# CS 188: Artificial Intelligence

## Machine Learning II: Perceptrons


Summer 2024: Eve Fleisig & Evgeny Pobachienko

[Slides based on those by Nicholas Tomlin, Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. CS188 materials are available at http://ai.berkeley.edu.]

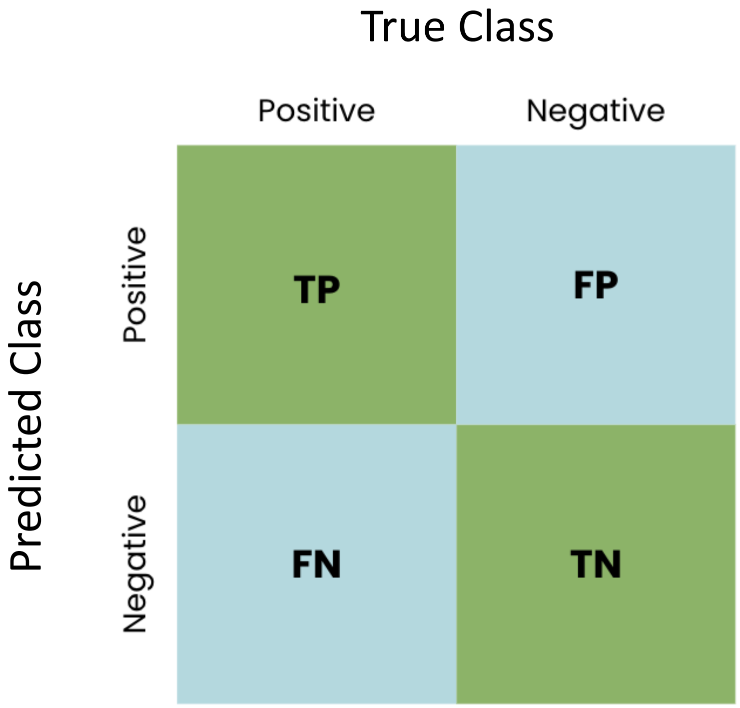
# Demo: Catching AI-Generated Text

- Feature design
  - Complexity in feature design vs. model design
- Evaluation: accuracy, precision & recall, F1 score
- Generalization, calibration, robustness

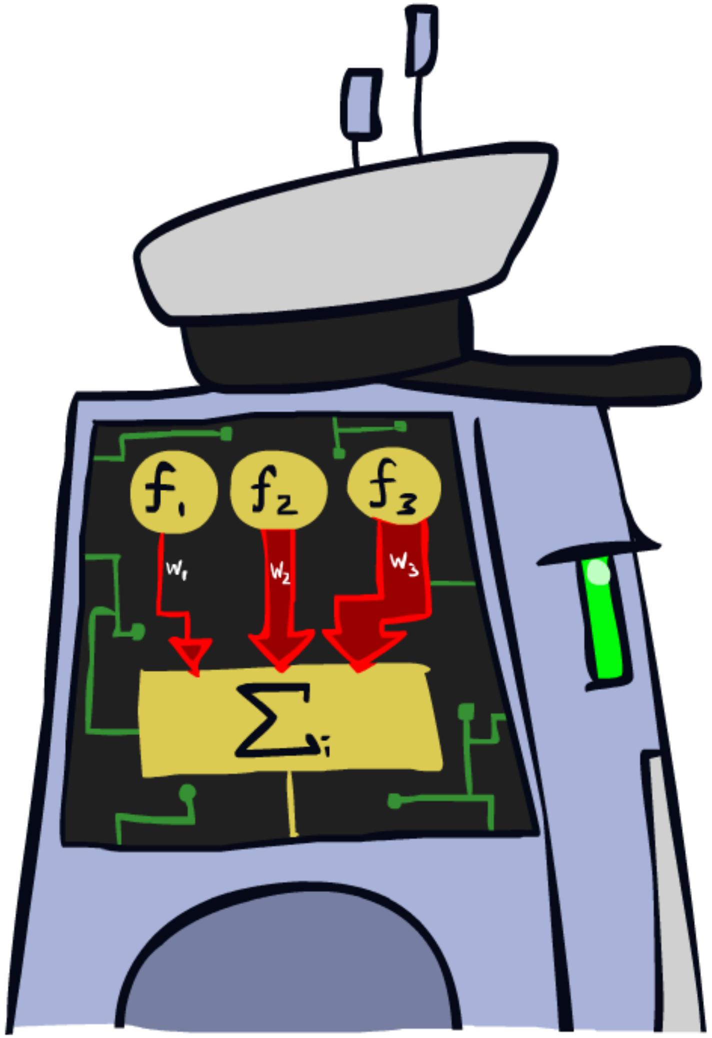
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$



# 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
...
