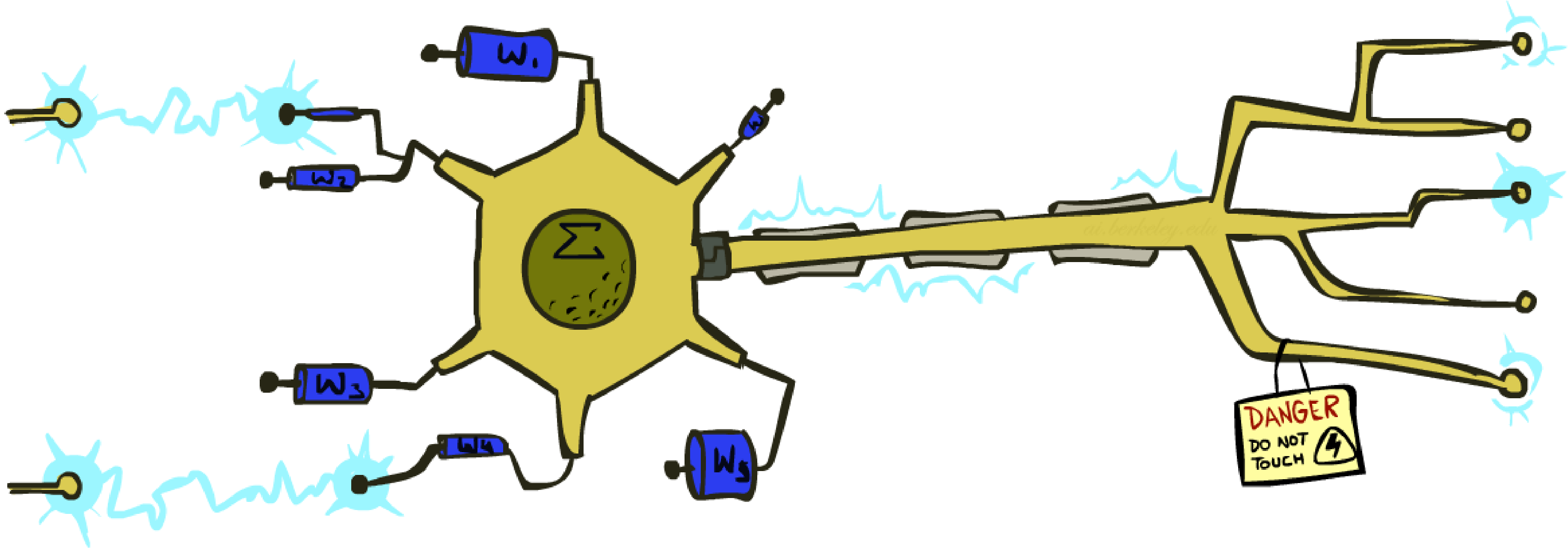
Machine Learning
Perceptrons



These slides are based on the slides created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley - http://ai.berkeley.edu.
The artwork is by Ketrina Yim.

Announcement

Office hours:

- Wednesday May 4 at 3 PM instead of 12 PM.

Where we are

- Last week:
  - Naïve Bayes Classification (AIMA 20.1 20.2)
- Today:
  - Perceptrons (Artificial Neural Networks AIMA 18.7)
- Homework 9: Programming Assignment
  - Perceptrons
  - Apprenticeship

Error-Driven Classification

Errors, and What to Do

- Examples of errors

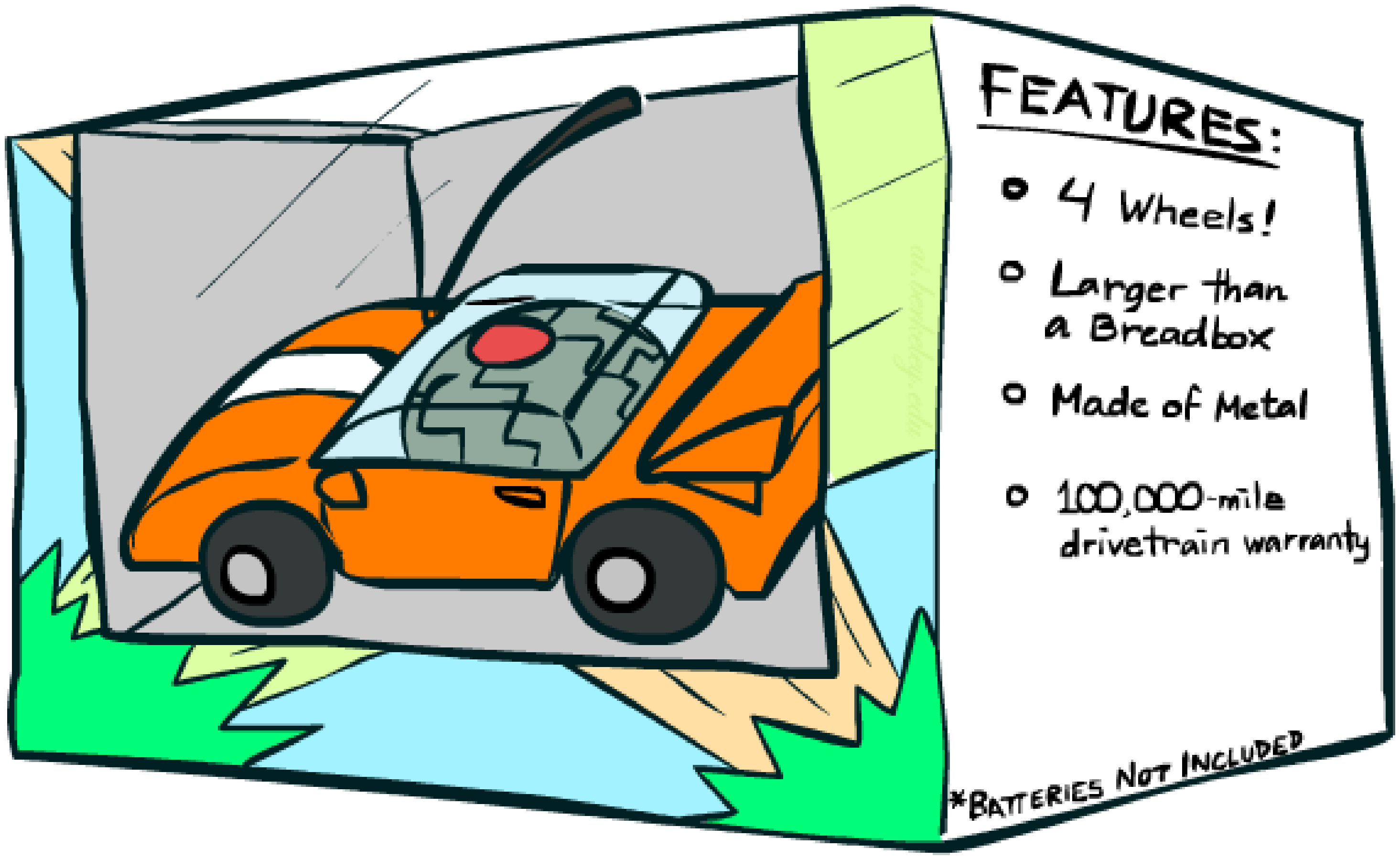
```
Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the
latest version of OmniPage Pro, for just $99.99* - the regular
list price is $499! The most common question we've received
about this offer is - Is this genuine? We would like to assure
you that this offer is authorized by ScanSoft, is genuine and
valid. You can get the . . .
```

```
. . . To receive your $30 Amazon.com promotional certificate,
click through to

  http://www.amazon.com/apparel

and see the prominent link for the $30 offer. All details are
there. We hope you enjoyed receiving this message. However, if
you'd rather not receive future e-mails announcing new store
launches, please click . . .
```

What to Do About Errors

- Problem: there's still spam in our inbox
- Need more features – words aren't enough!
  - Have we emailed the sender before?
  - Have 1M other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?
- Naïve Bayes models can incorporate a variety of features, but tend to do best in homogeneous cases (e.g. all features are word occurrences)

Features


FEATURES:
4 Wheels!
Larger than a Breadbox
Made of Metal
100,000-mile drivetrain warranty
*BATTERIES NOT INCLUDED

