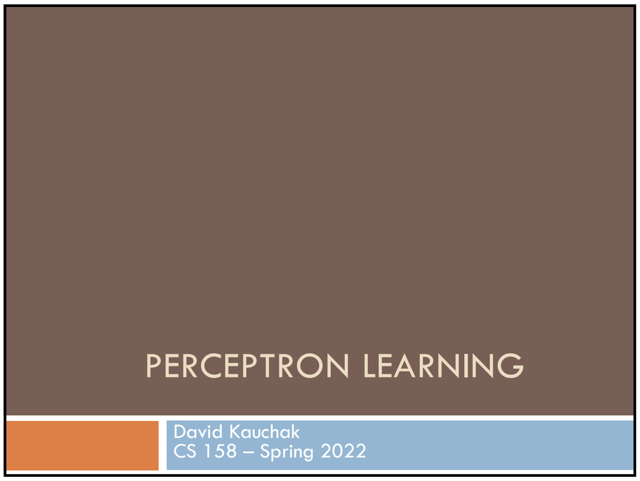
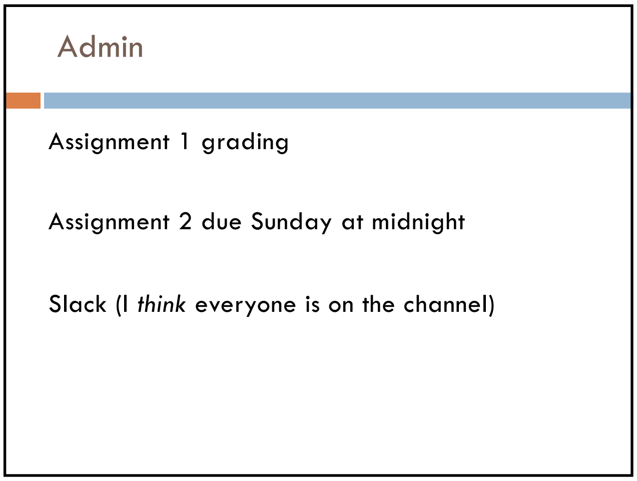


PERCEPTRON LEARNING

David Kauchak
CS 158 – Spring 2022

1



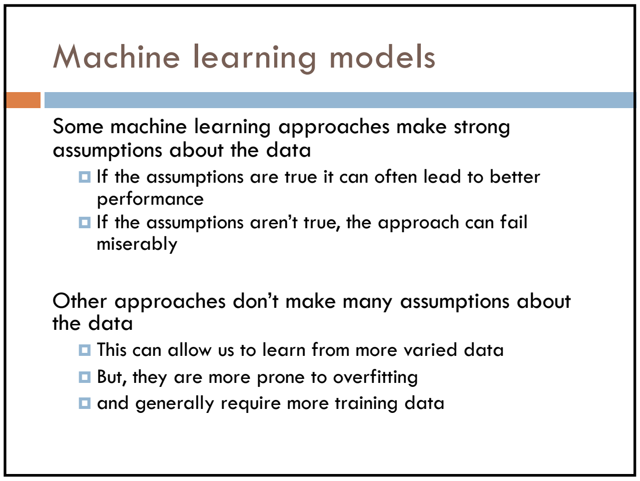
Admin

Assignment 1 grading

Assignment 2 due Sunday at midnight

Slack (I *think* everyone is on the channel)

2



Machine learning models

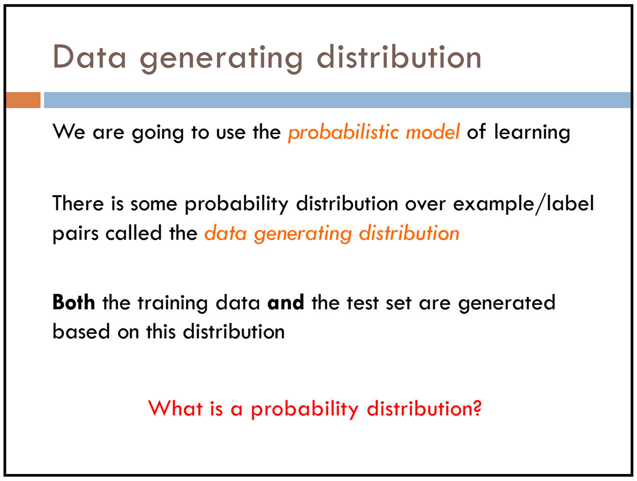
Some machine learning approaches make strong assumptions about the data