```

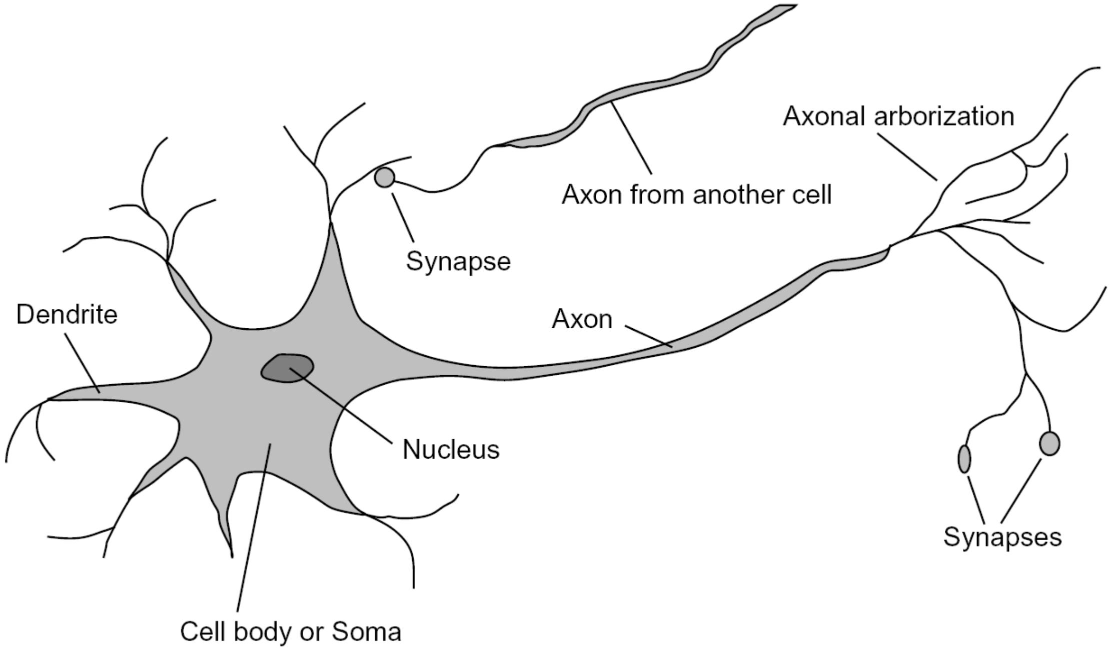
SPAM
or
+

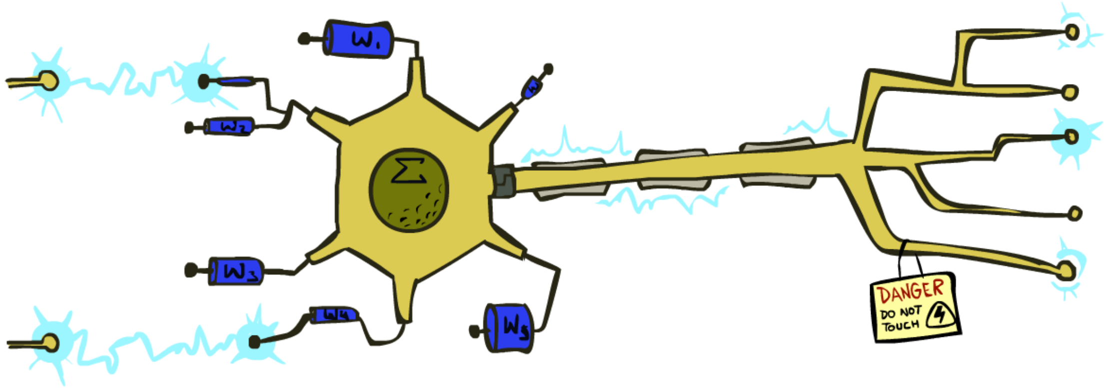
```
PIXEL-7,12  : 1
PIXEL-7,13  : 0
...
