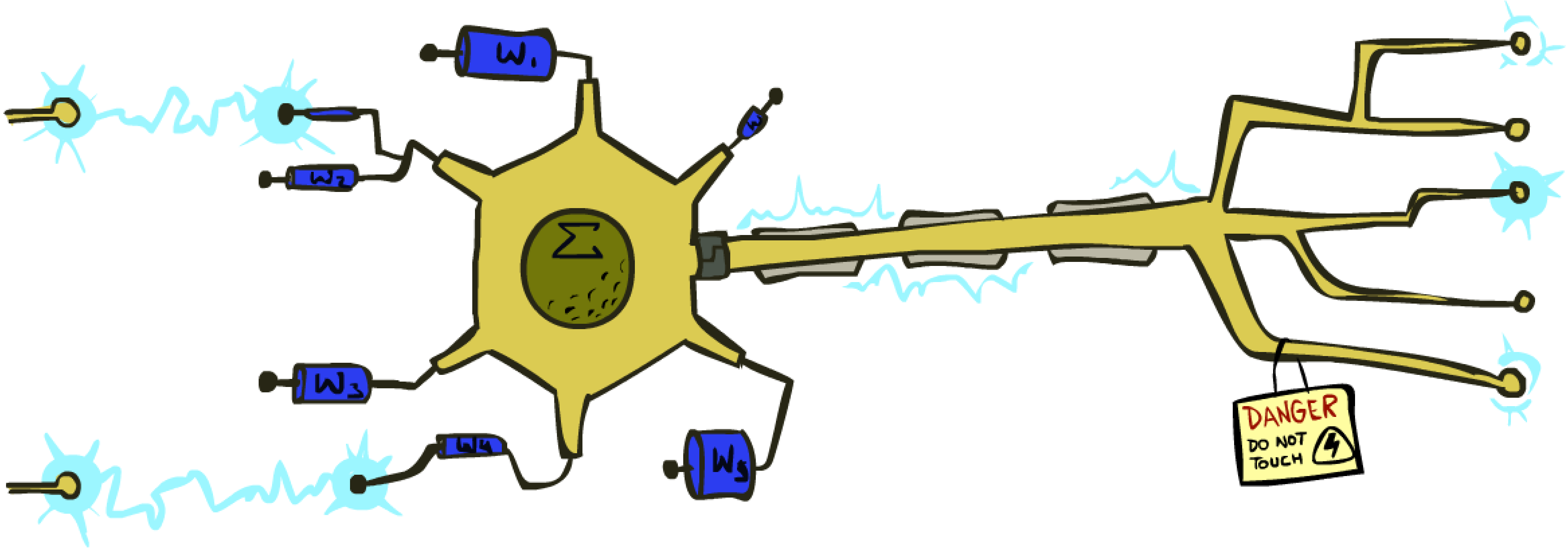
CS 188: Artificial Intelligence

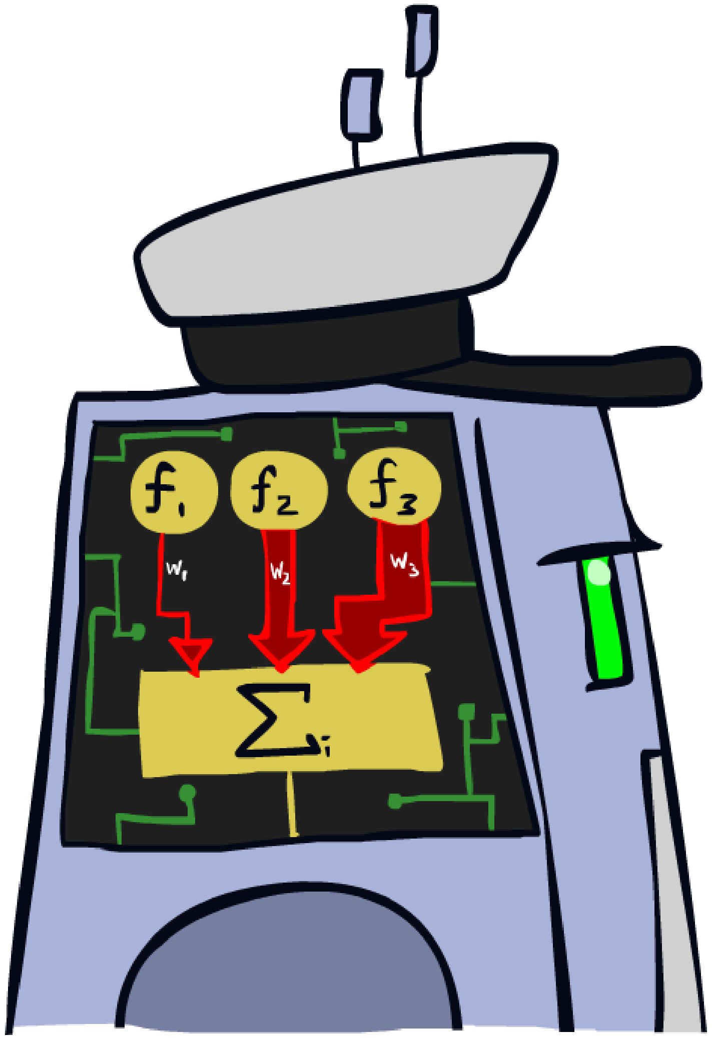
Perceptrons and Logistic Regression



Spring 2023

University of California, Berkeley

Linear Classifiers



Feature Vectors

x | $f(x)$ | y

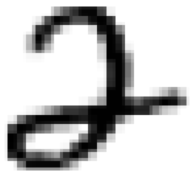
```
Hello,

Do you want free printr
cartriges?  Why pay more
when you can get them
ABSOLUTELY FREE!  Just
```

→

```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

→ SPAM or +



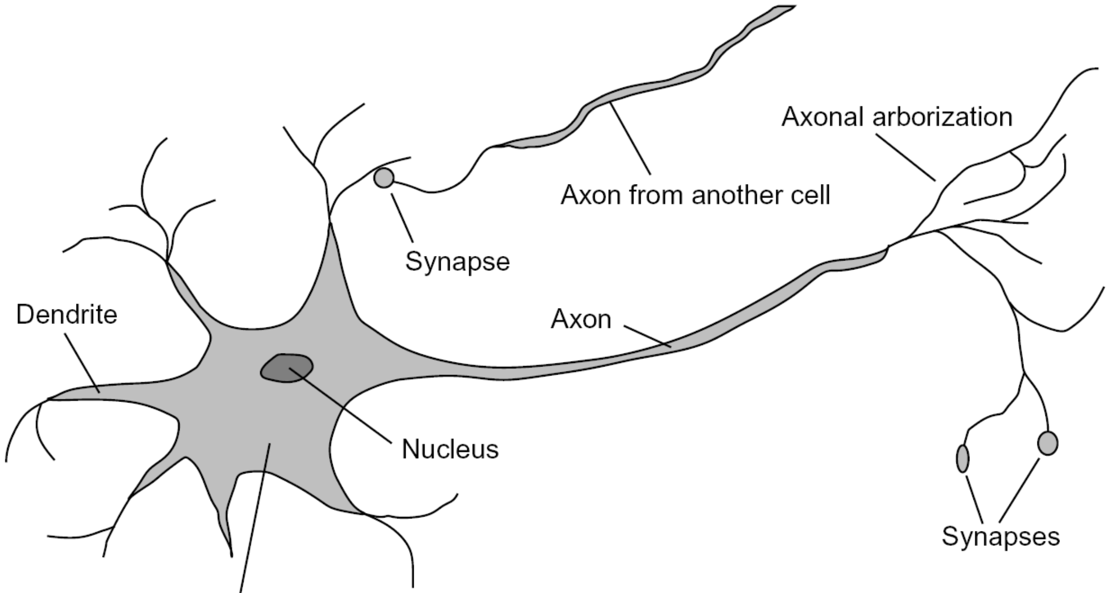
→

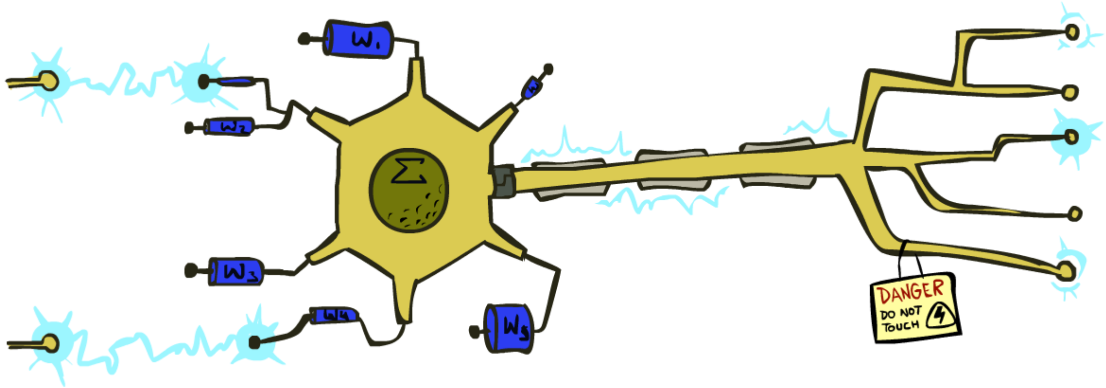
```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