- If the assumptions are true it can often lead to better performance
- If the assumptions aren't true, the approach can fail miserably

Other approaches don't make many assumptions about the data

- This can allow us to learn from more varied data
- But, they are more prone to overfitting
- and generally require more training data

3



Data generating distribution

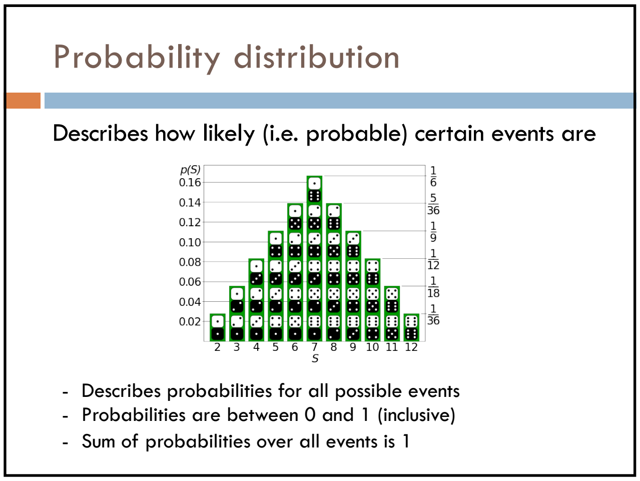
We are going to use the *probabilistic model* of learning

There is some probability distribution over example/label pairs called the *data generating distribution*

Both the training data **and** the test set are generated based on this distribution

What is a probability distribution?

4

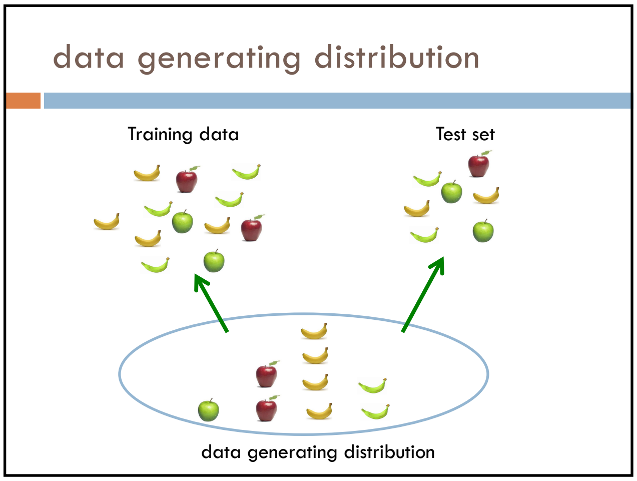


Probability distribution

Describes how likely (i.e. probable) certain events are

- Describes probabilities for all possible events
- Probabilities are between 0 and 1 (inclusive)
- Sum of probabilities over all events is 1