Feature Vectors

- Data is represented as a feature vector

x

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

$f(x)$

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

Feature Vectors

- Data is represented as a feature vector

x

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

$f(x)$

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

Feature Vectors

- Data is represented as a feature vector

x

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

$f(x)$

$$\begin{pmatrix} \texttt{\# free} & : & 2 \\ \texttt{YOUR\_NAME} & : & 0 \\ \texttt{MISSPELLED} & : & \\ \texttt{FROM\_FRIEND} & : & \\ \ldots & & \end{pmatrix}$$

Feature Vectors

- Data is represented as a feature vector

x

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

$f(x)$

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

Feature Vectors

- Data is represented as a feature vector

x

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

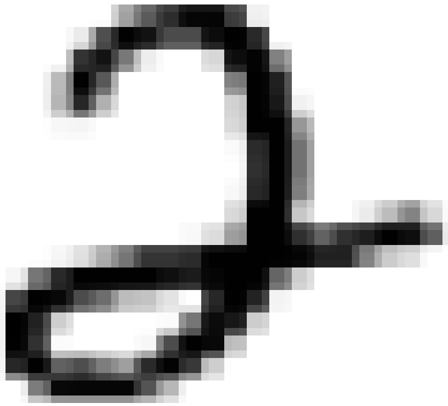
$f(x)$

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

Feature Vectors

- Data is represented as a feature vector

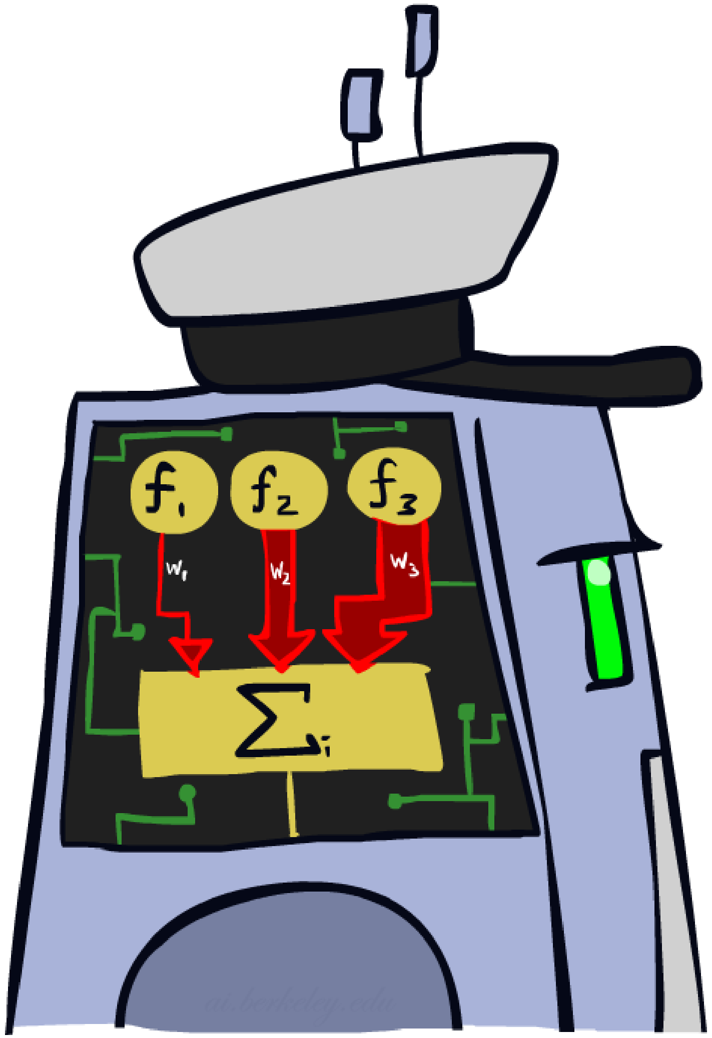
x



$f(x)$

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

Linear Classifiers



Linear Classifiers

x

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

$f(x)$

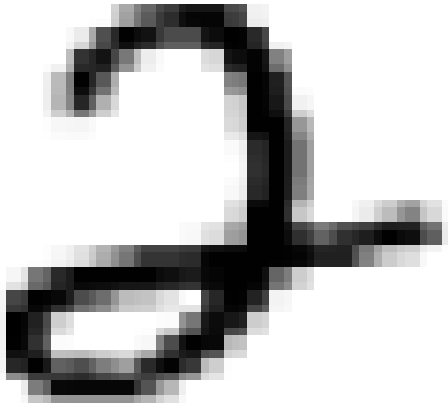
```
# free         : 2
YOUR_NAME      : 0
MISSPELLED     : 2
FROM_FRIEND    : 0
...
```