→ “2”

Some (Simplified) Biology

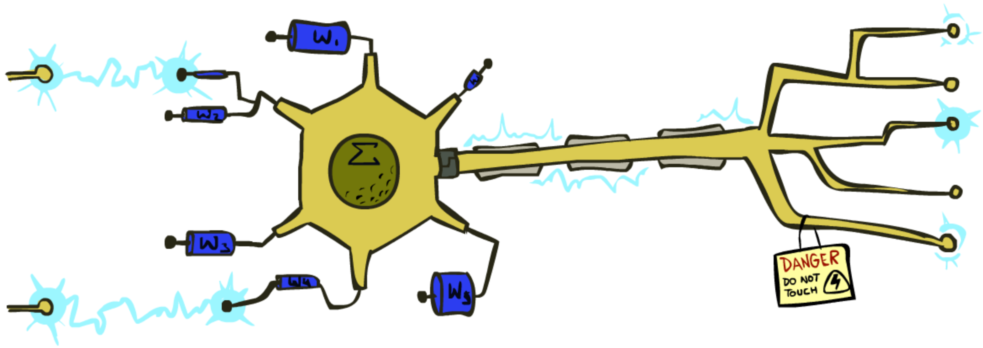
- Very loose inspiration: human neurons





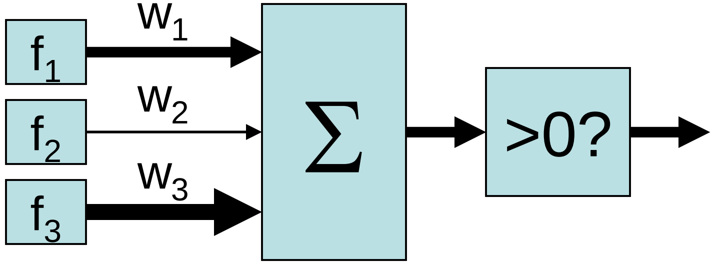
Linear Classifiers

- Inputs are feature values
- Each feature has a weight
- Sum is the activation



$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- If the activation is:
  - Positive, output +1
  - Negative, output -1



Weights

Dot product $w \cdot f$ *positive means the positive class (spam)*

$$w \cdot f(x_1)$$

```
# free      :  4
YOUR_NAME   : -1
MISSPELLED  :  1
FROM_FRIEND : -3
...
```

```
# free      : 2
YOUR_NAME   : 0
MISSPELLED  : 2
FROM_FRIEND : 0
...
```

$$w \cdot f(x_2)$$

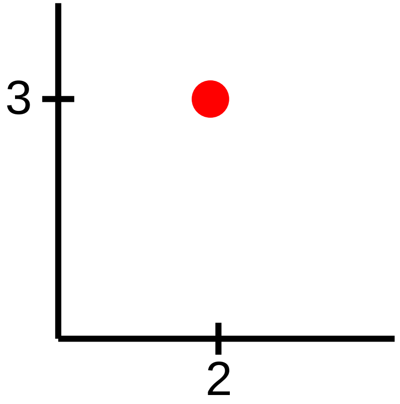
```
# free      :  4
YOUR_NAME   : -1
MISSPELLED  :  1
FROM_FRIEND : -3
...
```

```
# free      : 0
YOUR_NAME   : 1
MISSPELLED  : 1
FROM_FRIEND : 1
...
```

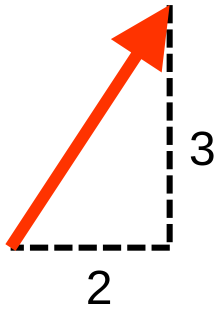
Do these weights make sense for spam classification?

Review: Vectors

- A tuple like (2,3) can be interpreted two different ways:



A **point** on a coordinate grid

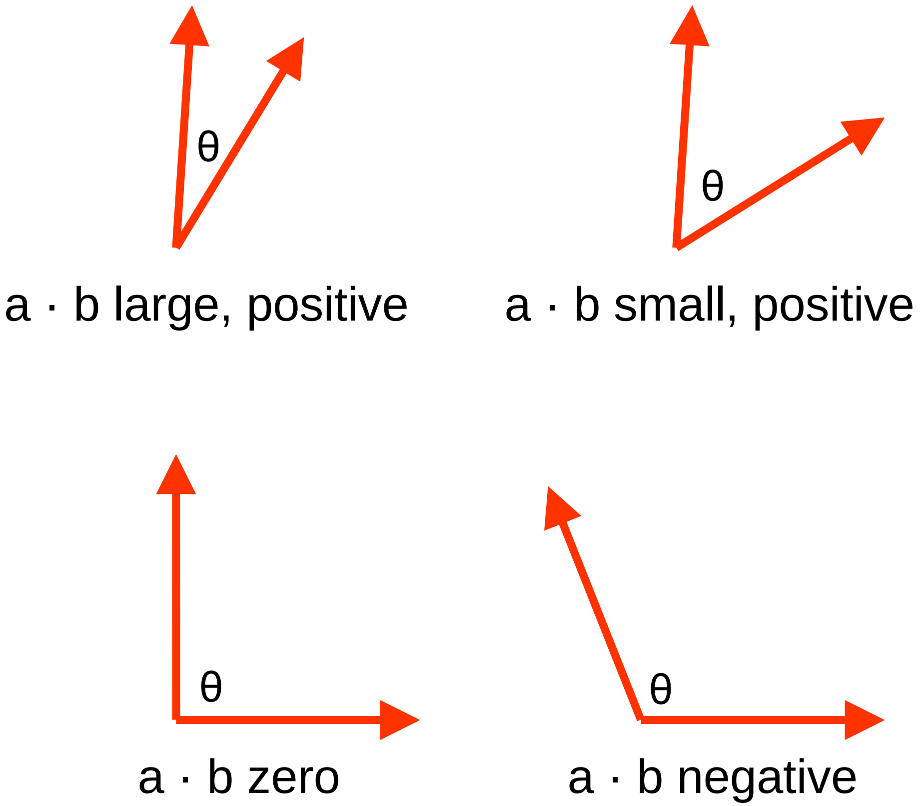


A **vector** in space. Notice we are not on a coordinate grid.

- A tuple with more elements like (2, 7, -3, 6) is a point or vector in higher-dimensional space (hard to visualize)

Review: Vectors

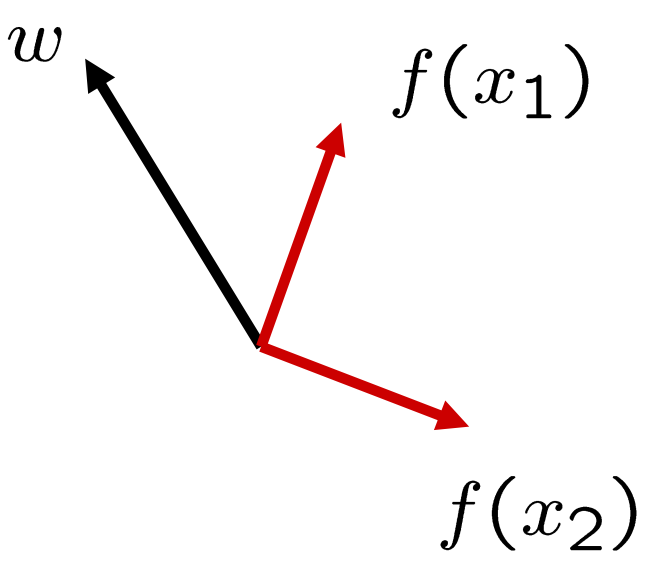
- Definition of dot product:
  - $a \cdot b = |a|\ |b| \cos(\theta)$
  - θ is the angle between the vectors a and b
- Consequences of this definition:
  - Vectors closer together
    = “similar” vectors
    = smaller angle θ between vectors
    = larger (more positive) dot product
  - If $\theta < 90°$, then dot product is positive
  - If $\theta = 90°$, then dot product is zero
  - If $\theta > 90°$, then dot product is negative



Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples

```
# free      : 4
YOUR_NAME   :-1
MISSPELLED  : 1
FROM_FRIEND :-3
...
```

