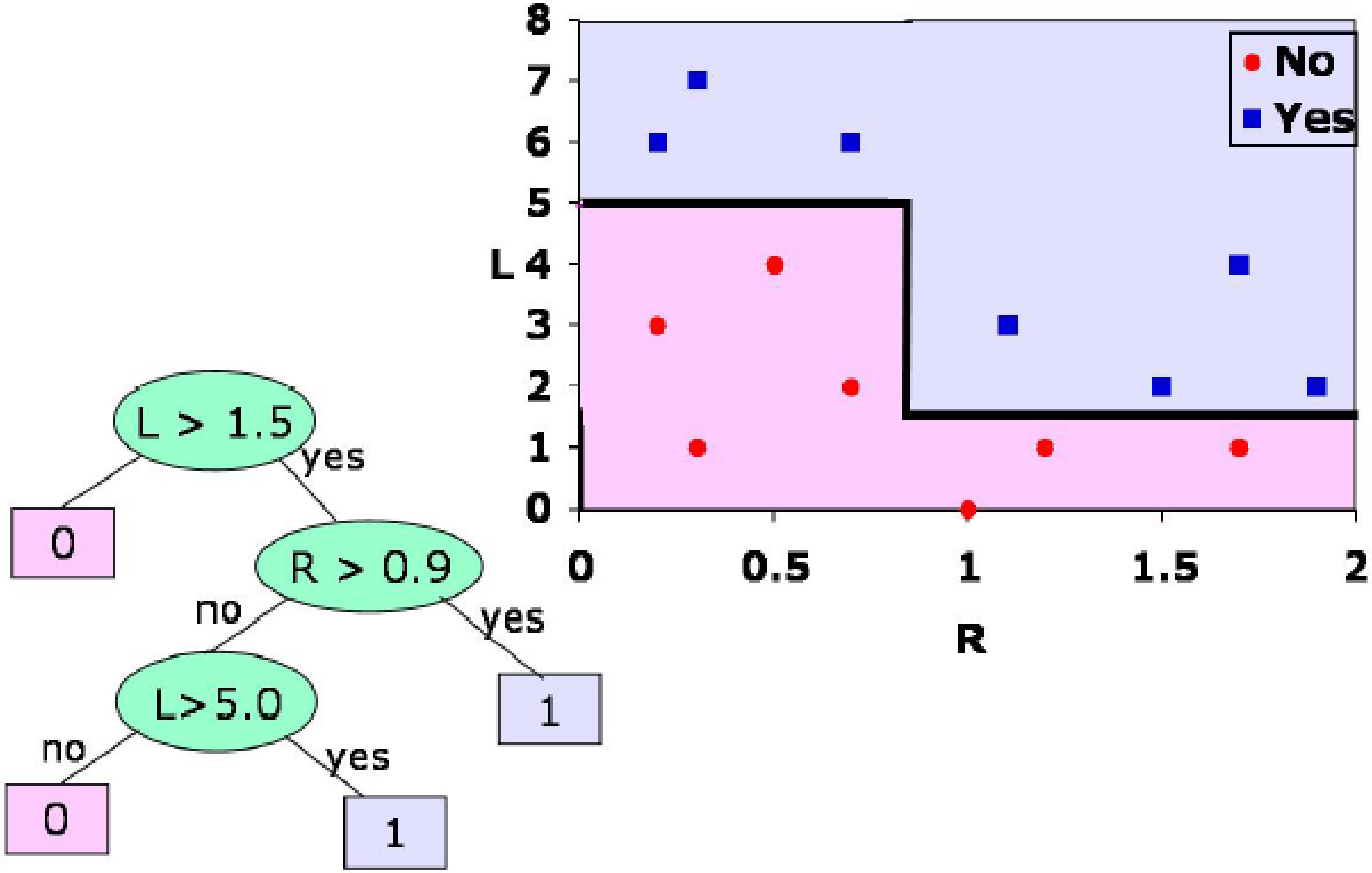
R is the ratio of earnings to expenses

L is the number of late payments on credit cards over the past year.