NUM_LOOPS   : 1
...
```

“2”

# Some (Simplified) Biology

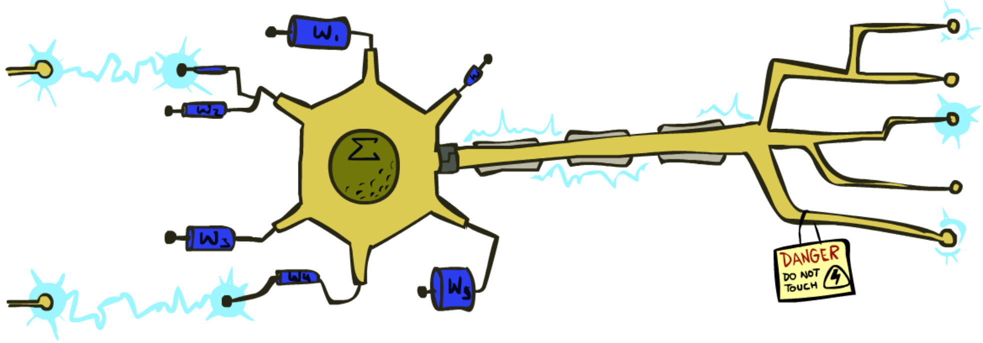
- Very loose inspiration: human neurons



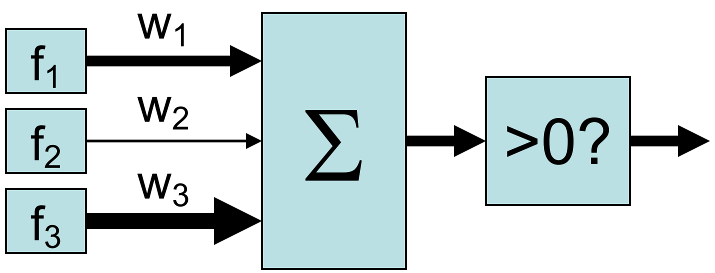
# Linear Classifiers

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




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

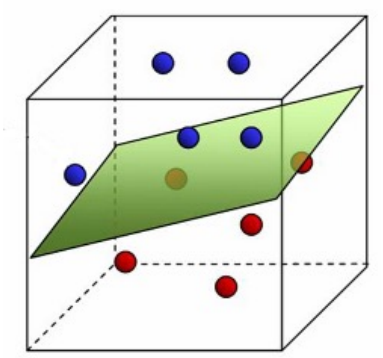
- If the activation is:
  - Positive, output +1
  - Negative, output -1

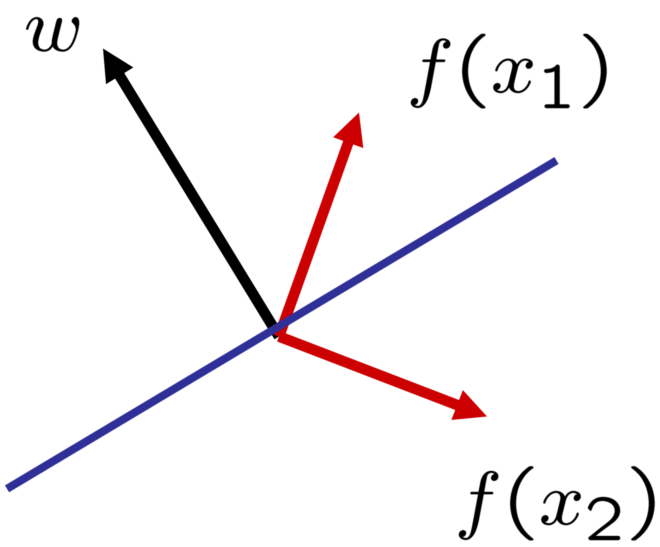


# Weights

- 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
...