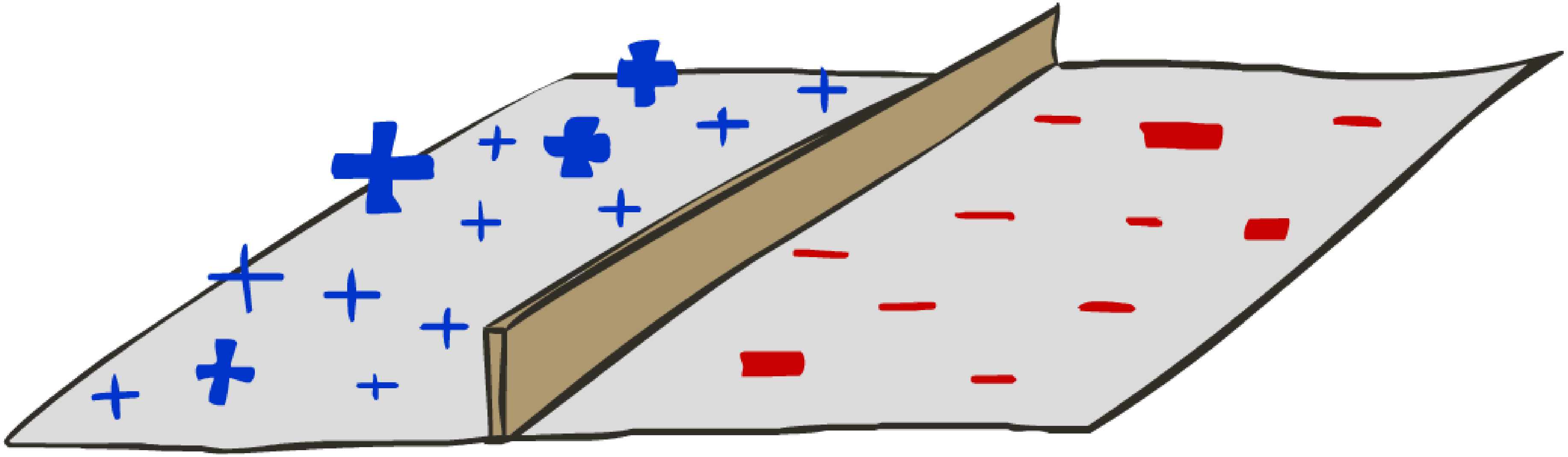



```
# free      : 2
YOUR_NAME   : 0
MISSPELLED  : 2
FROM_FRIEND : 0
...
```

```
# free      : 0
YOUR_NAME   : 1
MISSPELLED  : 1
FROM_FRIEND : 1
...
```

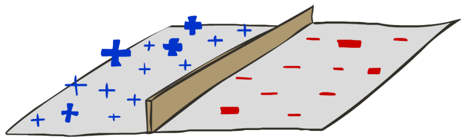
Dot product $w \cdot f$ *positive means the positive class*

Decision Rules



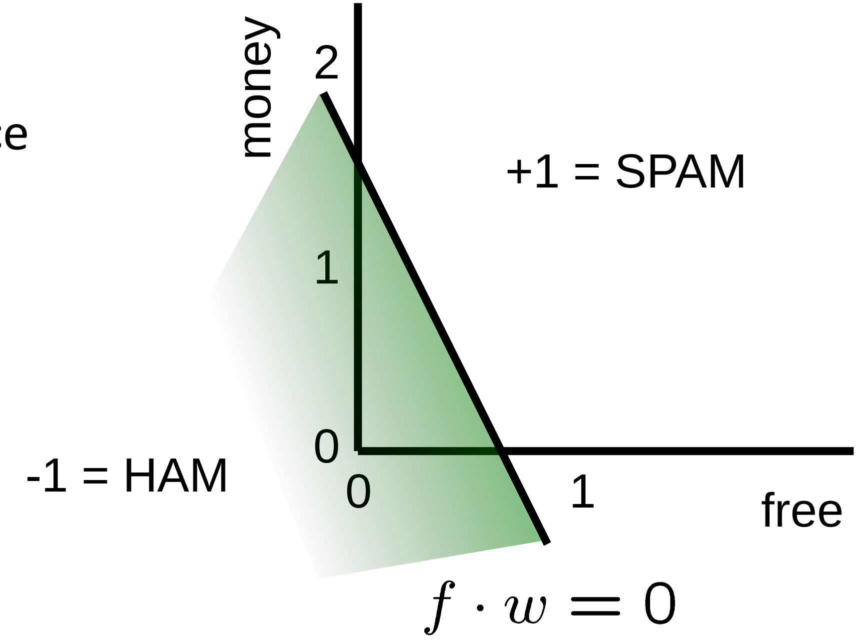
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane (divides space into two sides)
  - One side corresponds to Y=+1, the other corresponds to Y=-1
- In the example:
  - $f \cdot w > 0$ when 4*free + 2*money > 0
    $f \cdot w < 0$ when 4*free + 2*money < 0
    These equations correspond to two halves of the feature space
  - $f \cdot w = 0$ when 4*free + 2*money = 0
    This equation corresponds to the decision boundary (a line in 2D, a hyperplane in higher dimensions)



w

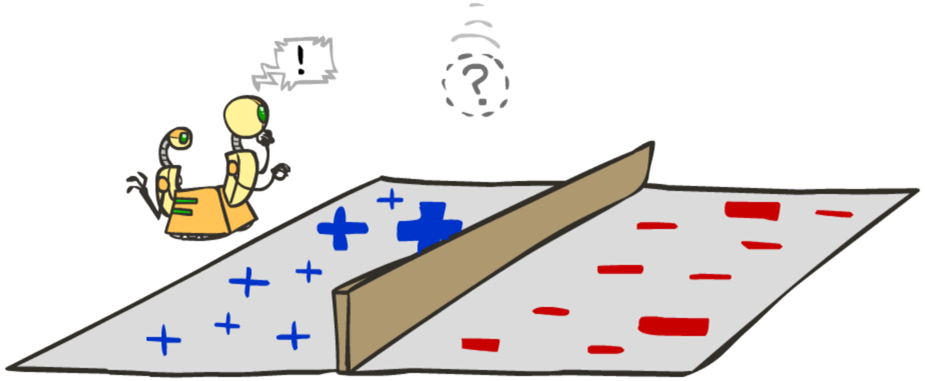
free	:	4
money	:	2



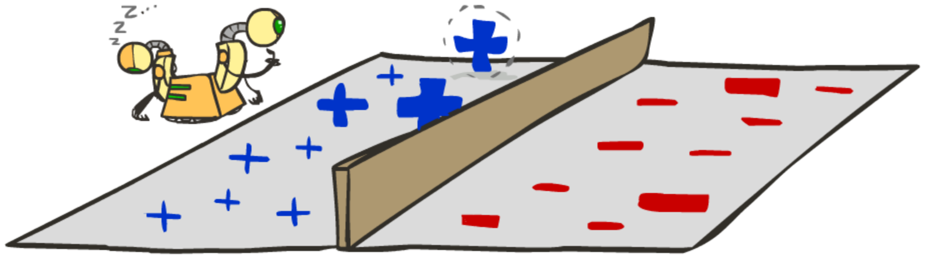
Weight Updates

Learning: Binary Perceptron

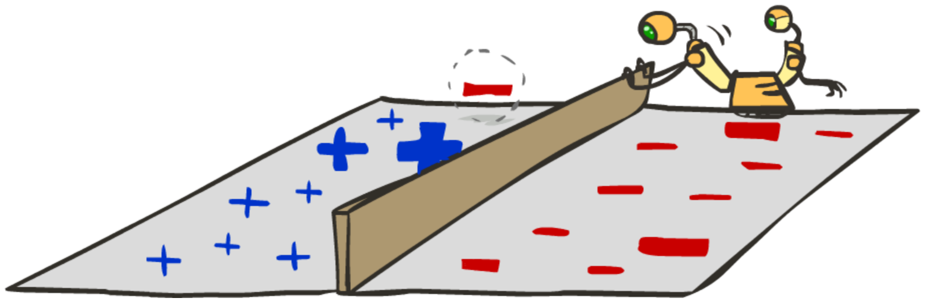
- Start with weights = 0
- For each training instance:
  - Classify with current weights



  - If correct (i.e., y=y*), no change!