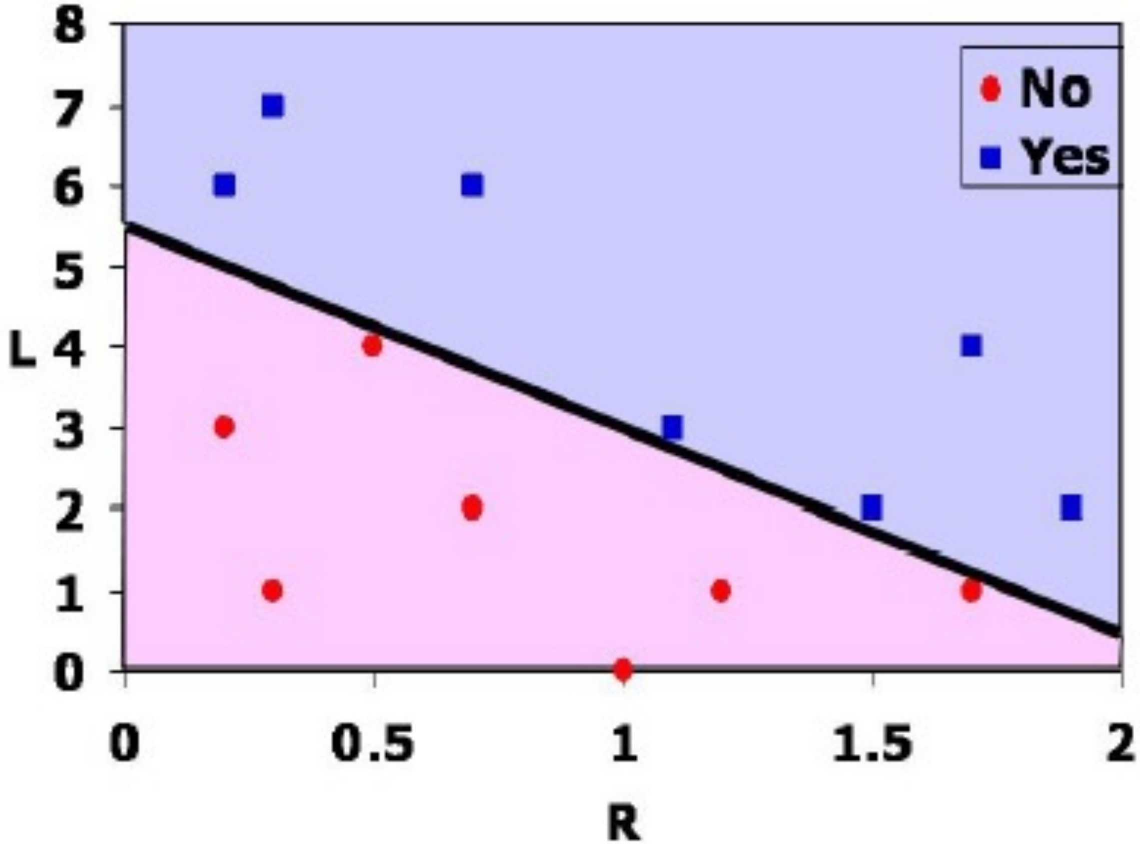
We would like to draw a **linear separator**, and thus get a classifier.



Classification as Boundary: E.g. Decision Tree Boundary



Simple Linear Boundary



What's different here compared to the decision tree?

This is a **linear** decision boundary

- Recall the equation of a line:

$$y = b + ax$$
$$y - ax - b = 0$$
$$y + ax + b = 0 \qquad \text{(why?)}$$

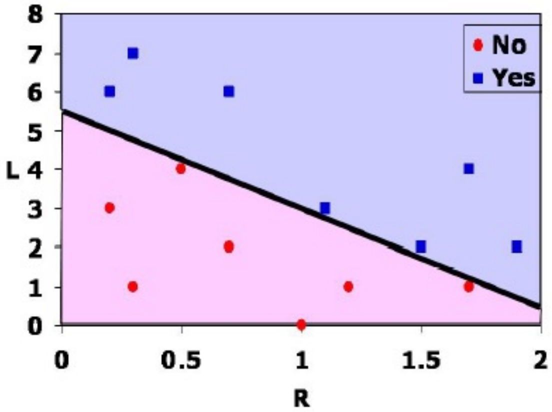
- That notation is a bit confusing because we often call our data matrix x and our labels y. So instead let's rename the axes

$$x_1 + ax_2 + b = 0$$

- Multiply both sides by a new weight

$$cx_1 + cax_2 + cb = 0$$
$$cx_1 + ax_2 + b = 0 \qquad \text{(why?)}$$

- Here we have L and R instead of x_1 and x_2



- a and c control the **slope** of the line, b is offset (also called **the bias**)

Linear Hypothesis Class

- Line equation (assume 2D first):

  $w_2x_2 + w_1x_1 + b=0$

- **Fact1**: All points (x_1, x_2) lying **on** the line make the equation true.