y

SPAM
or
HAM

Linear Classifiers

x $\qquad$ $f(x)$ $\qquad$ y

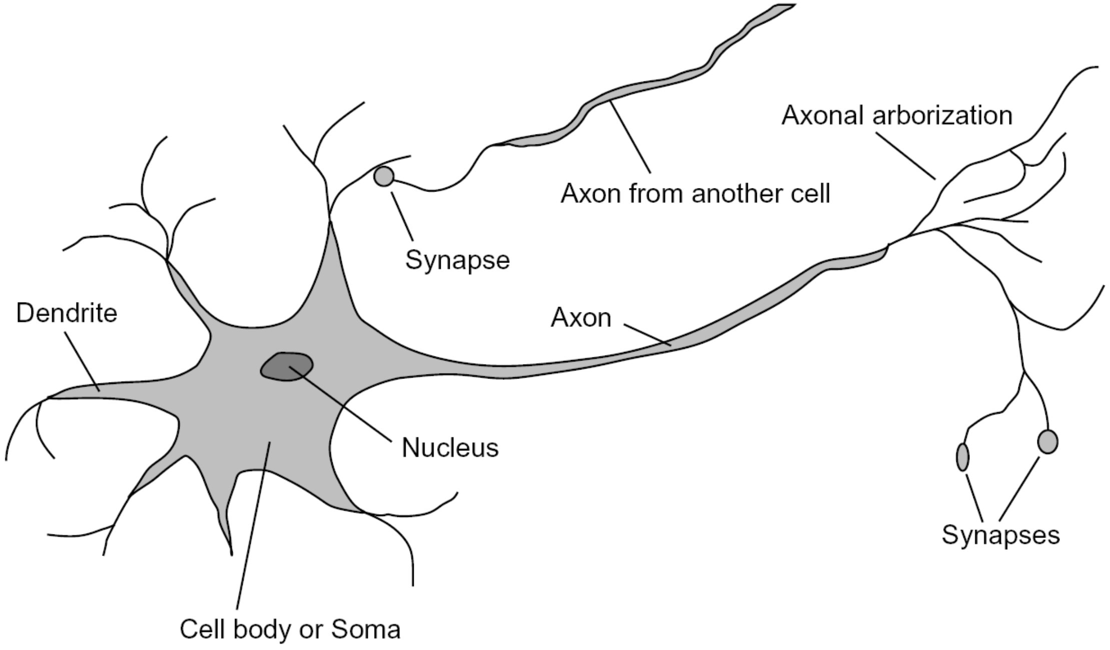


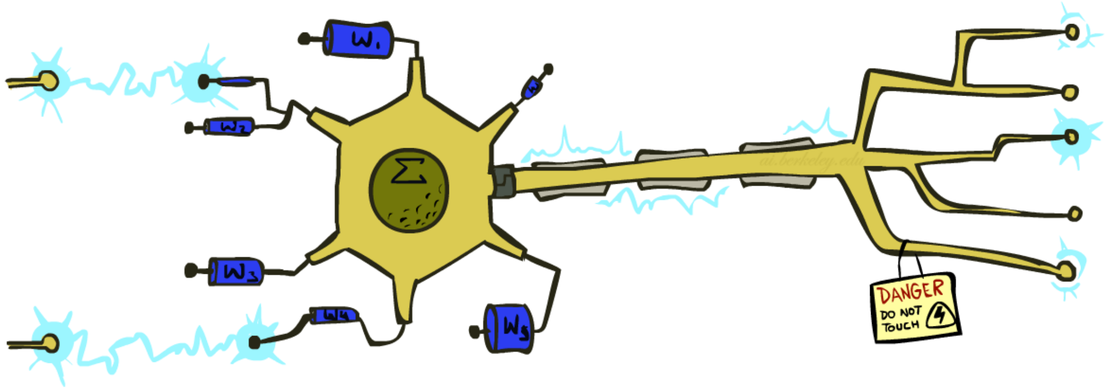
```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

"2"

Some (Simplified) Biology

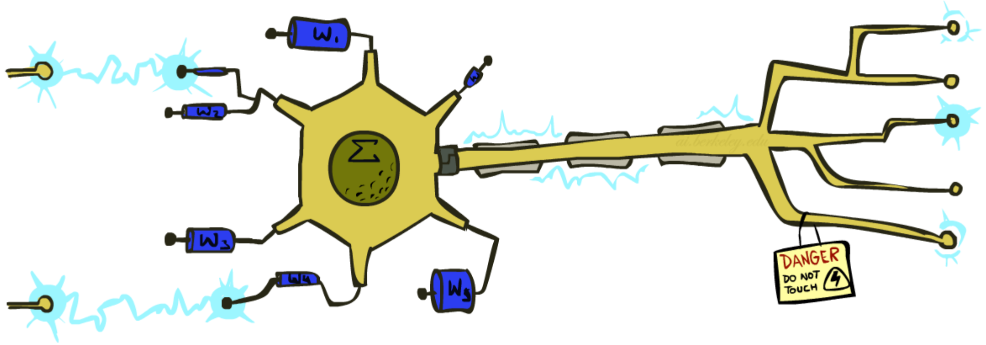
- Very loose inspiration: human neurons





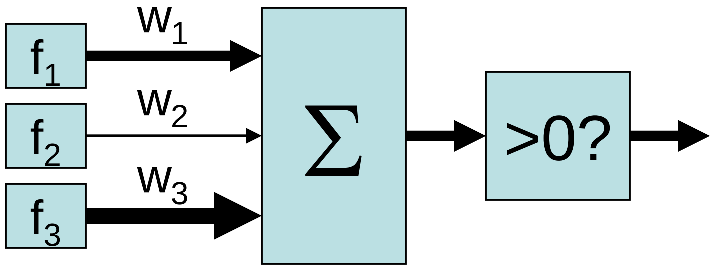
Linear Classifiers

- Inputs are feature values (vectors)
- 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

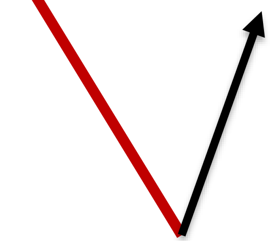


Weights

- Binary case: compare features to a weight vector

$$\begin{pmatrix} \texttt{\# free} & : & 4 \\ \texttt{YOUR\_NAME} & : & -1 \\ \texttt{MISSPELLED} & : & 1 \\ \texttt{FROM\_FRIEND} & : & -3 \end{pmatrix}$$

w



$$f(x_1) \begin{pmatrix} \texttt{\# free} & : & 2 \\ \texttt{YOUR\_NAME} & : & 0 \\ \texttt{MISSPELLED} & : & 2 \\ \texttt{FROM\_FRIEND} & : & 0 \end{pmatrix}$$

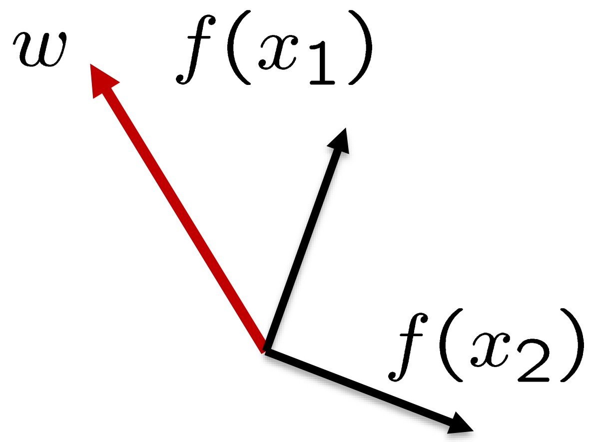
Dot product: $w \,.\, f(x1) = 4 \times 2 - 1 \times 0 + 1 \times 2 - 3 \times 0 = 10$

Dot product $w \,.\, f$ positive means the positive class (SPAM)

Weights

- Binary case: compare features to a weight vector

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



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

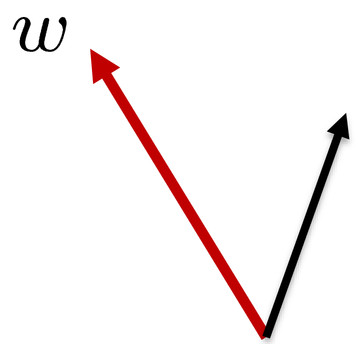
Dot product: $w \,.\, f(x2) = 4 \times 0 - 1 \times 1 + 1 \times 1 - 3 \times 1 = -3$

Dot product $w \,.\, f$ negative means the negative class (HAM)

Bias

- The bias shifts the decision boundary away from the origin and does not depend on any input value.

$$\begin{pmatrix} \texttt{BIAS} \\ \texttt{w1} \\ \texttt{w2} \\ \texttt{w3} \\ \\ \end{pmatrix}$$

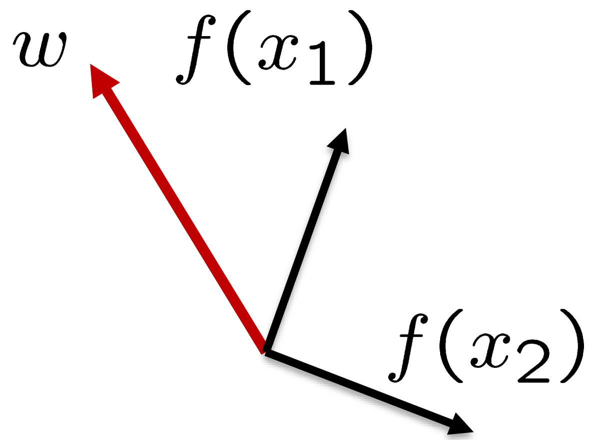