```



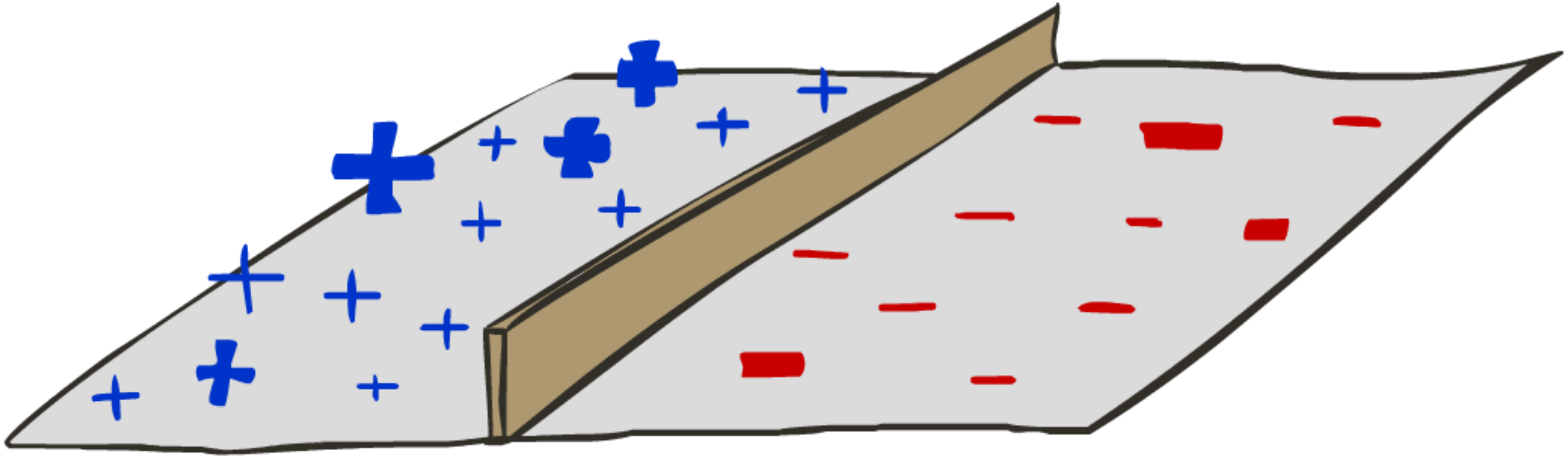



```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

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

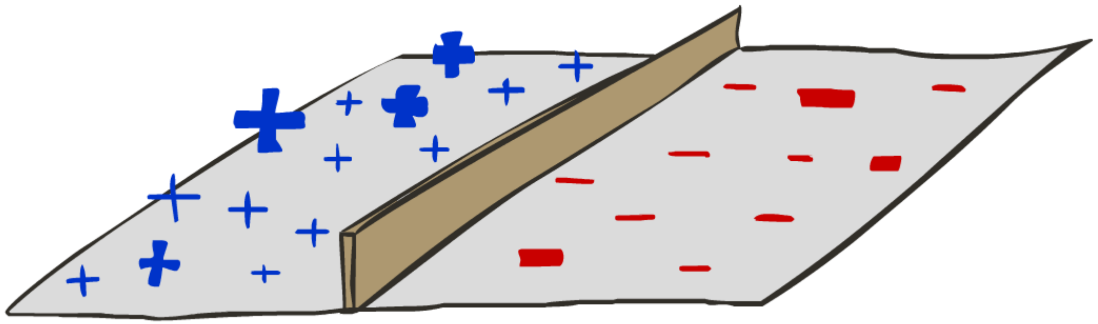
*Dot product* $w \cdot f$ *positive means the positive class*

# Decision Rules



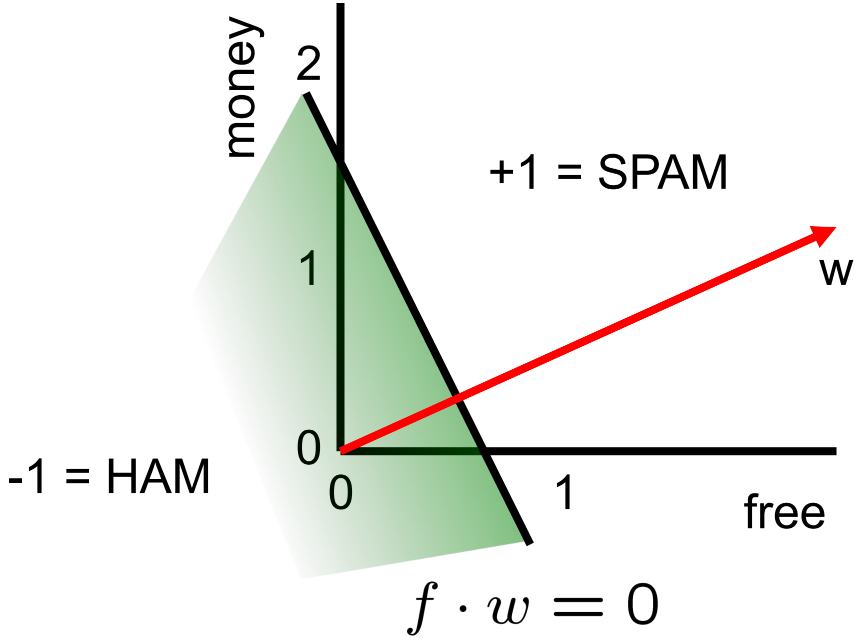
# Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector defines a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



$$w$$

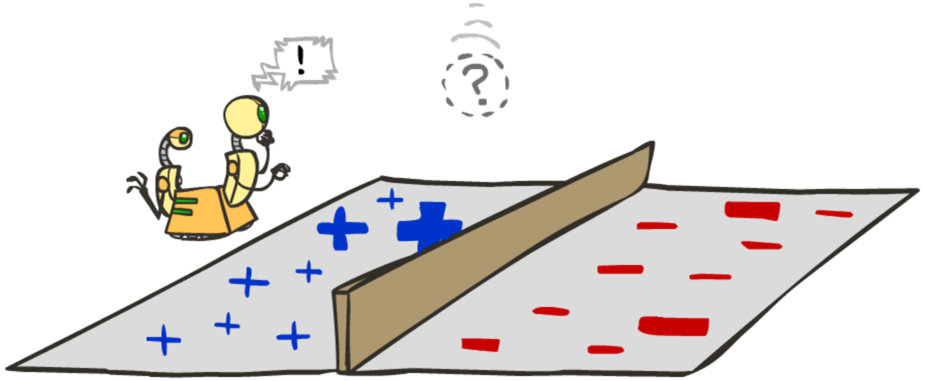
```