5



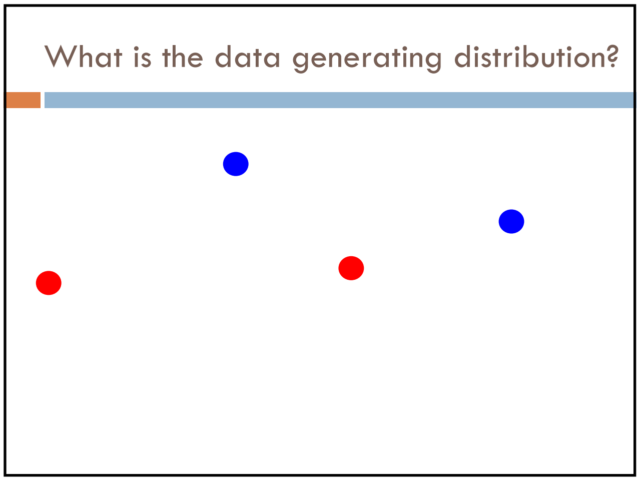
data generating distribution

Training data

Test set

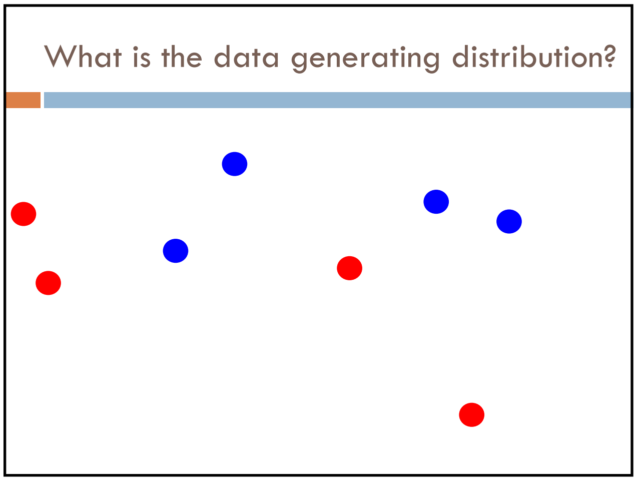
data generating distribution

6



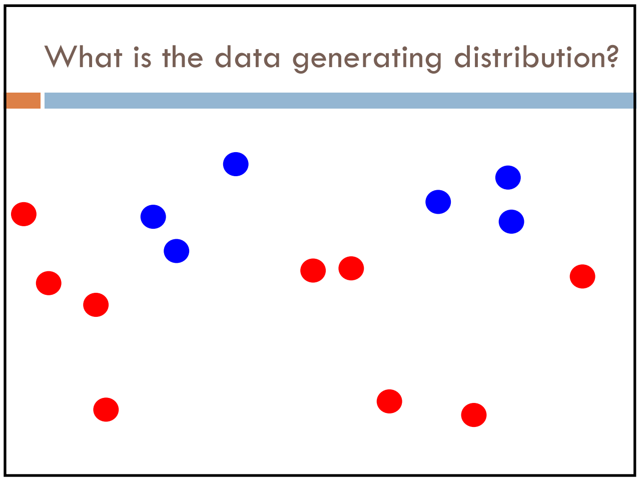
What is the data generating distribution?

7



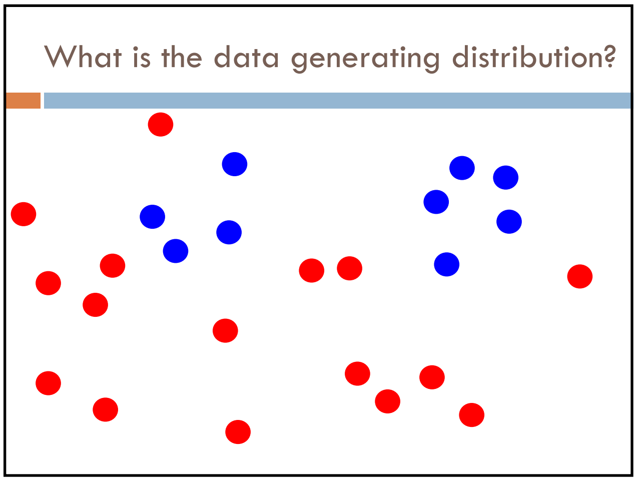
What is the data generating distribution?

8



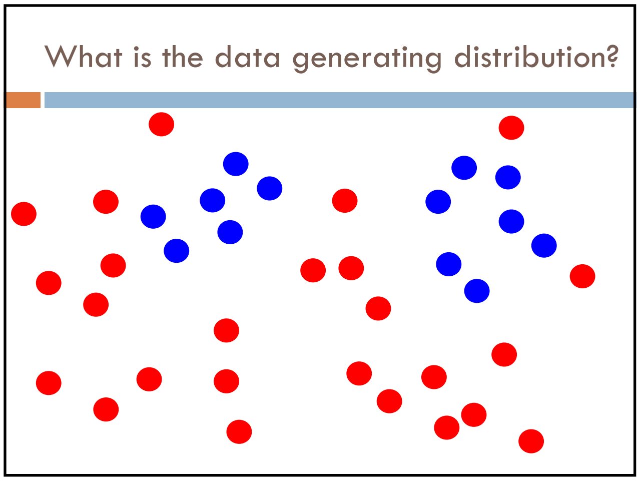
What is the data generating distribution?