$$f(x_1) \begin{pmatrix} \texttt{F}_0\texttt{:} & \texttt{1} \\ \texttt{F}_1\texttt{:} & \texttt{..} \\ \texttt{F}_2\texttt{:} & \texttt{..} \\ \texttt{F}_3\texttt{:} & \texttt{..} \end{pmatrix}$$

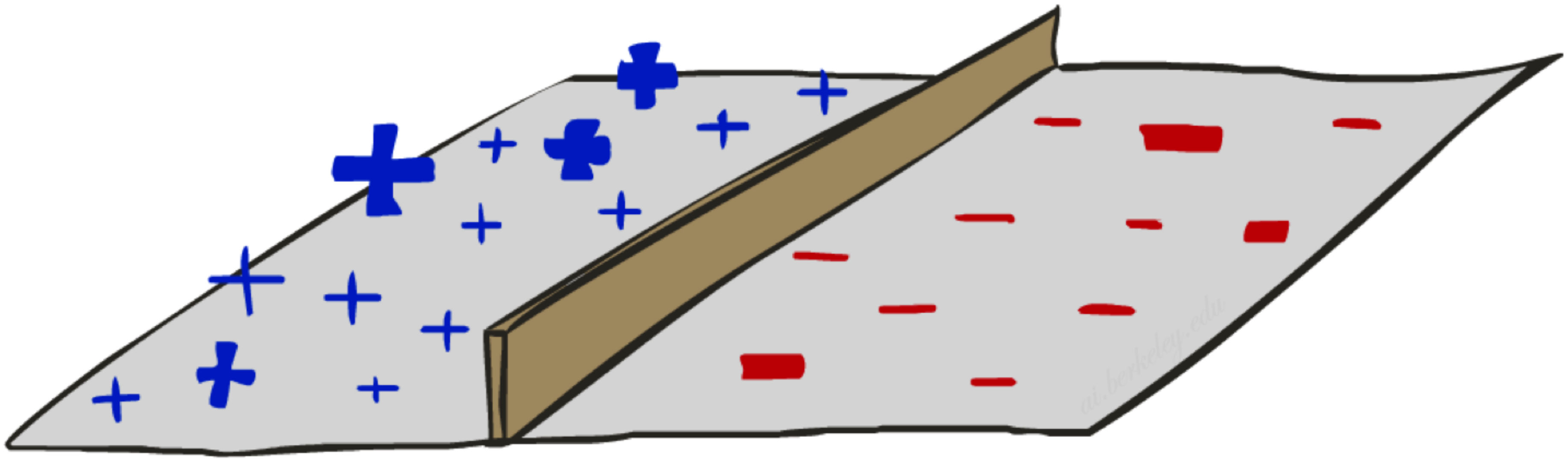
Weights

- Learning: figure out the weight vector from examples

$$\begin{bmatrix} \texttt{BIAS} \\ \texttt{w1} \\ \texttt{w2} \\ \texttt{w3} \\ \texttt{w4} \end{bmatrix}$$

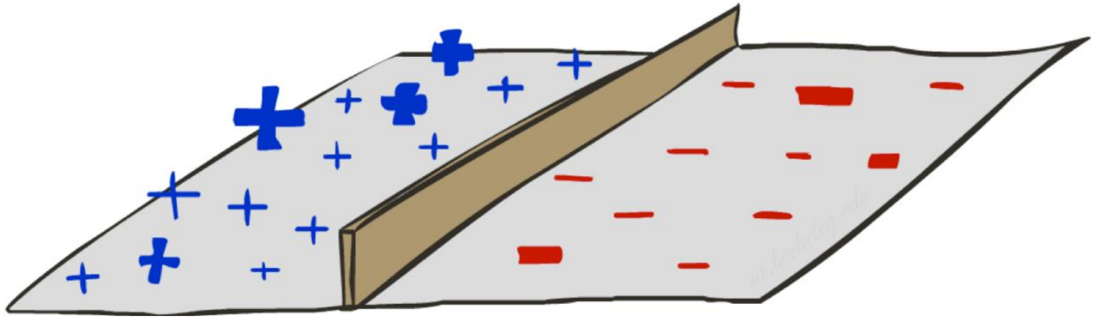


Decision Rules



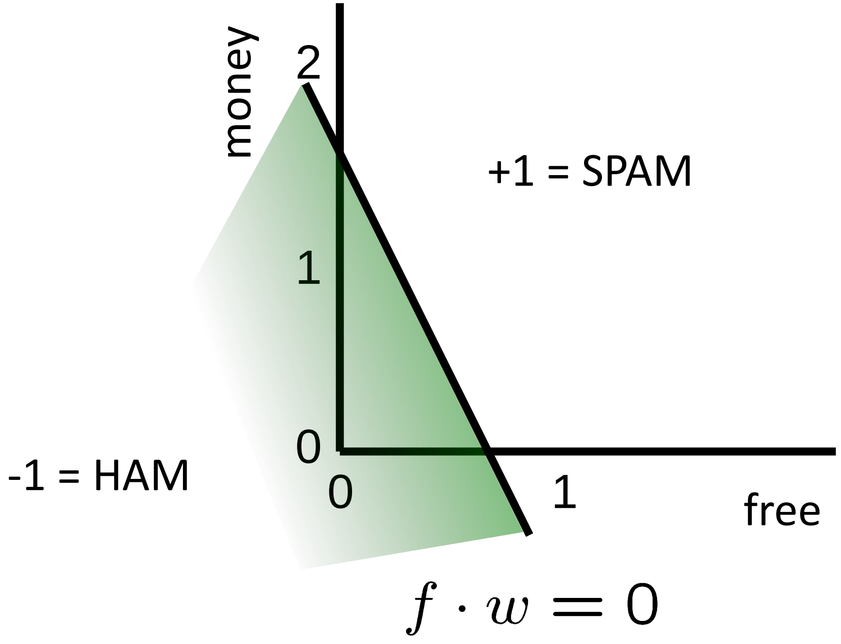
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - A weight vector corresponds to a decision boundary that is a hyperplane (a line in 2D, a plane in 3D)
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



w

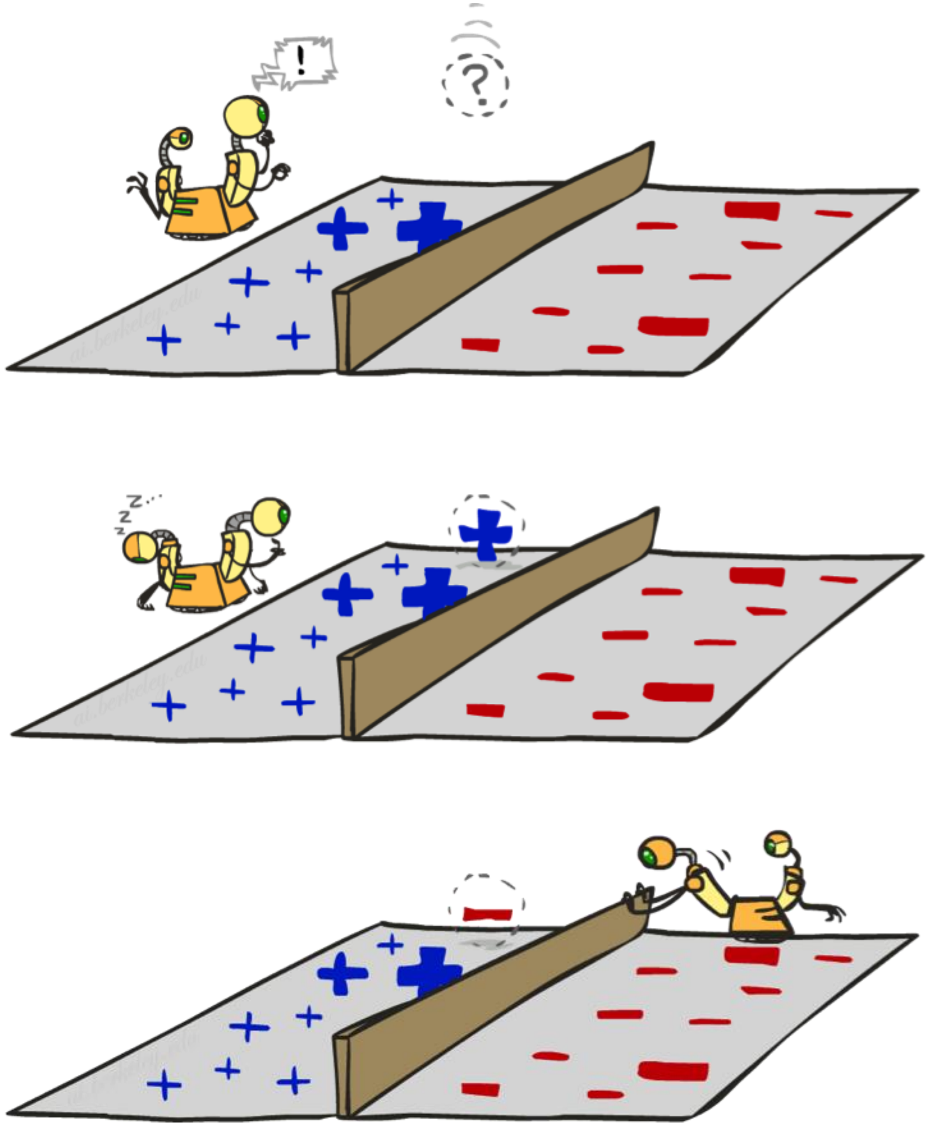
BIAS	:	-3
free	:	4
money	:	2



Weight Updates

Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights
  - y: guessed label
  - y*: training label
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector
- Repeat until no more significant changes (iterations)



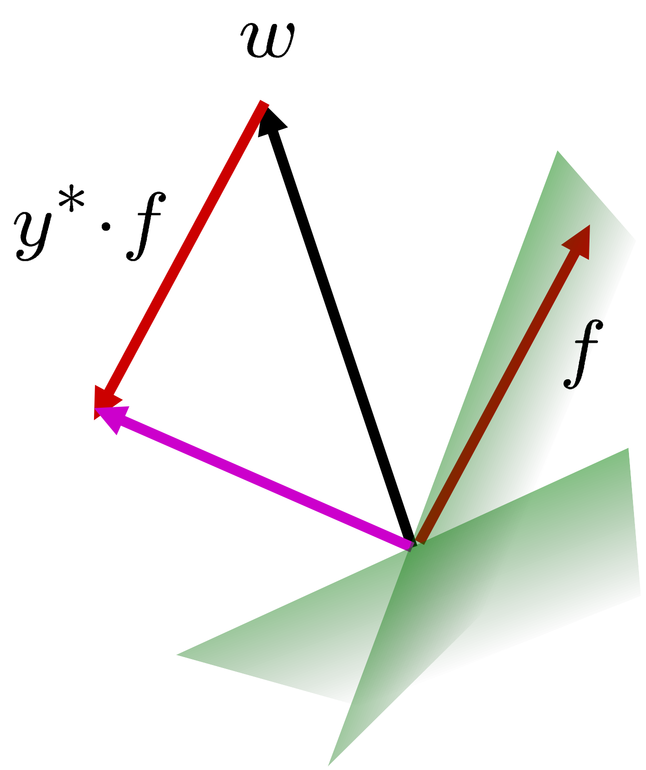
Learning: Binary Perceptron

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

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

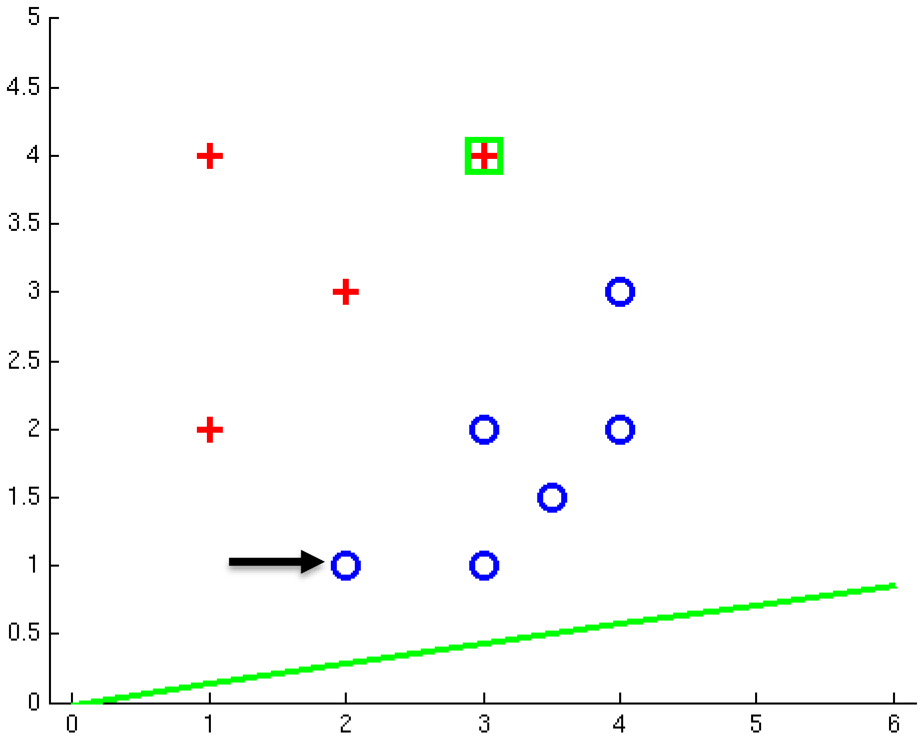
  - y*: training label
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

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



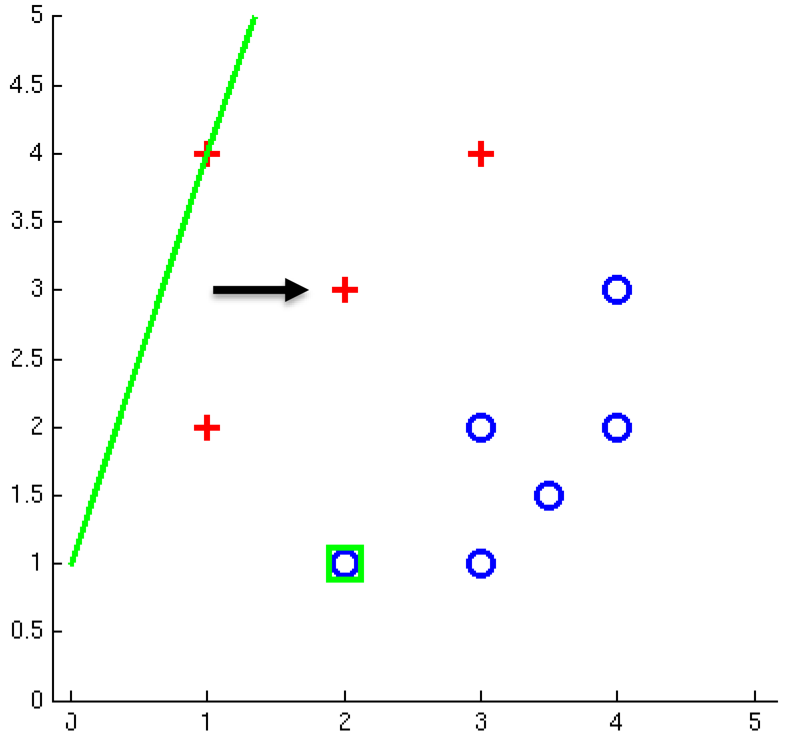
Examples: Perceptron

- Separable Case



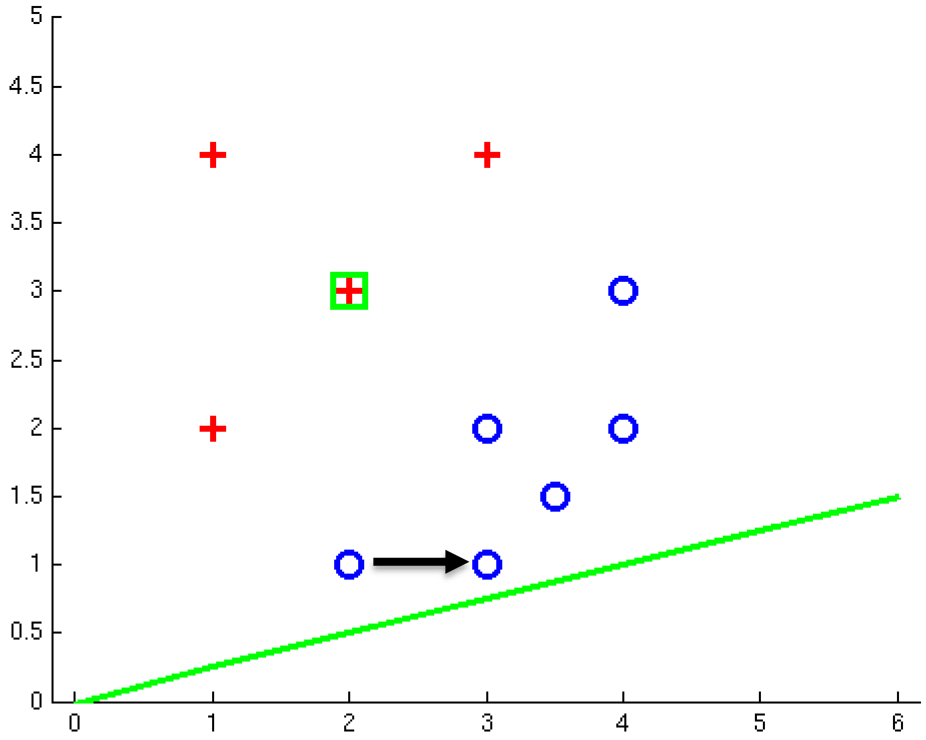
Examples: Perceptron

- Separable Case



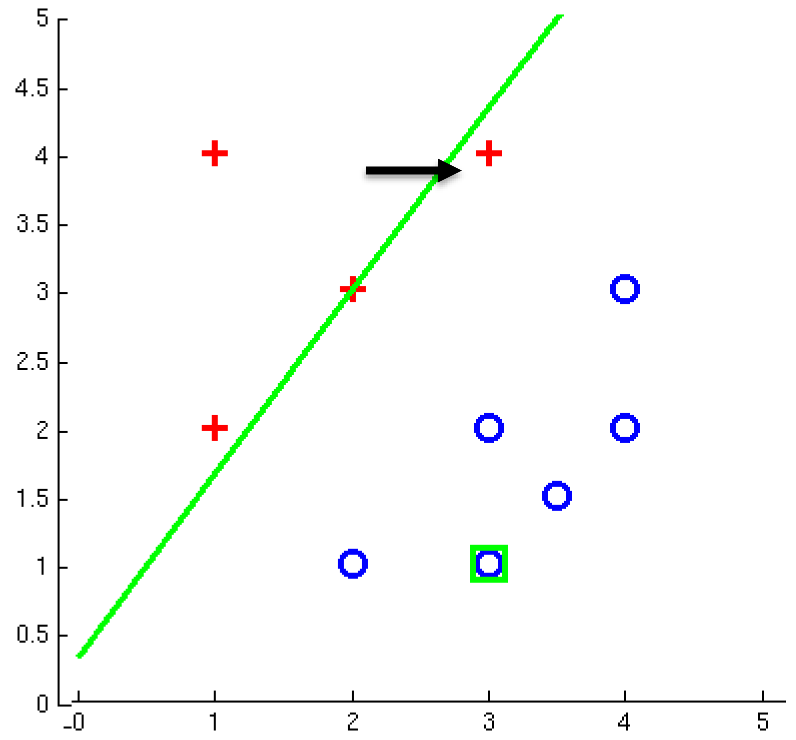
Examples: Perceptron

- Separable Case



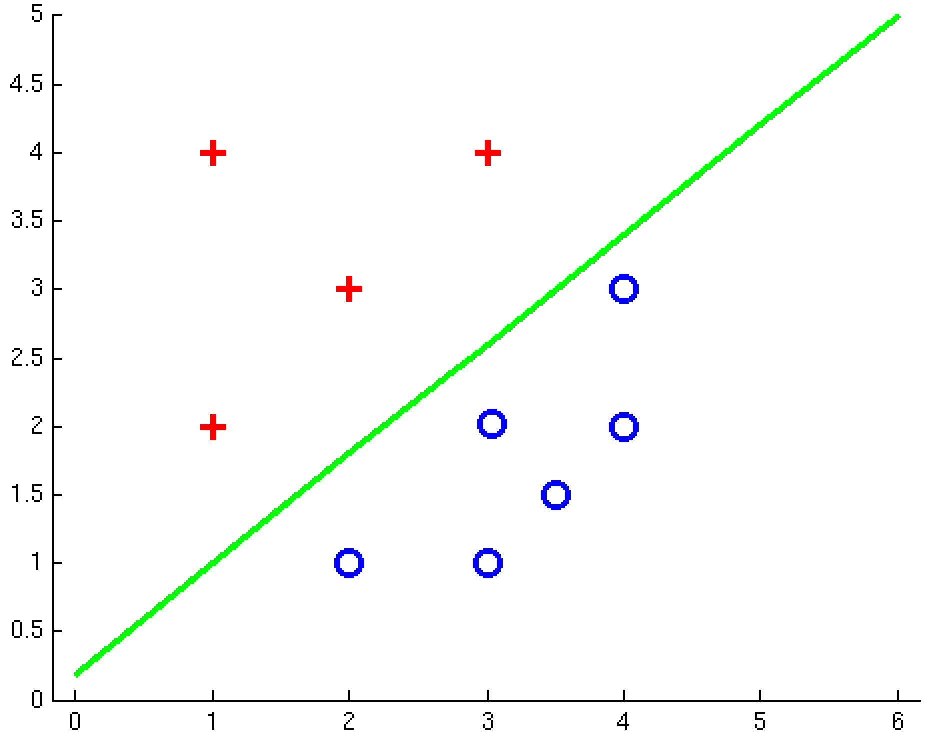