BIAS  : -3
free  :  4
money :  2
...
```



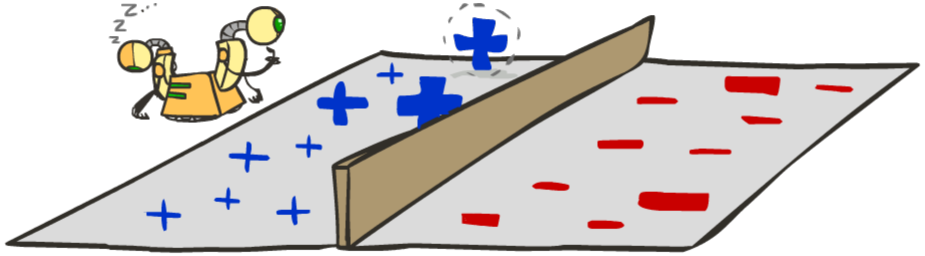
# Weight Updates

# Learning: Binary Perceptron

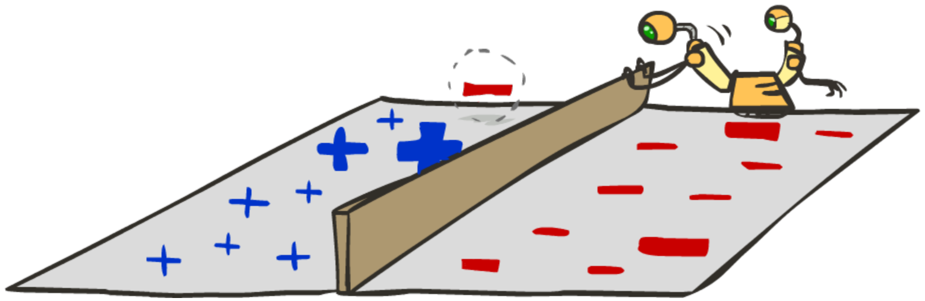
- Start with weights = 0
- For each training instance:
  - Classify with current weights




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



  - If wrong: adjust the weight vector



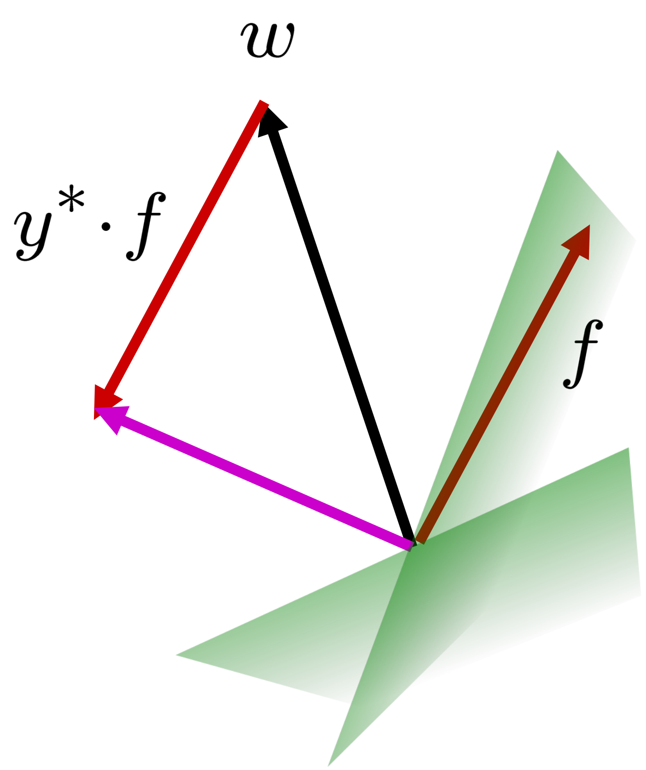
# Learning: Binary Perceptron

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