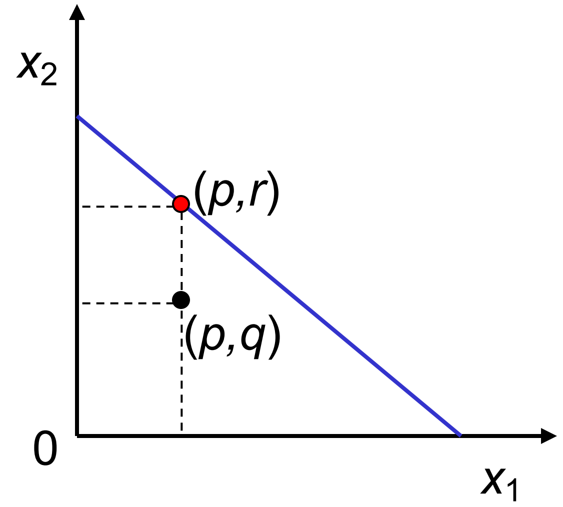
- **Fact2**: The line separates the plane in two half-planes. I.e. the space is separated into two halves by the line

- **Fact3**: The points (x_1, x_2) in one half-plane give us an inequality with respect to 0, which has the same direction for each of the points in the half-plane. E.g. > 0.
  - I.e. all points (x_1, x_2) on one side of the line will produce a number with the same sign when equation $w_2x_2 + w_1x_1 + b$ is used

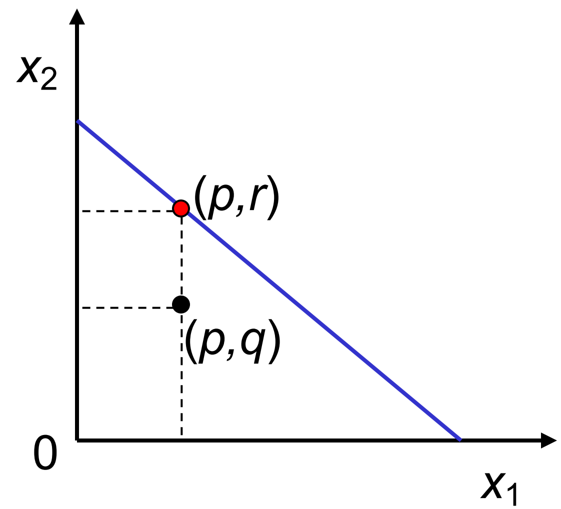
- **Fact4**: The points (x_1, x_2) in the other half-plane give us the reverse inequality with respect to 0. E.g. < 0

Fact 3 proof

$w_2x_2+w_1x_1+b=0$



Fact 3 proof



$w_2x_2+w_1x_1+b=0$

We can write it as: $x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$

(p,r) is on the line so:

$$r = -\frac{w_1}{w_2}p - \frac{b}{w_2}$$

For $q<r$, so we have: $q < r = -\frac{w_1}{w_2}p - \frac{b}{w_2}$ *i.e.*

$$w_2q + w_1p + b < 0 \qquad \text{if } w_2 > 0$$

$$w_2q + w_1p + b > 0 \qquad \text{if } w_2 < 0$$

Since (p,q) was an arbitrary point in the half-plane, we say that the same direction of inequality holds for any other point of the half-plane. Fact 4 is similar.