Examples: Perceptron

- Separable Case



Examples: Perceptron

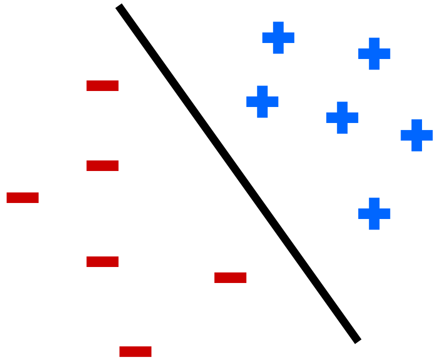
- Separable Case



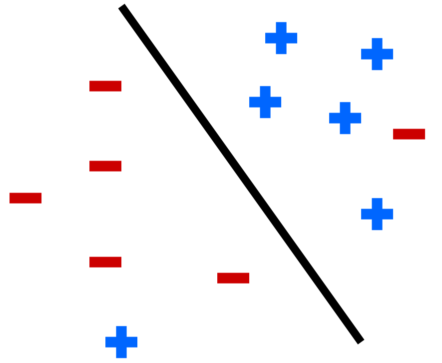
Properties of Perceptrons

- Separability: some parameters get the training set perfectly correct
- Not all data is separable
- Convergence: if the training data is separable, the binary perceptron rule is guaranteed to converge

Separable

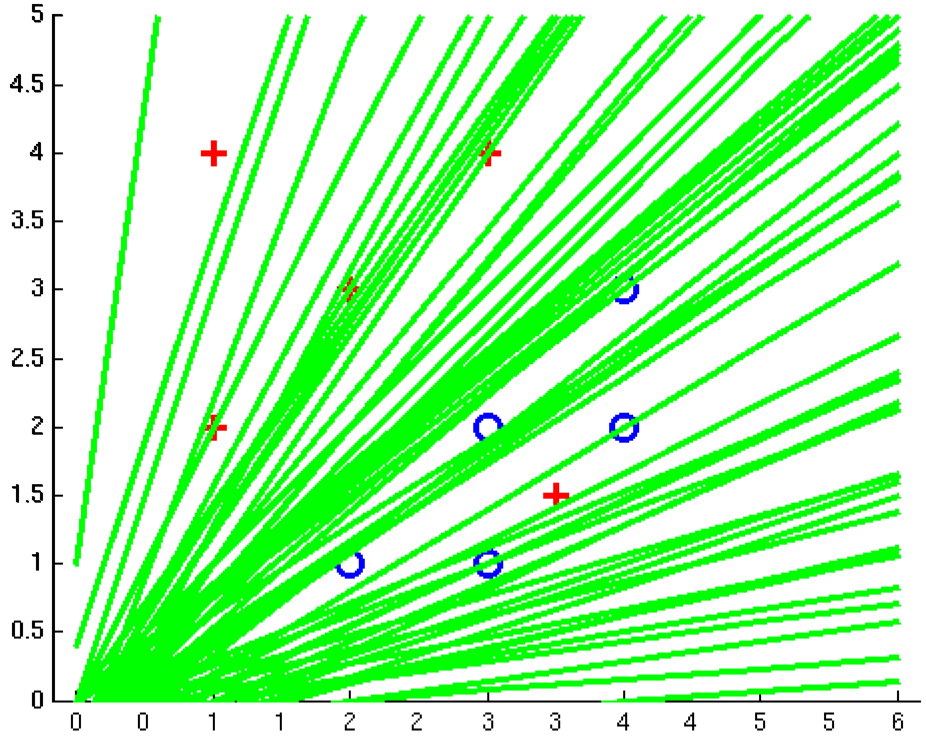


Non-Separable



Examples: Perceptron

- Non-Separable Case



Multiclass Decision Rule

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

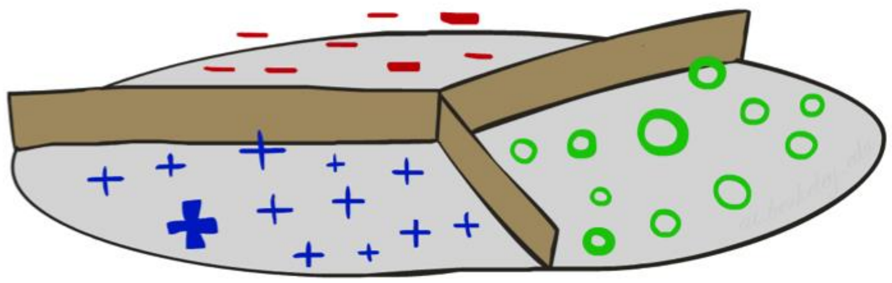
    $$w_y$$

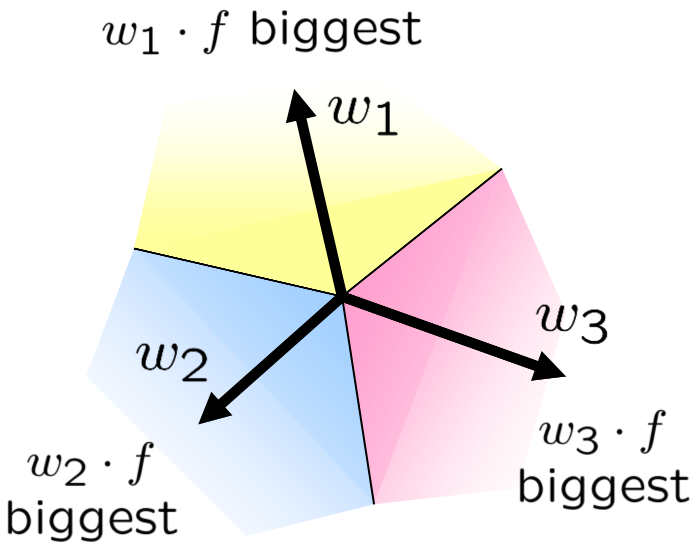
  - Score (activation) of a class y:

    $$w_y \cdot f(x)$$

  - Prediction: highest score wins

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





Learning: Multiclass Perceptron

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

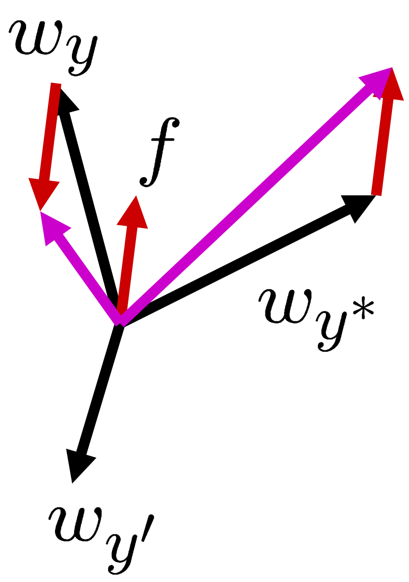
$$y = \arg\max_y \; w_y \cdot f(x)$$

- If correct, no change!
- If wrong: lower score of wrong answer (y), raise score of right answer (y*)

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

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

- Repeat until no more significant changes (iterations)



Example: Multiclass Perceptron

Training Examples:	Training Labels
"win the vote"	POLITICS
"win the election"	POLITICS
"win the game"	SPORTS

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

Example: Multiclass Perceptron

f(“win the vote”)

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

$w_{\text{SPORTS}} \cdot f$(“win the vote”) = 1

$w_{\text{POLITICS}} \cdot f$(“win the vote”) = 0