  - If wrong: adjust the weight vector



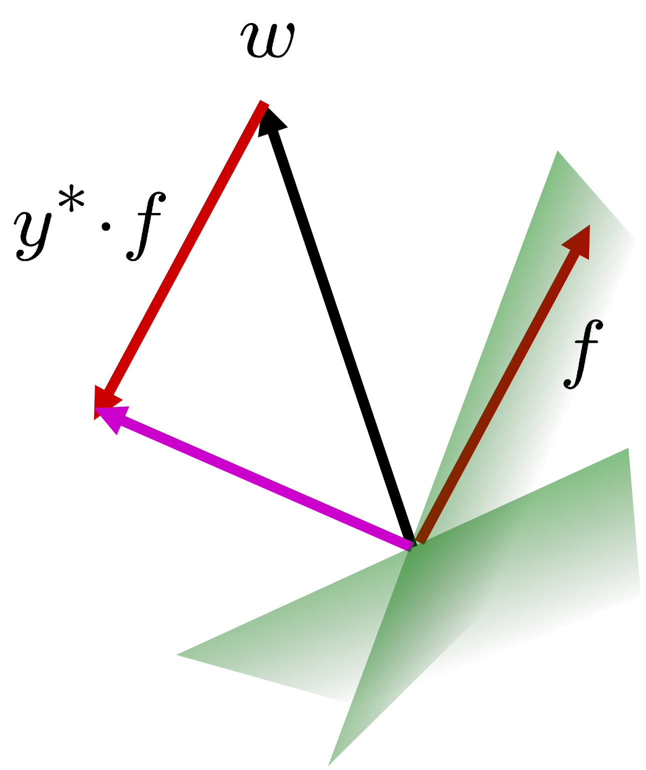
Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights

$$y = \begin{cases} +1 & \text{if } \ w \cdot f(x) \geq 0 \\ -1 & \text{if } \ w \cdot f(x) < 0 \end{cases}$$

  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

$$w = w + y^* \cdot f$$

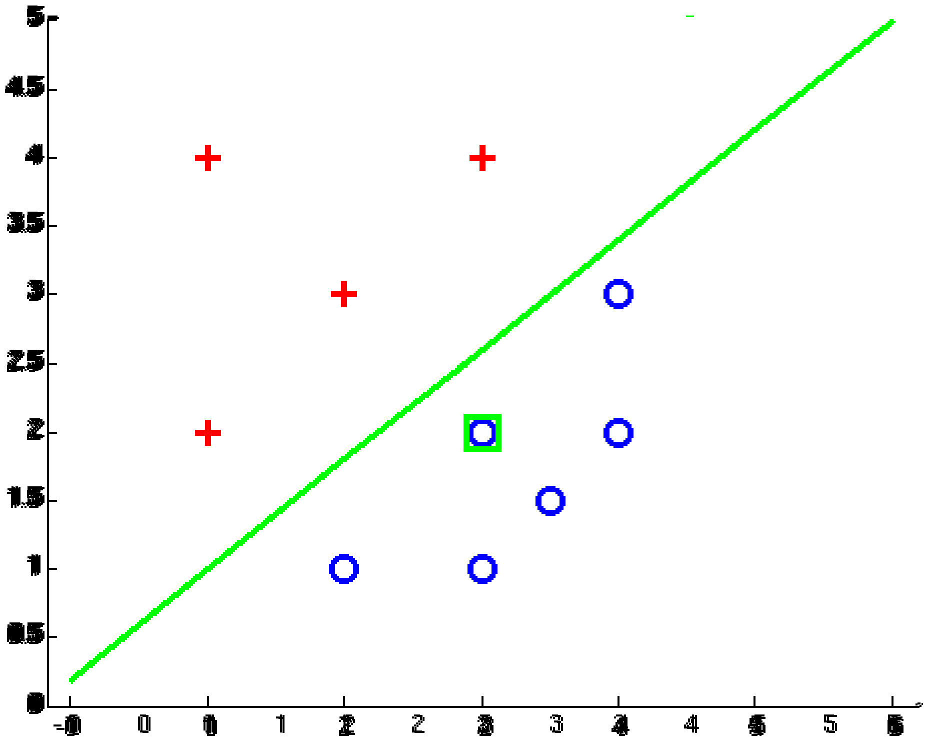


Learning: Binary Perceptron

- Misclassification, Case I:
  - $w \cdot f > 0$, so we predict +1
  - True class is -1
  - We want to modify w to w' such that dot product $w' \cdot f$ is *lower*
  - **Update if we misclassify a true class -1 sample: $w' = w - f$**
  - Proof: $w' \cdot f = (w - f) \cdot f = (w \cdot f) - (f \cdot f) = (w \cdot f) - |f|^2$
    Note that $|f|^2$ is always positive
- Misclassification, Case II:
  - $w \cdot f < 0$, so we predict -1
  - True class is +1
  - We want to modify w to w' such that dot product $w' \cdot f$ is *higher*
  - **Update if we misclassify a true class +1 sample: $w' = w + f$**
  - Proof: $w' \cdot f = (w + f) \cdot f = (w \cdot f) + (f \cdot f) = (w \cdot f) + |f|^2$
    Note that $|f|^2$ is always positive
- Write update compactly as $w' = w + y^* \cdot f$, where y^* = true class

Examples: Perceptron

- Separable Case



Multiclass Decision Rule

- If we have multiple classes:
  - A weight vector for each class:

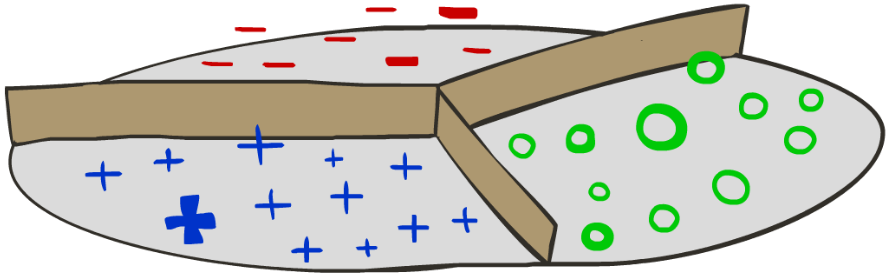
    $$w_y$$

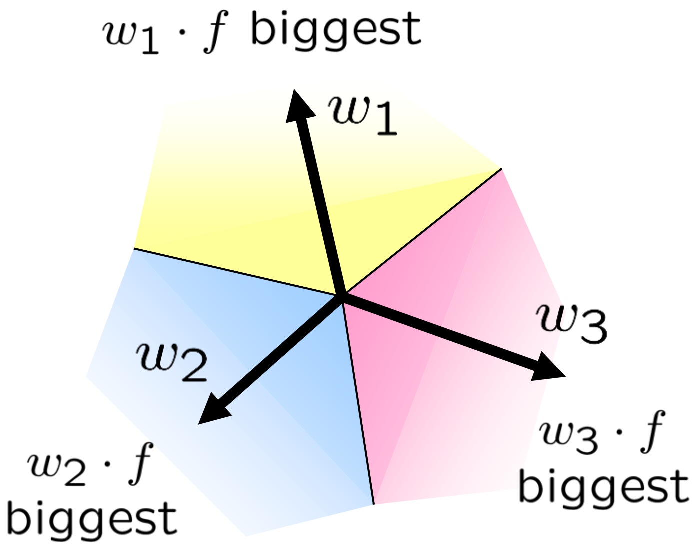
  - Score (activation) of a class y:

    $$w_y \cdot f(x)$$

  - Prediction highest score wins

    $$y = \arg\max_y \; w_y \cdot f(x)$$





Binary = multiclass where the negative class has weight zero

Learning: Multiclass Perceptron

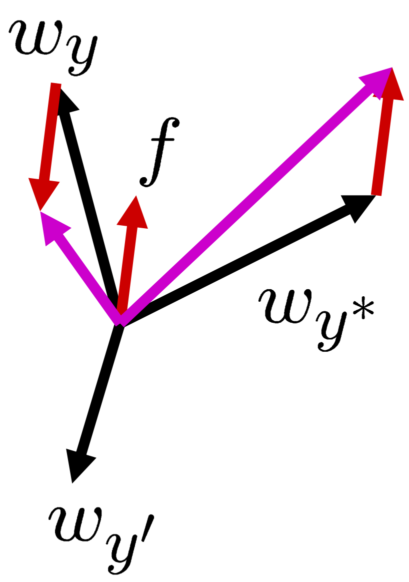
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

$$y = \arg\max_y \; w_y \cdot f(x)$$

- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer

$$w_y = w_y - f(x)$$

$$w_{y^*} = w_{y^*} + f(x)$$



Example: Multiclass Perceptron

"win the vote"

"win the election"

"win the game"

w_{SPORTS}

```
BIAS  : 1
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

$w_{POLITICS}$