This means we can use the equation of the line as a linear classifier!

$$\mathbf{x} = [x_1,\ x_2,\ \ldots,\ x_m]$$

Data (row of data matrix)

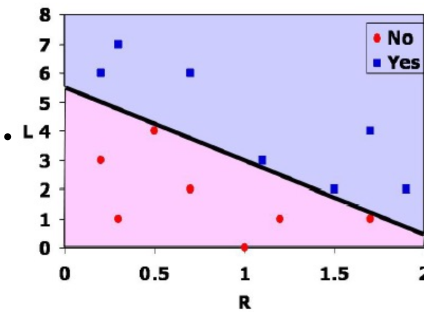
$$\mathbf{w} = [w_1,\ w_2,\ \ldots,\ w_m]$$

Weights we will learn

$$h(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

"*sign*" outputs +1 or –1 depending on the sign of wx+b

Then, assign +1 to blue, and -1 to red, or vice versa.



One small change (notational simplicity)

$$h(\mathbf{x}, \mathbf{w}, b) = \text{sign}\left(\left(\sum_{i=1}^{m} w_i x_i\right) + b\right) \qquad h(\mathbf{x}, \mathbf{w}) = \text{sign}\left(\left(\sum_{i=1}^{m} w_i x_i\right) + w_0\right)$$

$$\boxed{w_0 = b}$$

$$\boxed{\mathbf{x} = [1, x_1, .., x_m]} \qquad \boxed{\mathbf{w} = [w_0, w_1, .., w_m]}$$

$$\begin{aligned} h(\mathbf{x}, \mathbf{w}) &= \text{sign}\left(\sum_{i=0}^{m} w_i x_i\right) \\ &= \text{sign}(\mathbf{w} \cdot \mathbf{x}) \\ &= \text{sign}(\mathbf{w}^T \mathbf{x}) \end{aligned}$$

Now we have to learn!

$$h(\mathbf{x}, \mathbf{w}) = \operatorname{sign}\left(\sum_{i=0}^{m} w_i x_i\right)$$

$$= \operatorname{sign}(\mathbf{w} \cdot \mathbf{x})$$

$$= \operatorname{sign}(\mathbf{w}^T \mathbf{x})$$

- What is our ***w***?

Learning Algorithm

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

Start with random $\mathbf{w}$'s

Training tuples (1… n)

$\mathbf{x}^1, y^1$

$\mathbf{x}^2, y^2$

...

$\mathbf{x}^n, y^n$

Learning Algorithm

Work through the training tuples (1… n)

For each *misclassified* training tuple

i.e. $\text{sign}(\mathbf{w}^T\mathbf{x}^k) \neq y^k$

Update **w**:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \cdot y^k \cdot \mathbf{x}^k$$

η is the *learning rate*

Why is this a good update rule?