$w_{\text{POLITICS}} \cdot f$(“win the vote”) = 0

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

Example: Multiclass Perceptron

f("win the vote")

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

y = SPORTS

y* = POLITICS

$w_{\text{SPORTS}}\ x\ f$("win the vote") = 1

$w_{\text{POLITICS}}\ x\ f$("win the vote") = 0

$w_{\text{TECH}}\ x\ f$("win the vote") = 0

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

Example: Multiclass Perceptron

f("win the vote")

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

y = SPORTS

y* = POLITICS

$$w_{\text{SPORTS}} = w_{\text{SPORTS}} - f$$

$$w_{\text{POLITICS}} = w_{\text{POLITICS}} + f$$

w_{SPORTS}

BIAS	:	0
win	:	-1
game	:	0
vote	:	-1
the	:	-1

$w_{POLITICS}$

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

w_{TECH}

BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0

Example: Multiclass Perceptron

f("win the election")

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

w_{SPORTS} x f("win the election") = -2

w_{POLITICS} x f("win the election") = 3

w_{TECH} x f("win the election") = 0

w_{SPORTS}

BIAS	: 0
win	: -1
game	: 0
vote	: -1
the	: -1

$w_{POLITICS}$

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

w_{TECH}

BIAS	: 0
win	: 0
game	: 0
vote	: 0
the	: 0

Example: Multiclass Perceptron

f("win the election")

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

w_{SPORTS} x f("win the election") = -2

w_{POLITICS} x f("win the election") = 3

w_{TECH} x f("win the election") = 0

w_{SPORTS}

BIAS	:	0
win	:	-1
game	:	0
vote	:	-1
the	:	-1

$w_{POLITICS}$

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

w_{TECH}

BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0

Example: Multiclass Perceptron