```
BIAS  : 0
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

w_{TECH}

```
BIAS  : 0
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

Properties of Perceptrons

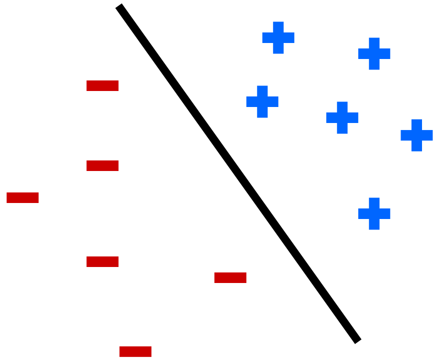
- Separability: true if some parameters get the training set perfectly correct

- Convergence: if the training is separable, perceptron will eventually converge (binary case)

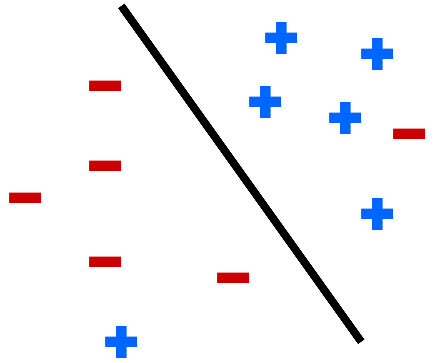
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2}$$

Separable

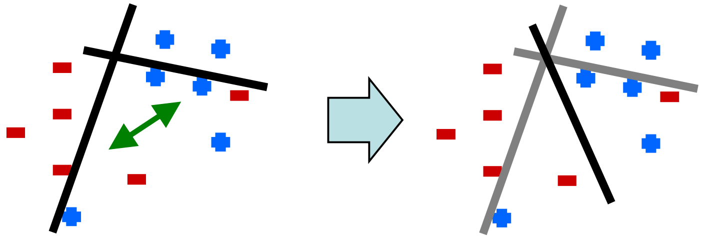


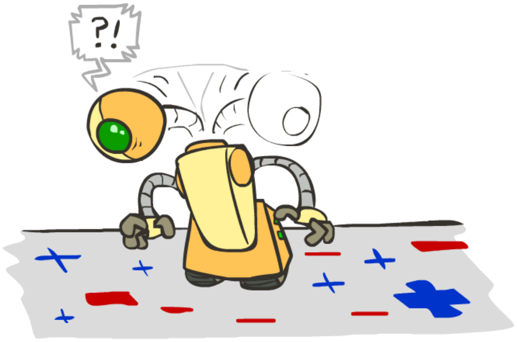
Non-Separable



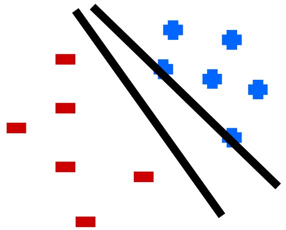
Problems with the Perceptron

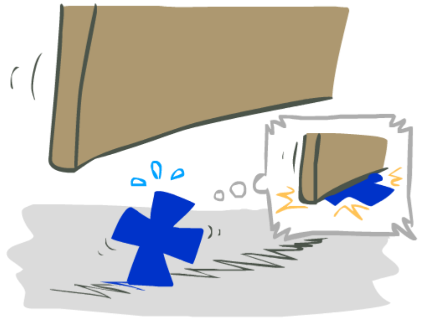
- Noise: if the data isn’t separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



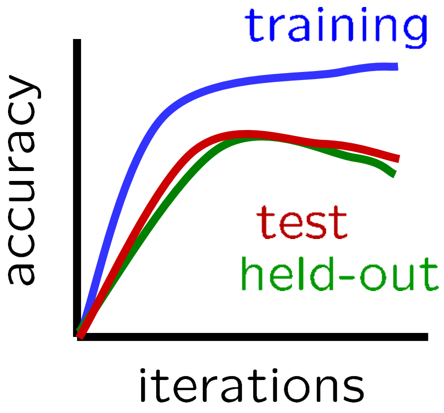


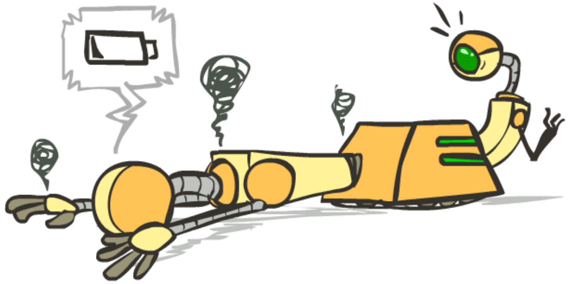
- Mediocre generalization: finds a “barely” separating solution



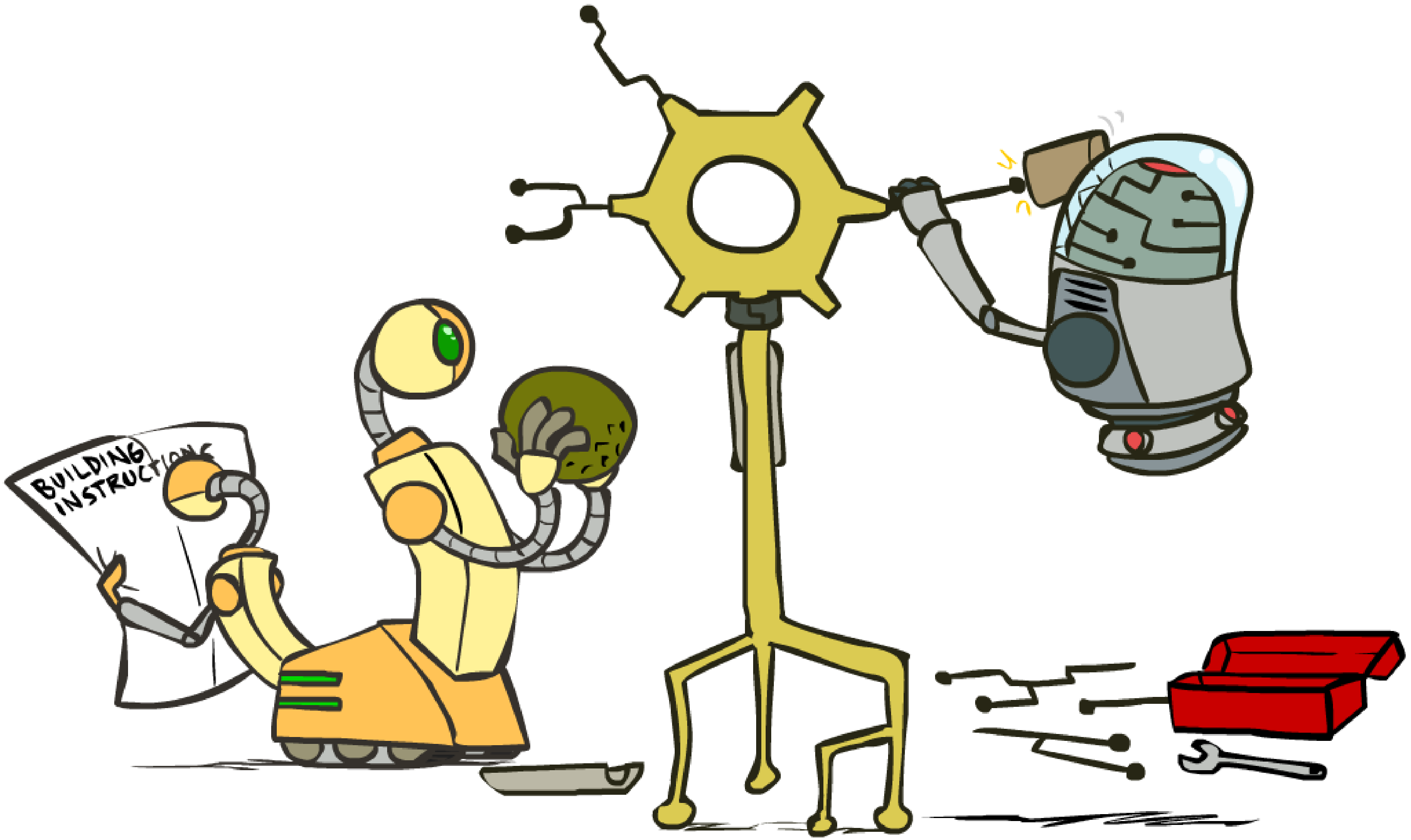


- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



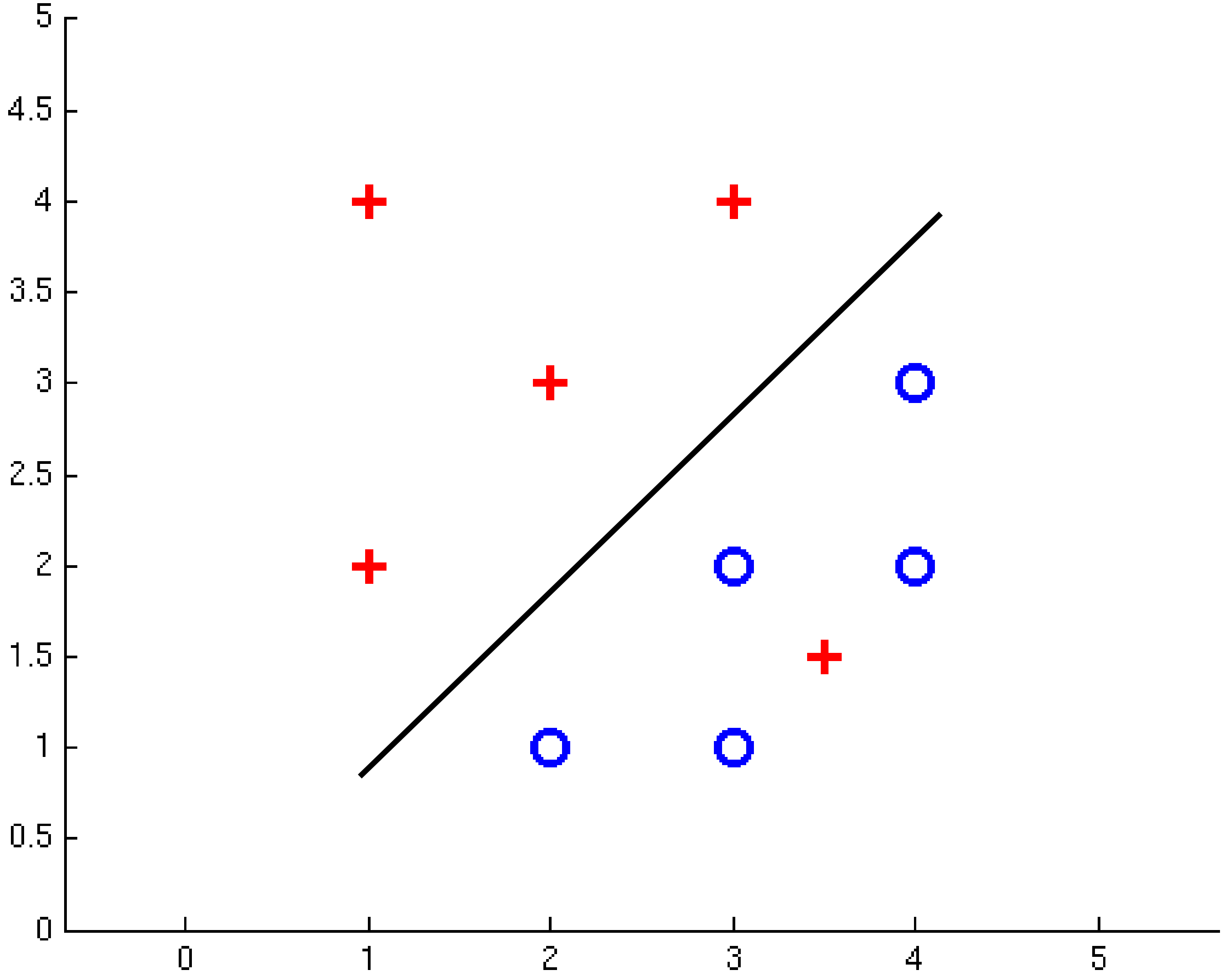


Improving the Perceptron

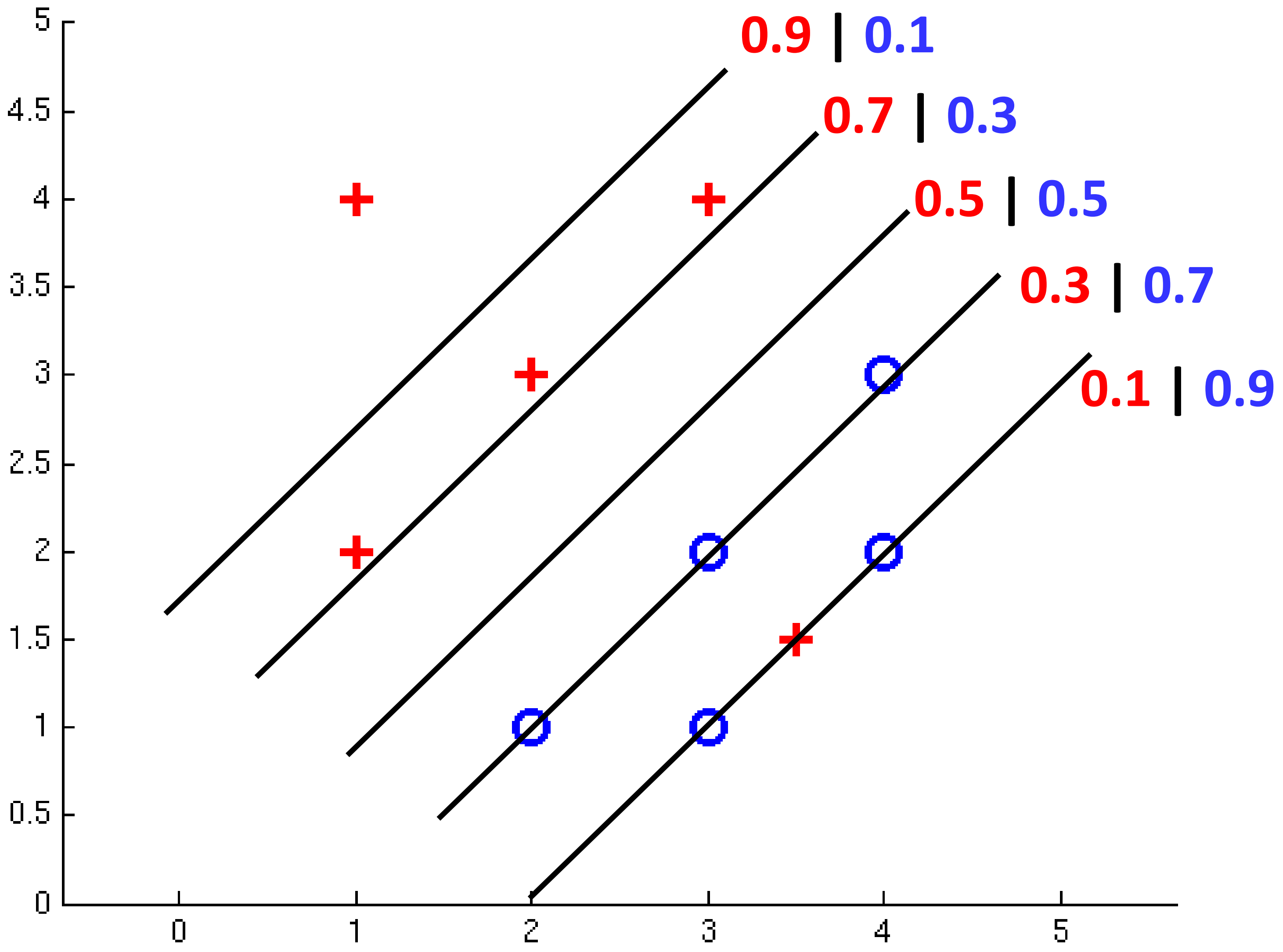


Non-Separable Case: Deterministic Decision

Even the best linear boundary makes at least one mistake



Non-Separable Case: Probabilistic Decision

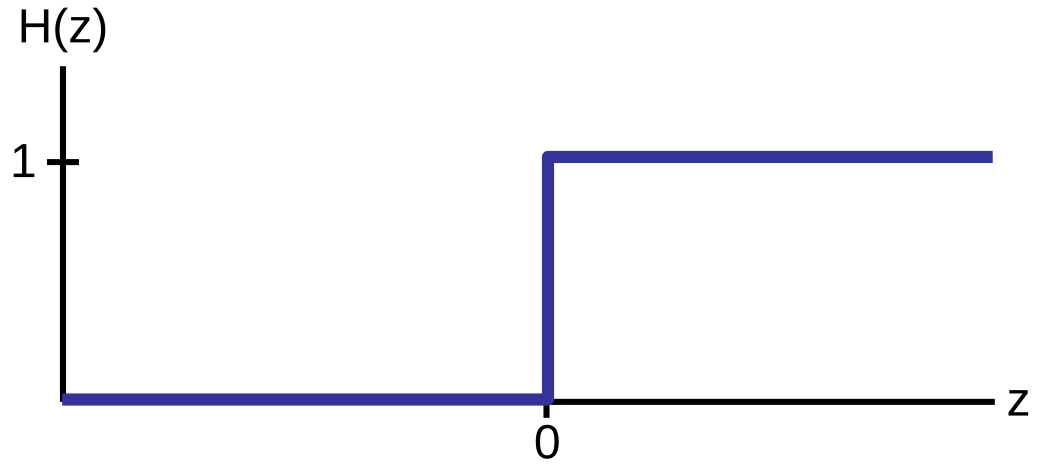


How to get deterministic decisions?

- Perceptron scoring: $z = w \cdot f(x)$
- If $z = w \cdot f(x)$ positive → classifier says: 1.0 probability this is class +1