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

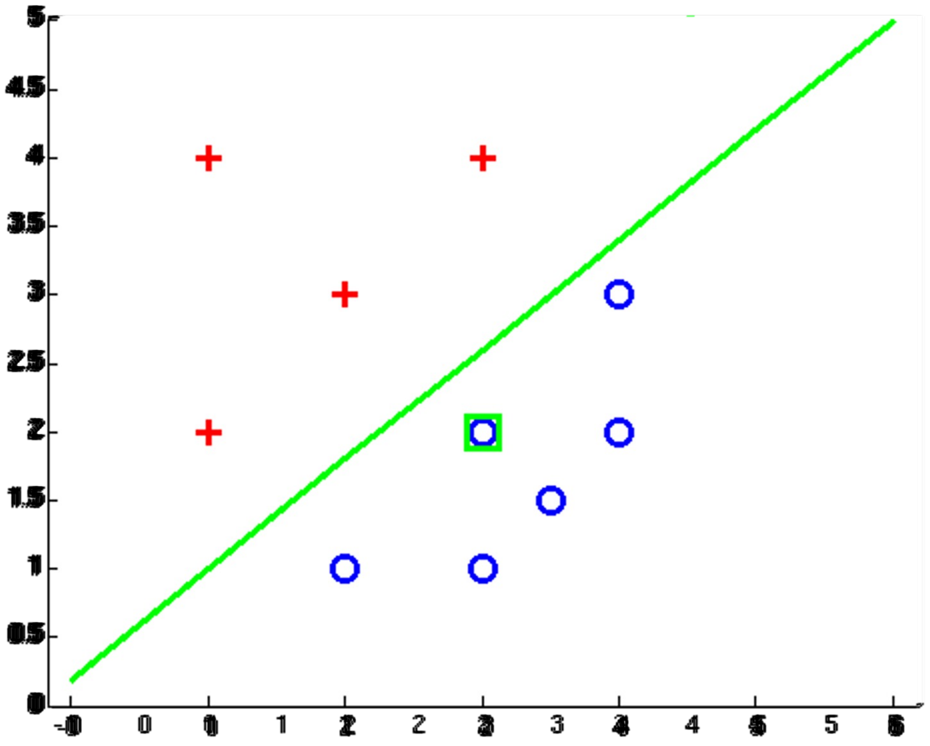
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

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



# Examples: Perceptron

- Separable Case



# Multiclass Decision Rule

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

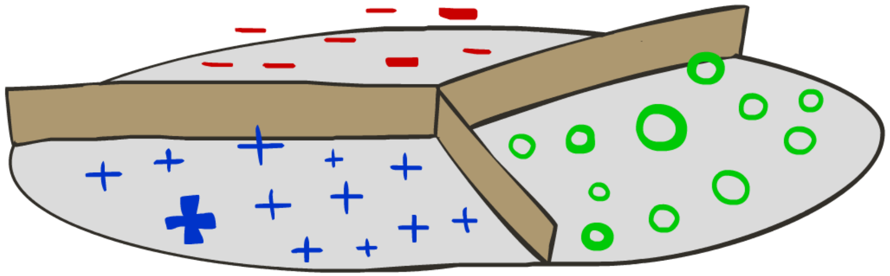
    $$w_y$$

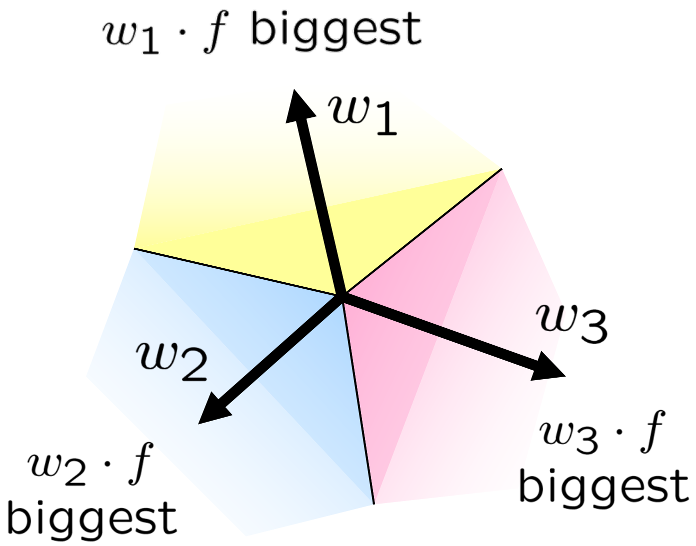
  - Score (activation) of a class y:

    $$w_y \cdot f(x)$$

  - 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