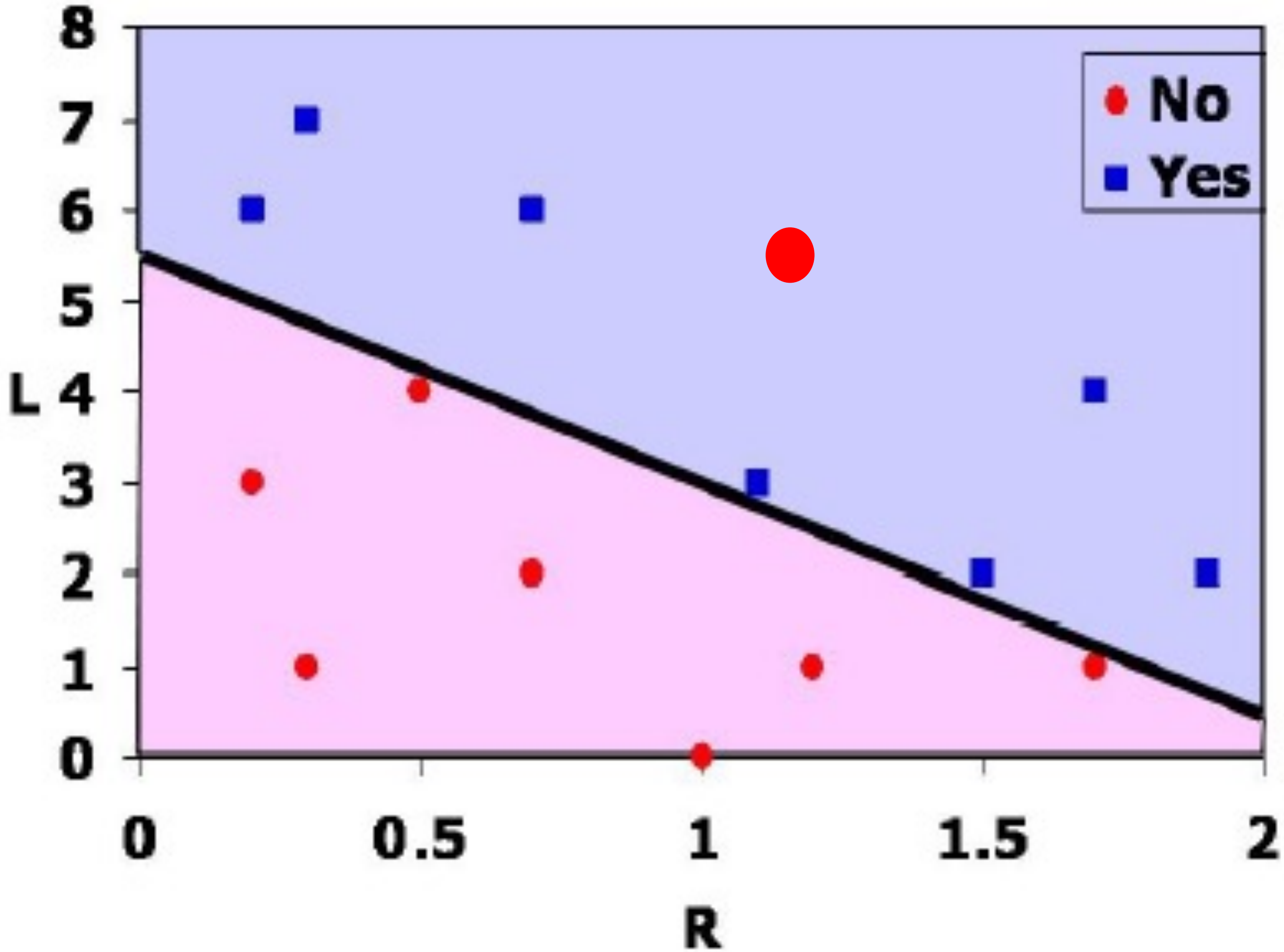
It can be shown that if the data is **linearly separable**, and we repeat this procedure many times, we will get a line that separates the training tuples.

You will show this on your homework!

The learning rate

- “Learning rate” is a term you will see often
- It governs how quickly parameters change during learning
- Set it too small and it will take a very long time to converge
- Set it too large and the algorithm will actually diverge (i.e. make more and more errors)

Linearly Separable



Some intuition behind the perceptron learning algorithm

Recall the dot product

$$A \cdot B = \sum_i A_i B_i$$

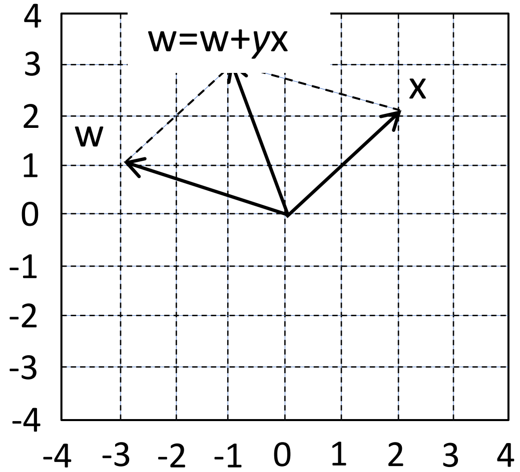
$$= ||A|| * ||B|| * cos(\theta)$$

- ||A|| and ||B|| can only be positive
- cos(θ) is negative when the angle is between ½ pi and 3/2 pi
  - that is, when the angle is obtuse
  - so we want obtuse when y = -1, acute otherwise