- If $z = w \cdot f(x)$ negative → classifier says: 0.0 probability this is class +1

- Step function

$$H(z) = \begin{cases} 1 & z > 0 \\ 0 & z \leq 0 \end{cases}$$



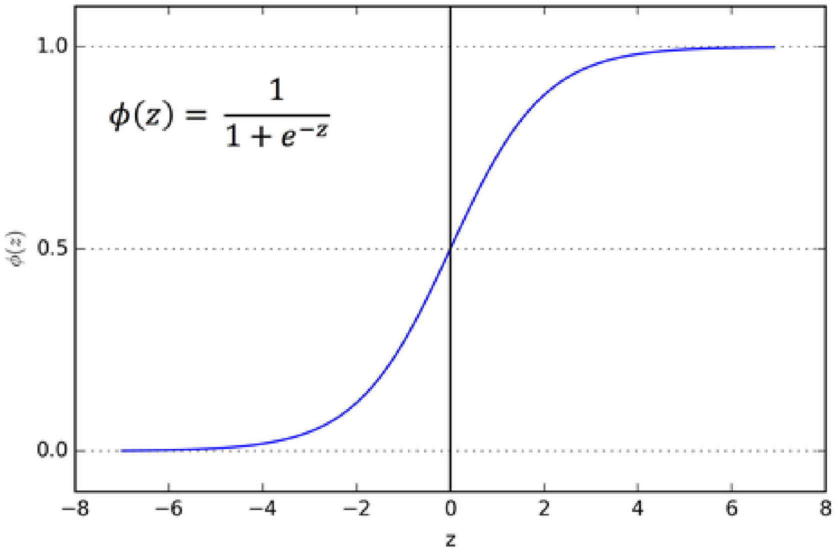
- z = output of perceptron
  H(z) = probability the class is +1, according to the classifier

How to get probabilistic decisions?

- Perceptron scoring: $z = w \cdot f(x)$
- If $z = w \cdot f(x)$ very positive → probability of class +1 should approach 1.0
- If $z = w \cdot f(x)$ very negative → probability of class +1 should approach 0.0

- Sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

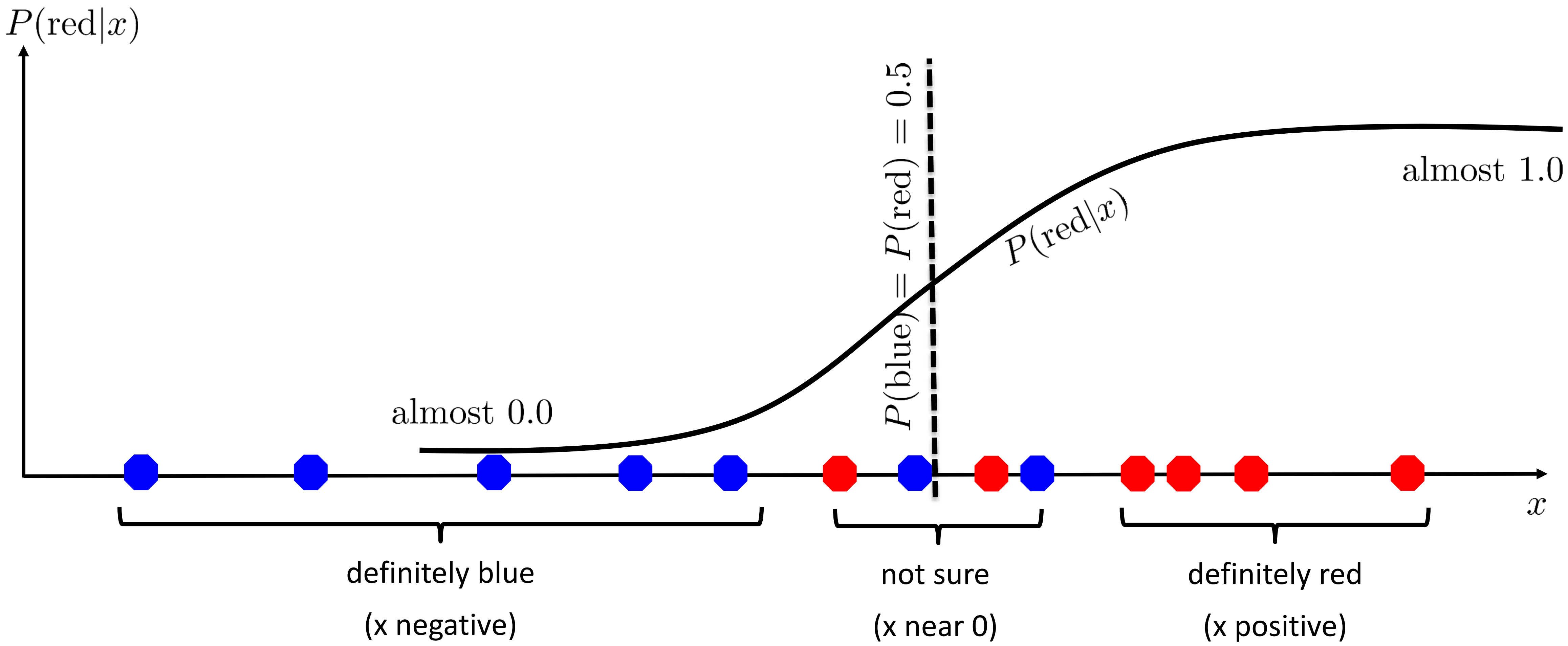


- z = output of perceptron
  $\phi(z)$ = probability the class is +1, according to the classifier

A 1D Example

$$P(\text{red}|x) = \frac{1}{1 + e^{-wx}}$$

where w is some weight constant (1D vector) we have to learn
(assume w is positive in this example)



Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$
\begin{aligned}
\text{Likelihood} &= P(\text{training data}|w) \\
&= \prod_i P(\text{training datapoint } i \mid w) \\
&= \prod_i P(\text{point } x^{(i)} \text{ has label } y^{(i)}|w) \\
&= \prod_i P(y^{(i)}|x^{(i)};w) \\
\text{Log Likelihood} &= \sum_i \log P(y^{(i)}|x^{(i)};w)
\end{aligned}
$$

Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$
\begin{aligned}
& P(\text{point } x^{(i)} \text{ has label } y^{(i)} = +1 \mid w) \\
= \; & P(y^{(i)} = +1 \mid x^{(i)}; w) \\