f("win the election")

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

y = POLITICS

y* = POLITICS

No updates

w_{SPORTS} x f("win the election") = -2

$w_{POLITICS}$ x f("win the election") = 3

w_{TECH} x f("win the election") = 0

w_{SPORTS}

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

$w_{POLITICS}$

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

w_{TECH}

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

Example: Multiclass Perceptron

f("win the game")

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

$w_{\text{SPORTS}}\ x\ f$("win the game") = -2

$w_{\text{POLITICS}}\ x\ f$("win the game") = 3

$w_{\text{TECH}}\ x\ f$("win the game") = 0

w_{SPORTS}

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

$w_{POLITICS}$

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

w_{TECH}

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

Example: Multiclass Perceptron

f("win the game")

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

y = POLITICS

y* = SPORTS

$w_{SPORTS} \times f$("win the game") = -2

$w_{POLITICS} \times f$("win the game") = 3

$w_{TECH} \times f$("win the game") = 0

w_{SPORTS}

BIAS	:	0
win	:	-1
game	:	0
vote	:	-1
the	:	-1

$w_{POLITICS}$

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

w_{TECH}

BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0

Example: Multiclass Perceptron

f("win the game")

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

y = POLITICS

y* = SPORTS

$$w_{\text{POLITICS}} = w_{\text{POLITICS}} - f$$

$$w_{\text{SPORTS}} = w_{\text{SPORTS}} + f$$

$$w_{SPORTS}$$

BIAS	:	1
win	:	0
game	:	1
vote	:	-1
the	:	0

$$w_{POLITICS}$$

BIAS	:	0
win	:	0
game	:	-1
vote	:	1