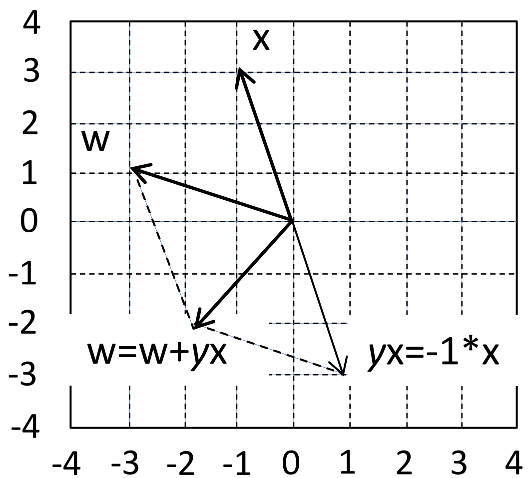
- Note that we want $w^T*x*y > 0$, which is not true when $\text{sign}(\mathbf{w}^T \mathbf{x}^k) \neq y^k$

Sign of dot product and **misclassification**

y=+1
w.x<0
y.w.x<0



y=-1
w.x>0
y.w.x<0



The Perceptron

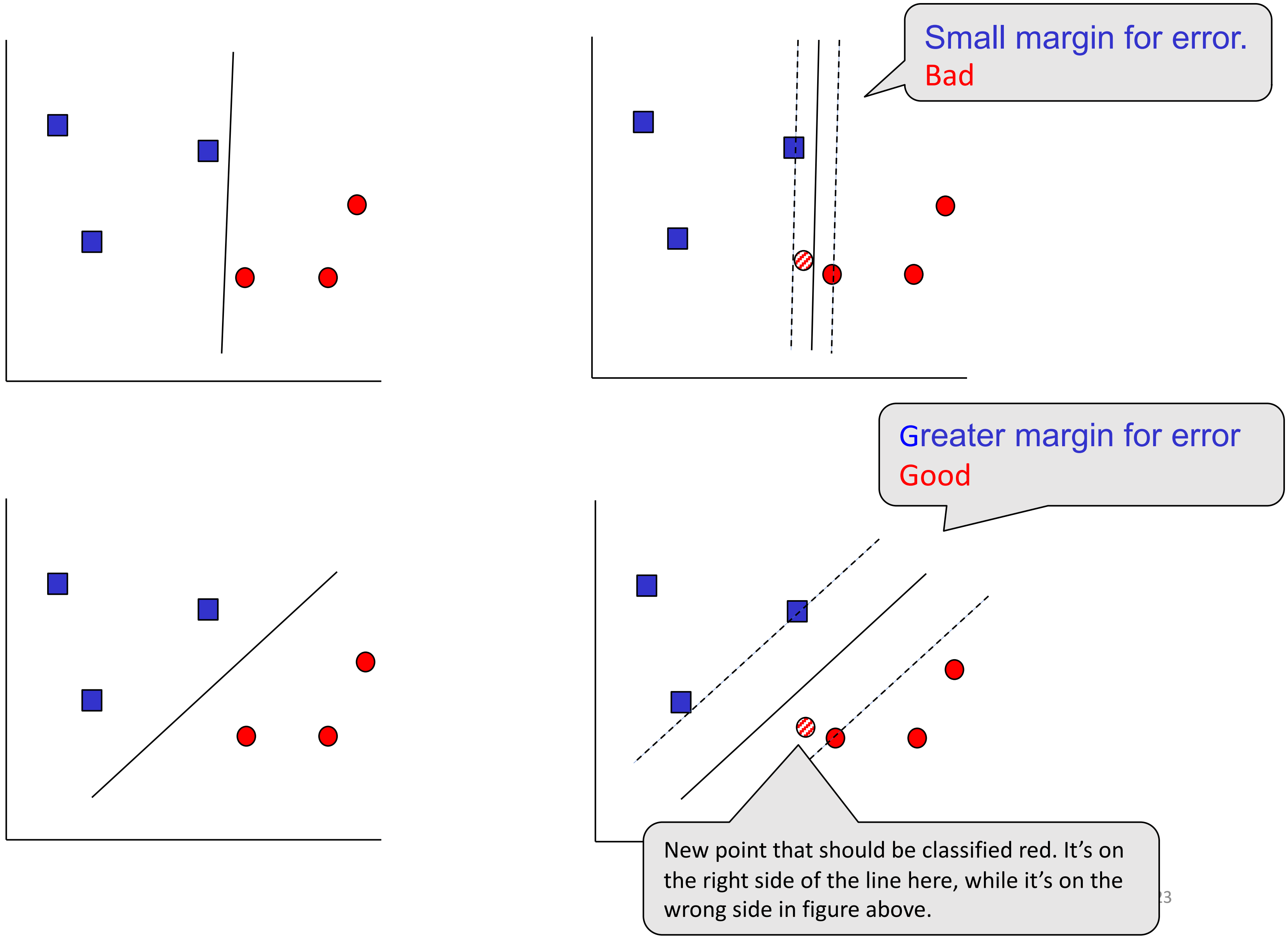
- This learning update rule is so simple it can be coded in excel!