= \; & \frac{1}{1 + e^{-w \cdot x^{(i)}}}
\end{aligned}
$$

$$
\begin{aligned}
& P(\text{point } x^{(i)} \text{ has label } y^{(i)} = -1 \mid w) \\
= \; & P(y^{(i)} = -1 \mid x^{(i)}; w) \\
= \; & 1 - \frac{1}{1 + e^{-w \cdot x^{(i)}}}
\end{aligned}
$$

Best w?

- Maximum likelihood estimation:

$$\max_{w} \; ll(w) = \max_{w} \; \sum_{i} \log P(y^{(i)}|x^{(i)};w)$$

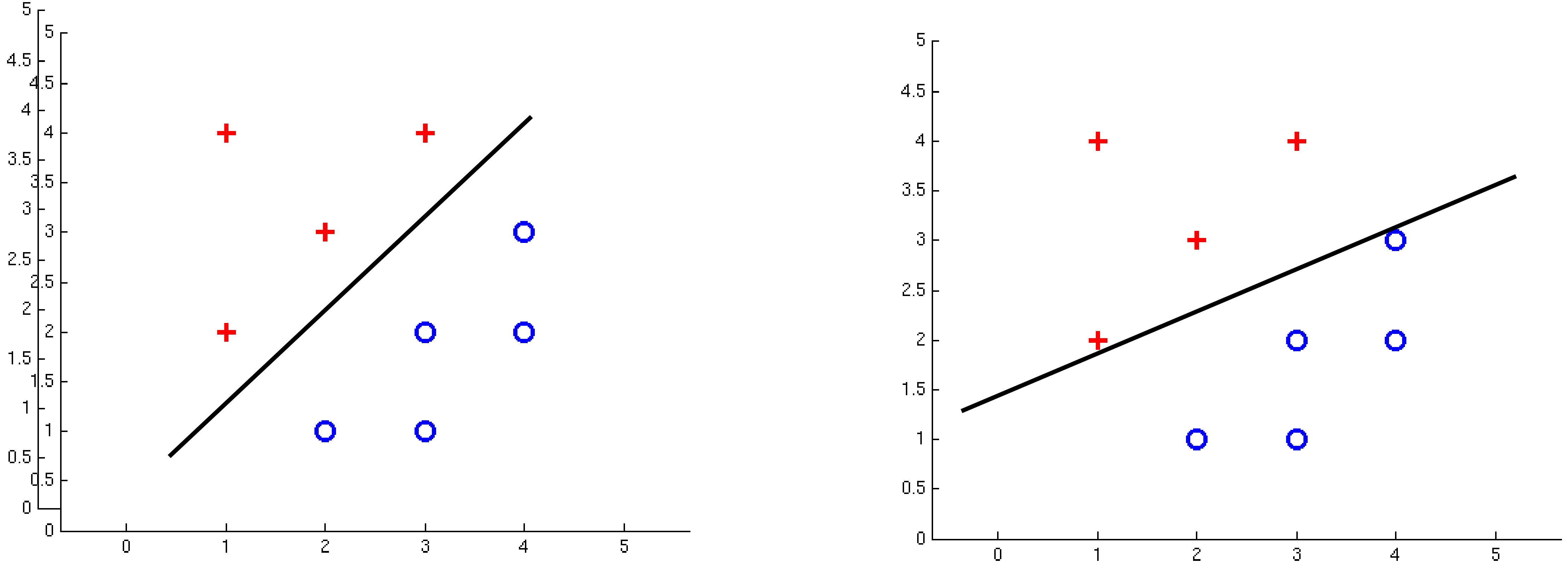
with:

$$P(y^{(i)} = +1|x^{(i)};w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

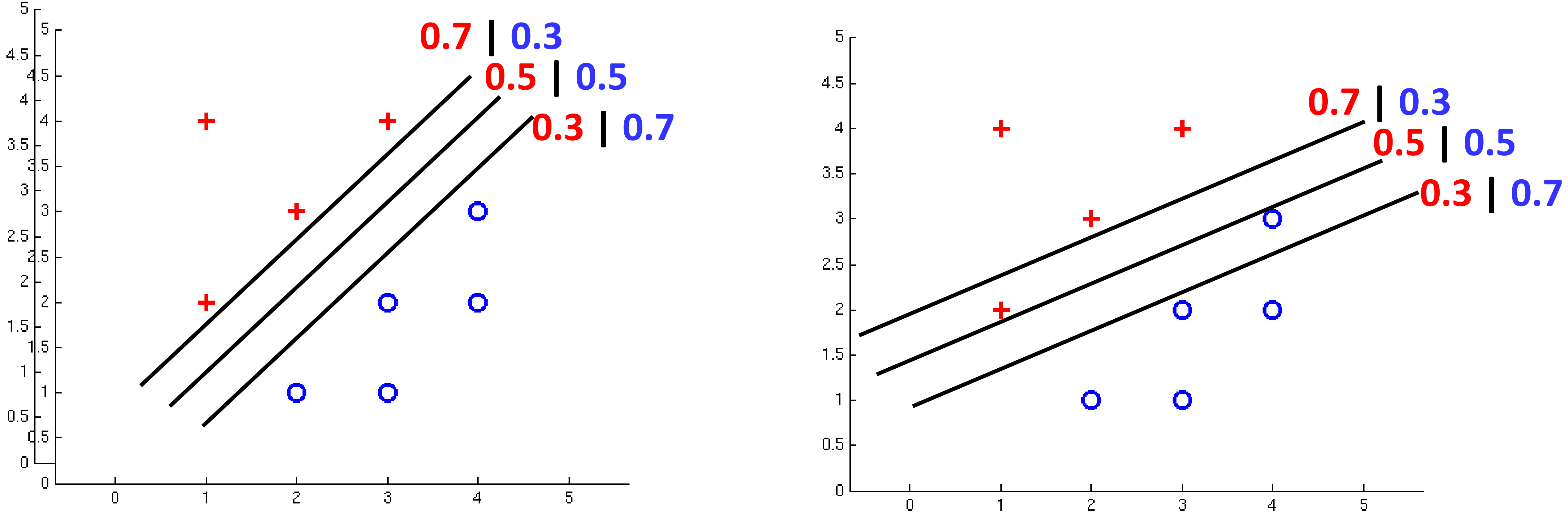
$$P(y^{(i)} = -1|x^{(i)};w) = 1 - \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

= Logistic Regression

Separable Case: Deterministic Decision – Many Options

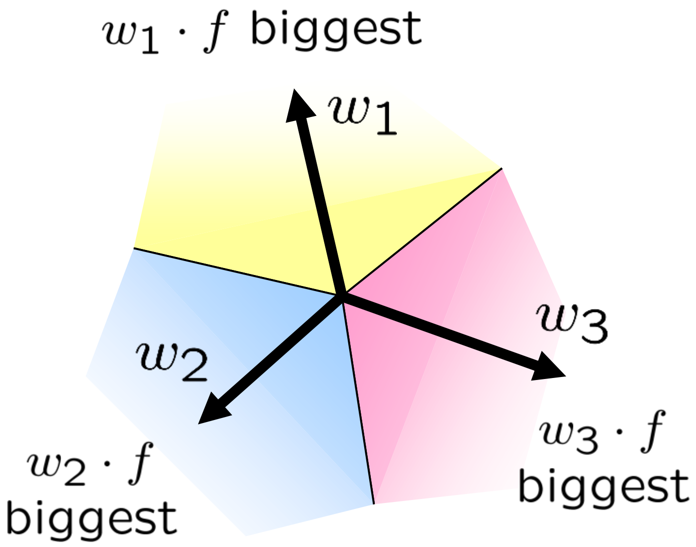


Separable Case: Probabilistic Decision – Clear Preference



Multiclass Logistic Regression

- Recall Perceptron:
  - A weight vector for each class: w_y
  - Score (activation) of a class y: $w_y \cdot f(x)$
  - Prediction highest score wins $y = \arg\max_y \; w_y \cdot f(x)$



- How to make the scores into probabilities?

$$z_1, z_2, z_3 \rightarrow \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

original activations — softmax activations

Best w?

- Recall maximum likelihood estimation: Choose the w value that maximizes the probability of the observed (training) data

$$
\begin{aligned}
\text{Likelihood} &= P(\text{training data}|w) \\
&= \prod_i P(\text{training datapoint } i \mid w) \\