9



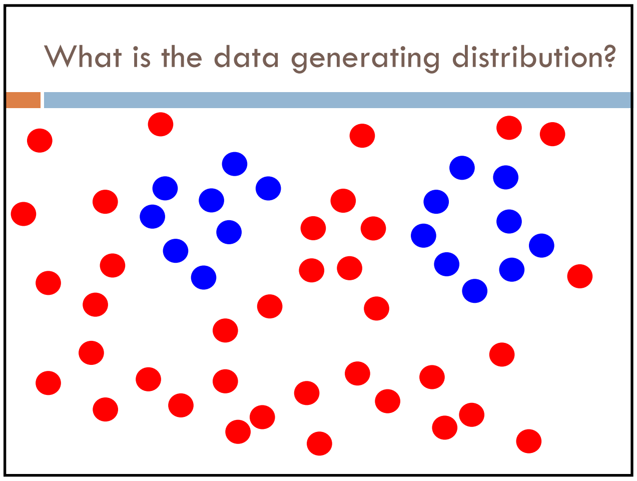
What is the data generating distribution?

10



What is the data generating distribution?

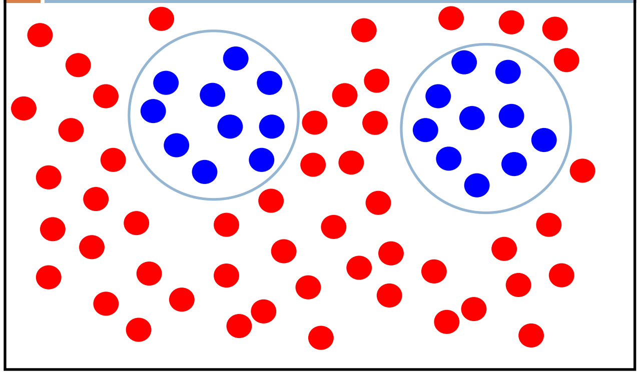
11



What is the data generating distribution?

12

Actual model



13

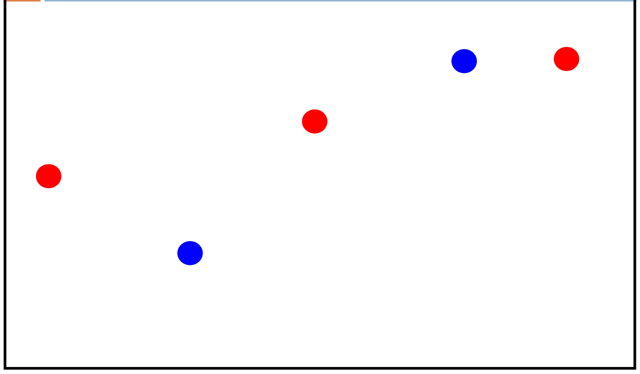
Model assumptions

If you don’t have strong assumptions about the model, it can take you a longer to learn

Assume now that our model of the blue class is two circles

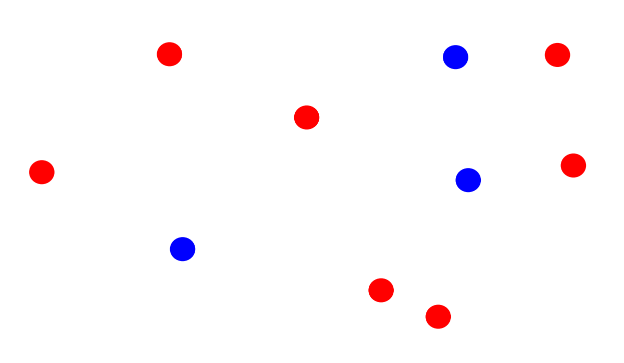
14

What is the data generating distribution?