the	:	0

$$w_{TECH}$$

BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0

Example: Multiclass Perceptron

We're not done.

We go back for a second iteration over the examples and update the weights.

We keep iterating over the training examples until there are no more changes.

w_{SPORTS}

BIAS	:	1
win	:	0
game	:	1
vote	:	-1
the	:	0

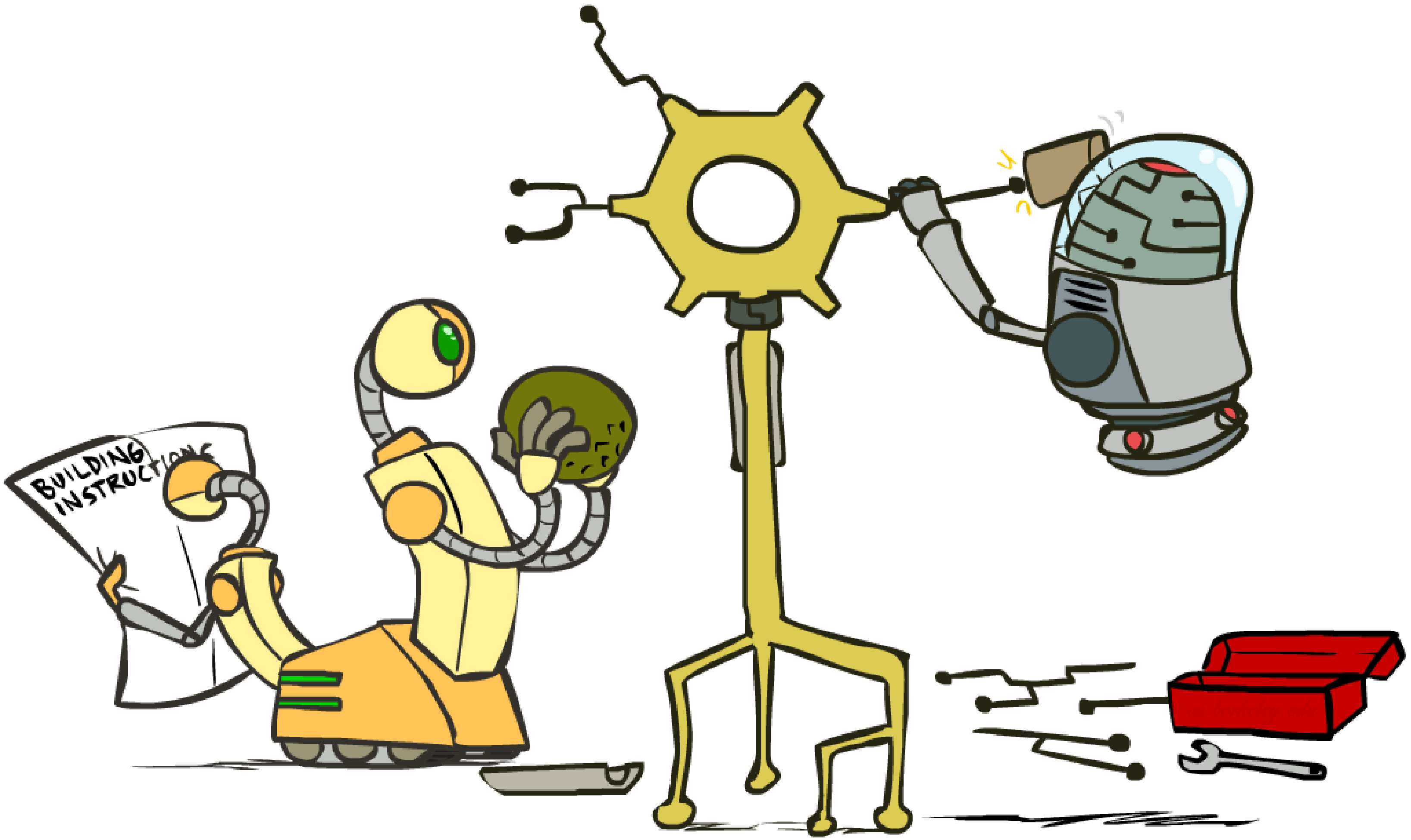
$w_{POLITICS}$

BIAS	:	0
win	:	0
game	:	-1
vote	:	1
the	:	0

w_{TECH}

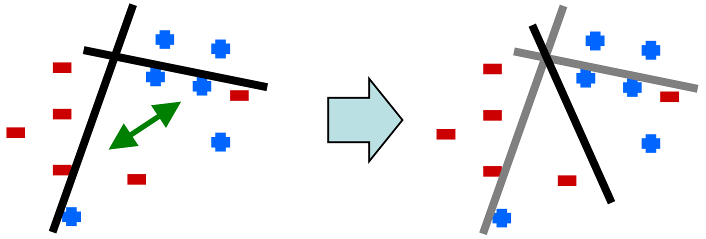
BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0

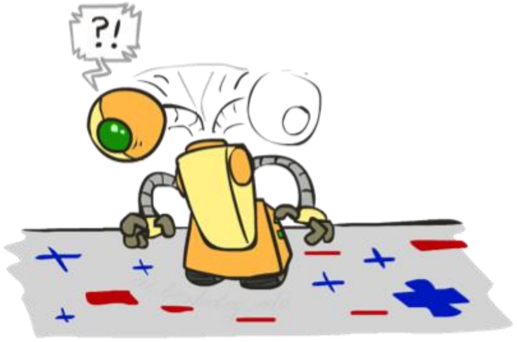
Improving the Perceptron



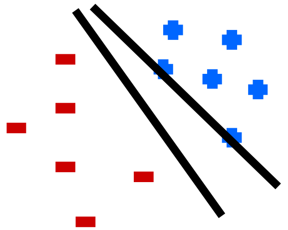
Problems with the Perceptron

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

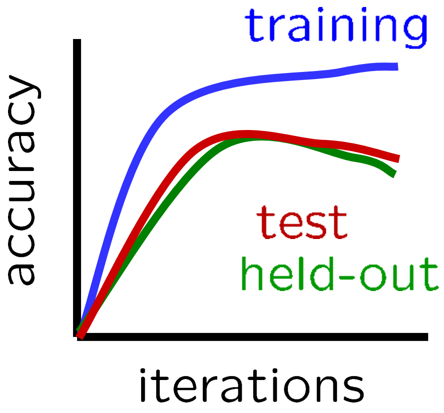


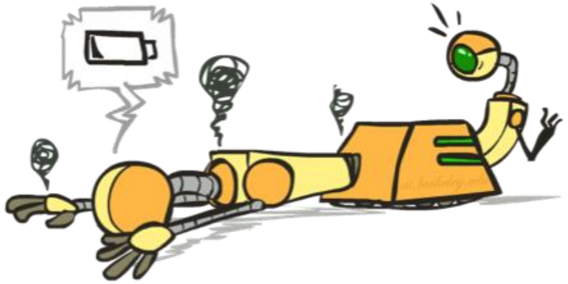


- Mediocre generalization: finds a "barely" separating solution



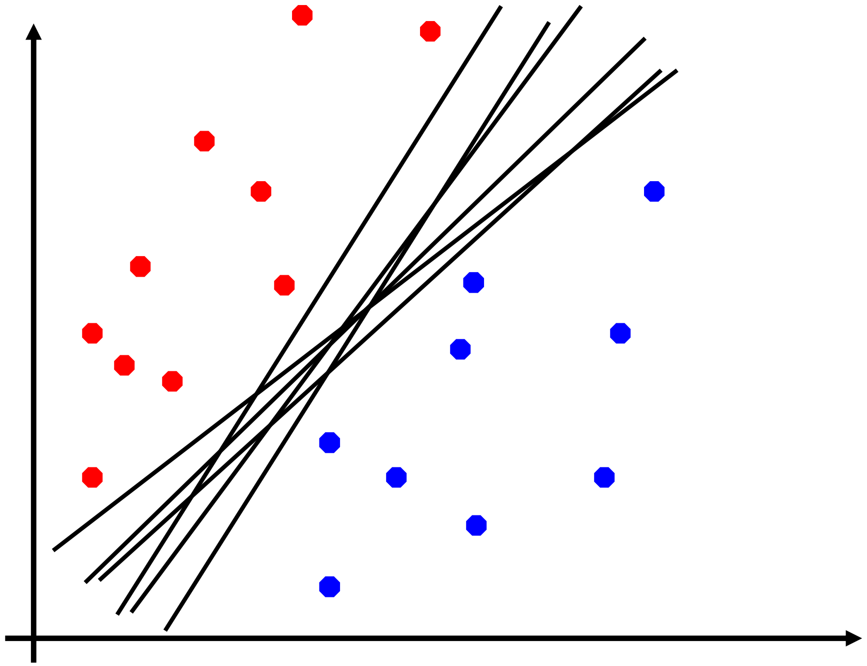
- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting





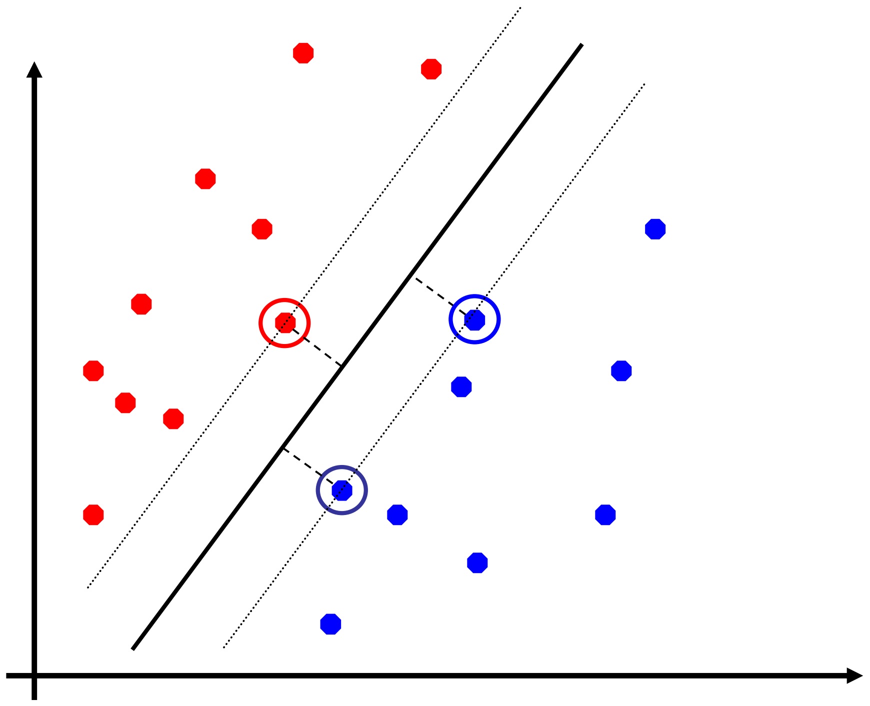
Linear Separators

- Which of these linear separators is better?



Support Vector Machines

- Maximizing the margin: good according to intuition, theory, practice
- Only support vectors (vectors on the margin) matter; other training examples are ignorable
- Support vector machines (SVMs) find the separator with the maximum margin



Classification: Comparison

- Naïve Bayes
  - Builds a model training data
  - Gives prediction probabilities
  - Strong assumptions about feature independence
  - One pass through data (counting)

- Perceptrons:
  - Makes less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data (prediction)
  - Often more accurate