The Perceptron

- This learning update rule is so simple it can be coded in excel!

- Caveats:
  - If linearly separable, this algorithm will learn “a” line that separates the classes. The order of the data will effect which separating line it will learn
  - If not linearly separable, this algorithm will never terminate

Margin for different separators

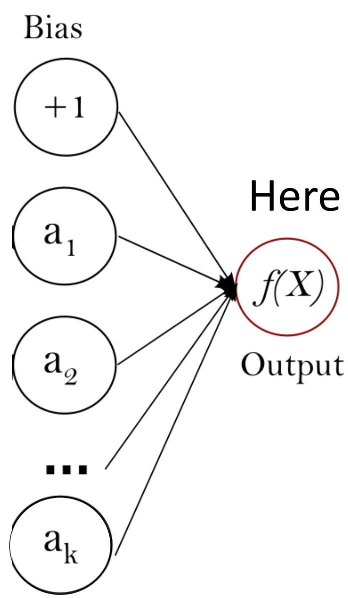


The Perceptron

- A perceptron is actually a **very simple neural network**
  - Here “sign” is the *activation* function
    - Step function with step point n=0
  - There are other kinds of activation functions (we will talk more about this later)

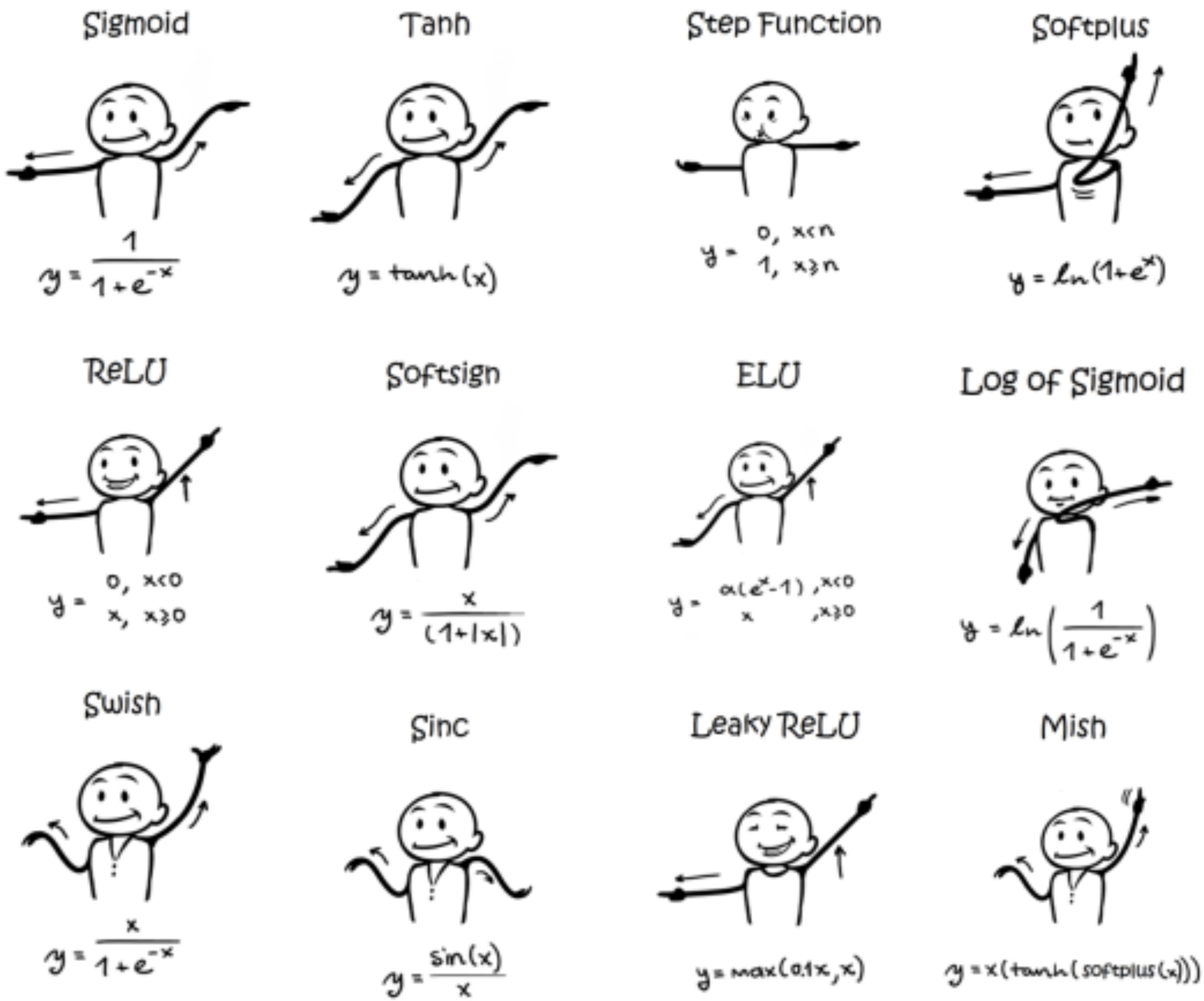
Input features
[a_1 … a_k]