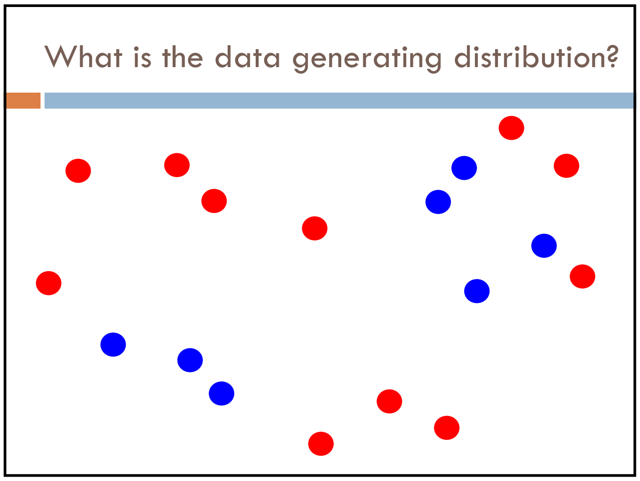


15

What is the data generating distribution?

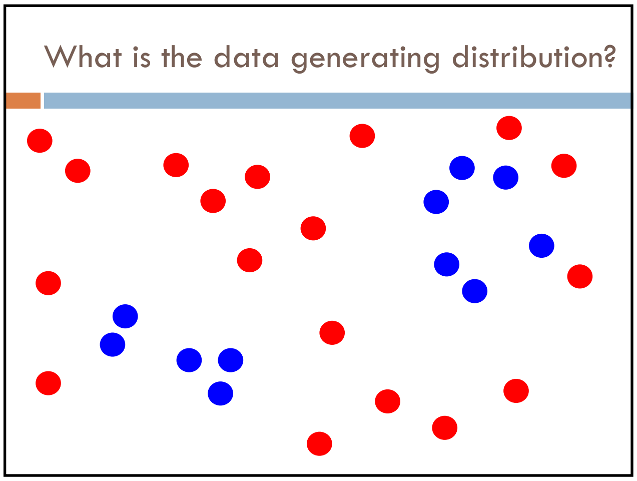


16



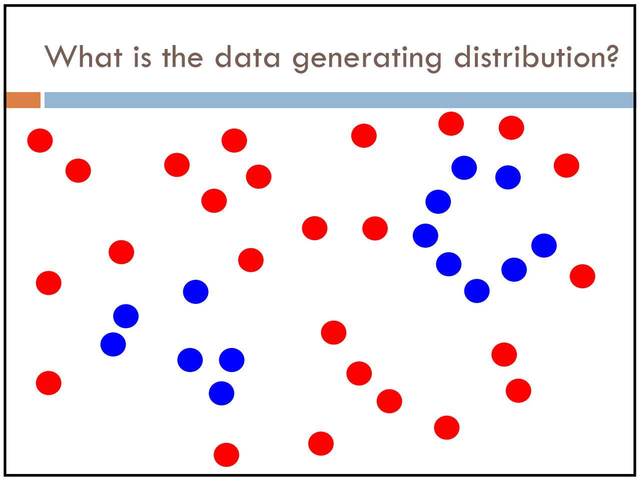
What is the data generating distribution?

17



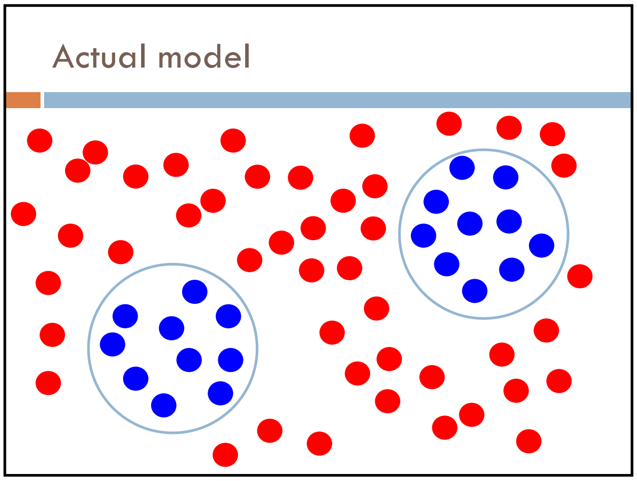
What is the data generating distribution?