&= \prod_i P(\text{point } x^{(i)} \text{ has label } y^{(i)}|w) \\
&= \prod_i P(y^{(i)}|x^{(i)};w) \\
\text{Log Likelihood} &= \sum_i \log P(y^{(i)}|x^{(i)};w)
\end{aligned}
$$

Best w?

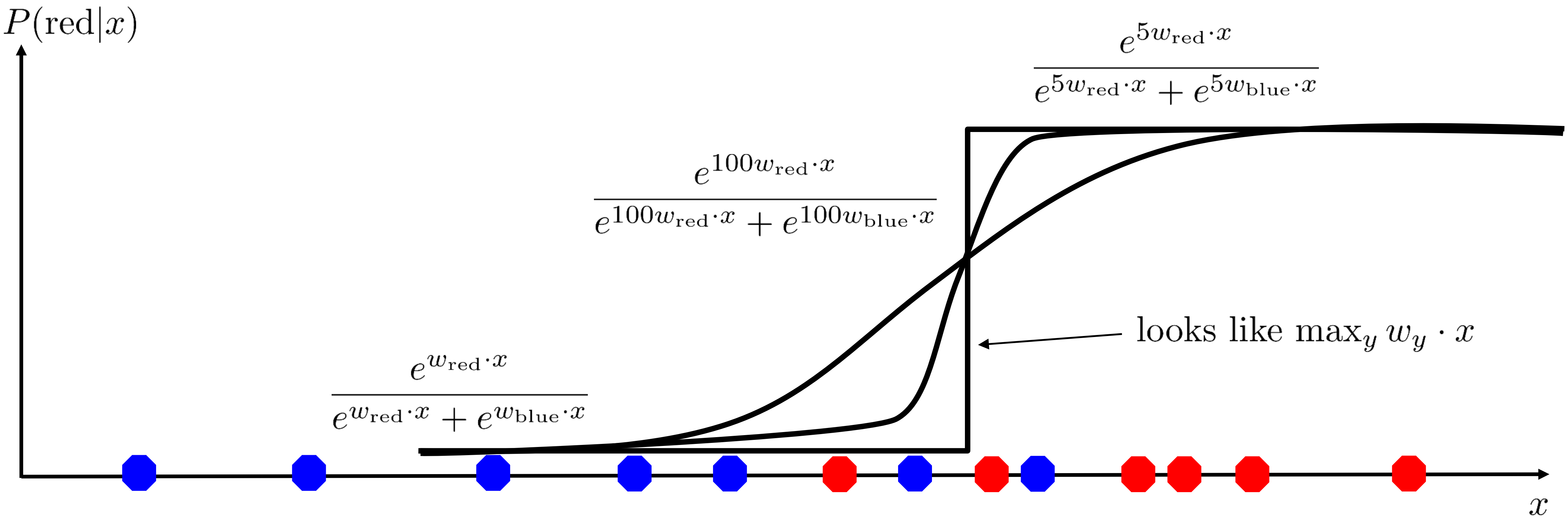
- Maximum likelihood estimation:

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

with: $$P(y^{(i)}|x^{(i)}; w) = \frac{e^{w_{y^{(i)}} \cdot f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}$$

= Multi-Class Logistic Regression

Softmax with Different Bases



$$P(\text{red}|x) = \frac{e^{w_{\text{red}}\cdot x}}{e^{w_{\text{red}}\cdot x}+e^{w_{\text{blue}}\cdot x}}$$

Softmax and Sigmoid

- Recall: Binary perceptron is a special case of multi-class perceptron
  - Multi-class: Compute $w_y \cdot f(x)$ for each class y, pick class with the highest activation
  - Binary case:
    Let the weight vector of +1 be w (which we learn).
    Let the weight vector of -1 always be 0 (constant).
  - Binary classification as a multi-class problem:
    Activation of negative class is always 0.
    If w · f is positive, then activation of +1 (w · f) is higher than -1 (0).
    If w · f is negative, then activation of -1 (0) is higher than +1 (w · f).

Softmax

$$P(\mathrm{red}|x) = \frac{e^{w_{\mathrm{red}} \cdot x}}{e^{w_{\mathrm{red}} \cdot x} + e^{w_{\mathrm{blue}} \cdot x}}$$

with $w_{red} = 0$ becomes:

Sigmoid

$$P(\mathrm{red}|x) = \frac{1}{1 + e^{-wx}}$$

Next Lecture

- Optimization
  - i.e., how do we solve:

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$