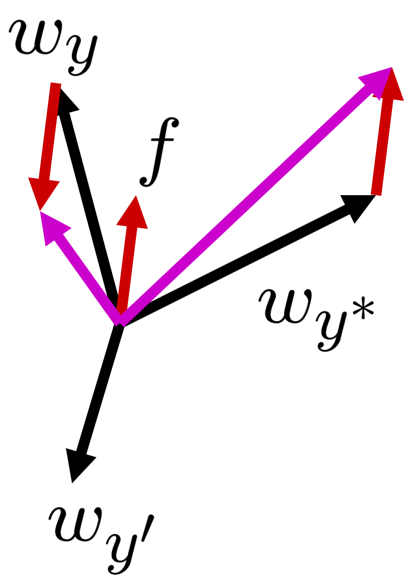
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

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

- 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
...
```

$w_{TECH}$

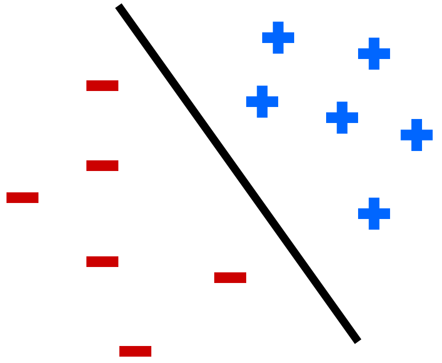
```
BIAS  : 0
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

# Properties of Perceptrons

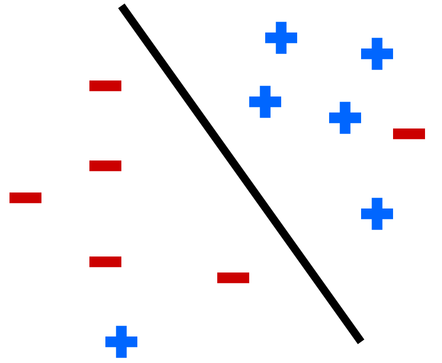
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