18



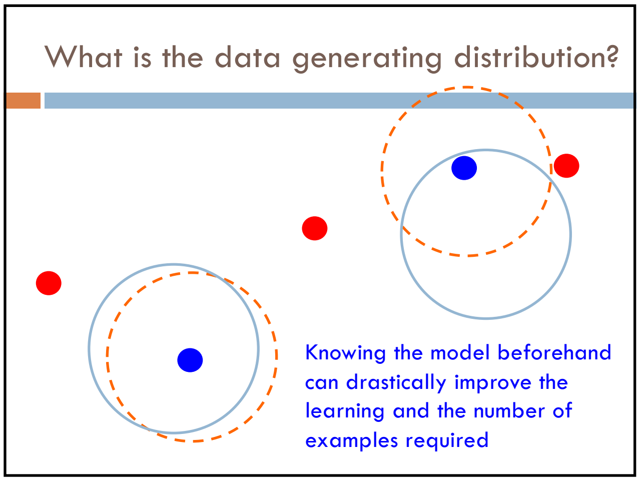
What is the data generating distribution?

19



Actual model

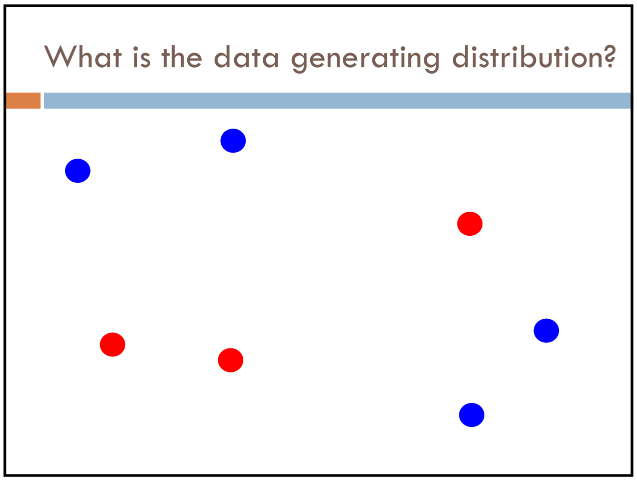
20



What is the data generating distribution?

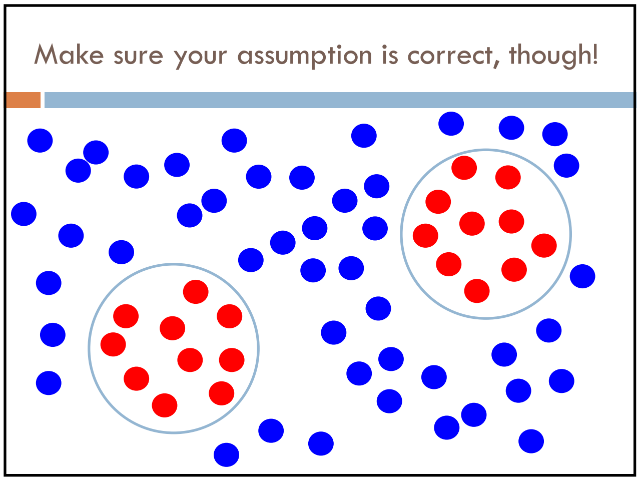
Knowing the model beforehand can drastically improve the learning and the number of examples required

21



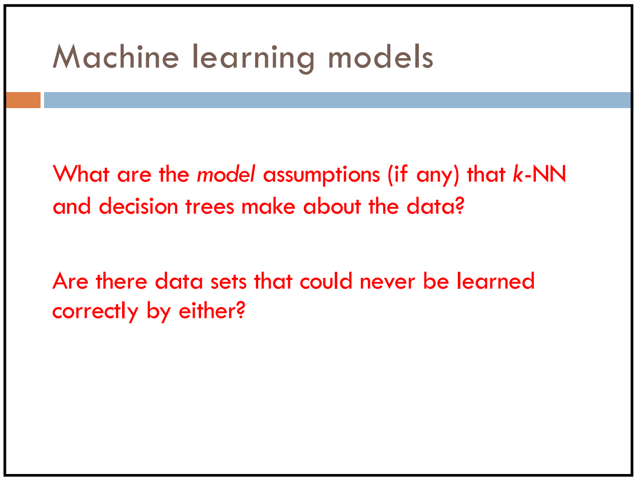
What is the data generating distribution?