Here f(x) is the sign function



https://scikit-learn.org/stable/modules/neural_networks_supervised.html

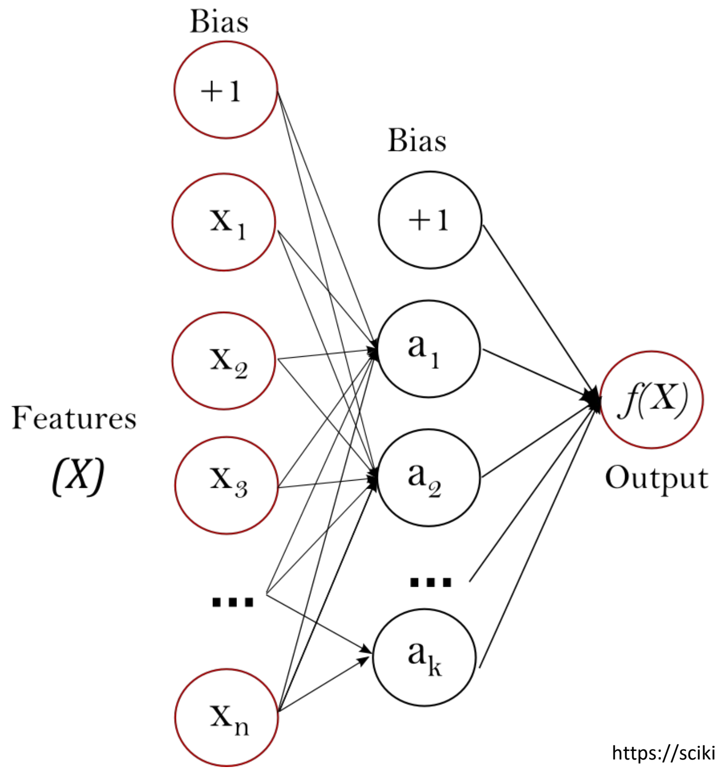
Activation functions



https://sefiks.com/2020/02/02/dance-moves-of-deep-learning-activation-functions/

The Perceptron

- You may run into the term *multilayer perceptron*
  - Several layers of perceptrons chained together
  - Each layer consists of multiple perceptrons, each with their own learned ***w***
  - The output of the activation function becomes the input to the next layer of perceptrons



https://scikit-learn.org/stable/modules/neural_networks_supervised.html

Take home messages

- Perceptrons: the simplest NN
- Perceptron learning alg
  - So simple we can code it in excel!
  - Guaranteed to converge when data in lin. Separable

- Some fun history (won't be on your exam)
  - https://web.csulb.edu/~cwallis/artificialn/History.htm