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

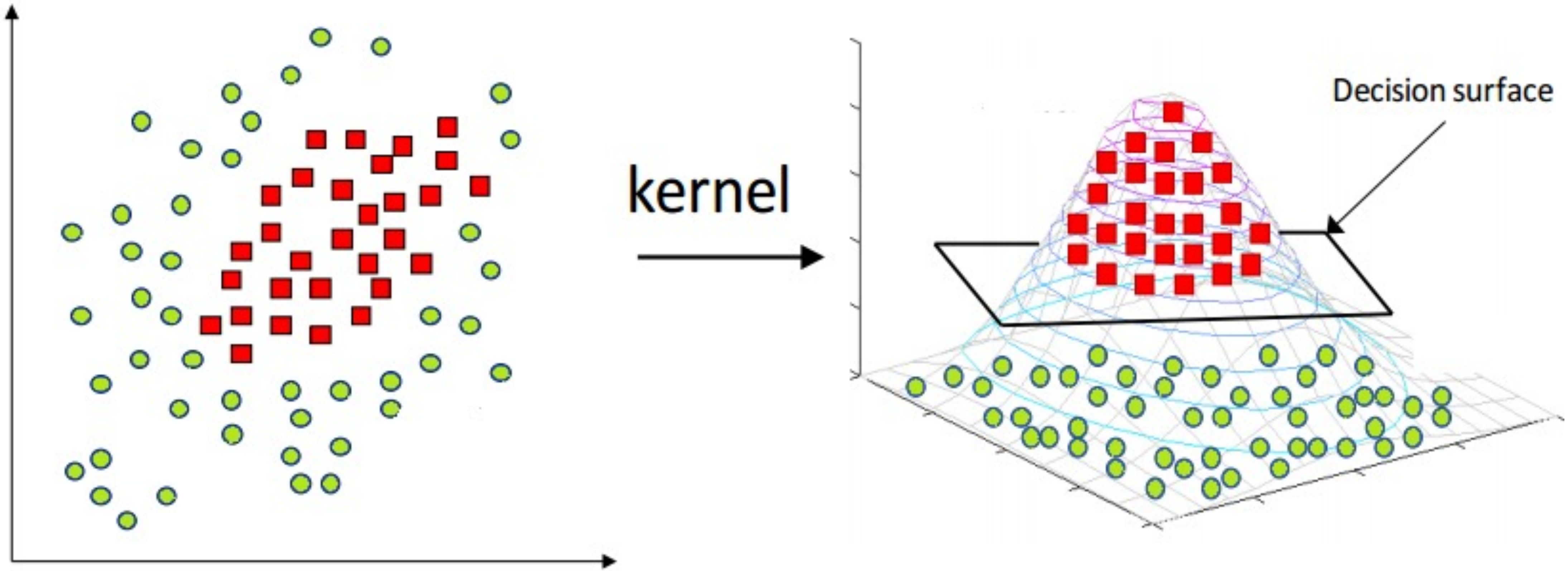
Separable



Non-Separable

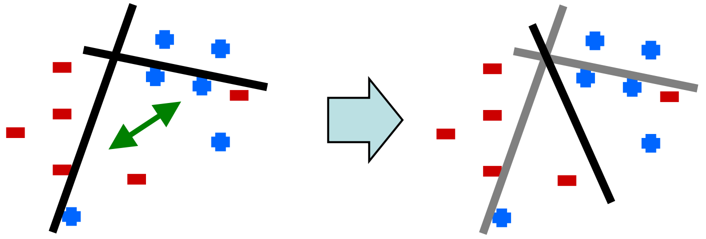


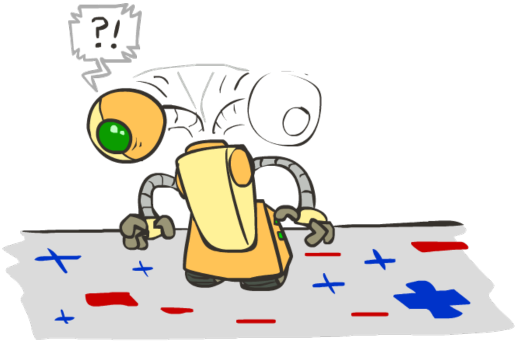
# [Bonus] Kernel Trick



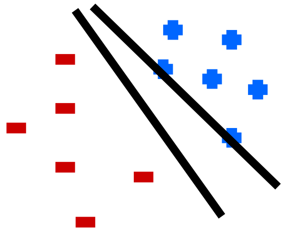
# Problems with the Perceptron

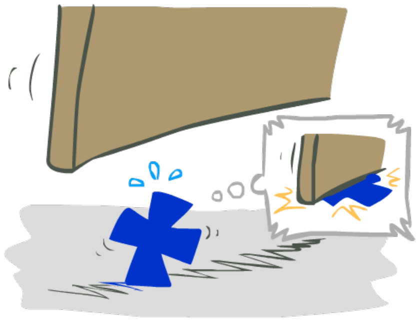
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)






- Mediocre generalization: finds a "barely" separating solution





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