22



Make sure your assumption is correct, though!

23



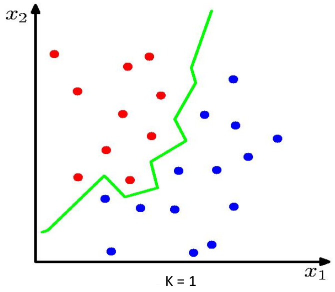
Machine learning models

What are the *model* assumptions (if any) that *k*-NN and decision trees make about the data?

Are there data sets that could never be learned correctly by either?

24

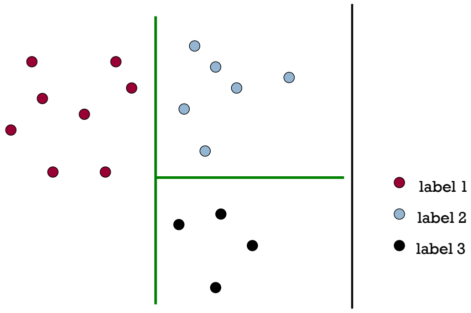
k-NN model



No model assumptions. Assumes that proximity relates to class

25

Decision tree model



Axis-aligned splits/cuts of the data

26

Bias

The "bias" of a model is how strong the model assumptions are.

low-bias classifiers make minimal assumptions about the data (*k*-NN and DT are generally considered low bias)

high-bias classifiers make strong assumptions about the data

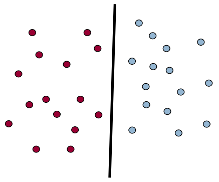
27

Linear models

A strong high-bias assumption is *linear separability*:

- in 2 dimensions, can separate classes by a line
- in higher dimensions, need hyperplanes

A *linear model* is a model that assumes the data is linearly separable



28

Hyperplanes

A hyperplane is a line/plane in a high-dimensional space

What defines a line?

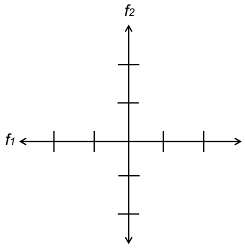
What defines a hyperplane?

29

Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$



30

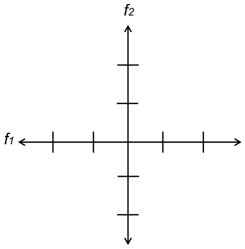
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

-2	1
-1	0.5
0	0
1	-0.5
2	-1



31

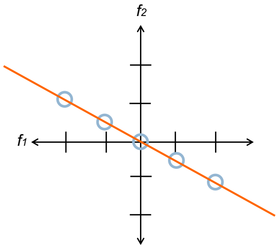
Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

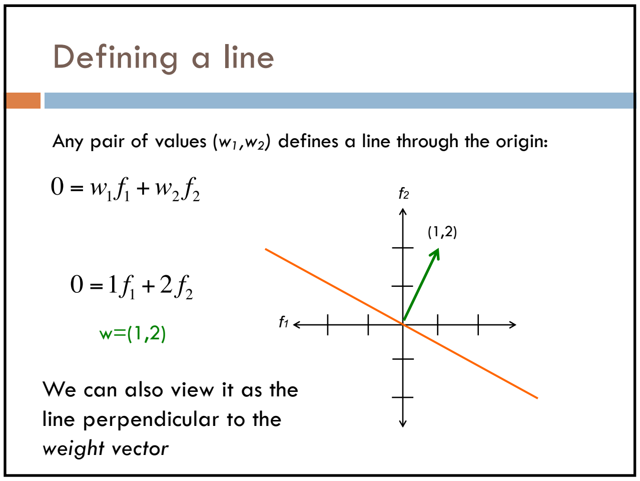
$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

-2	1
-1	0.5
0	0
1	-0.5
2	-1



32



Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

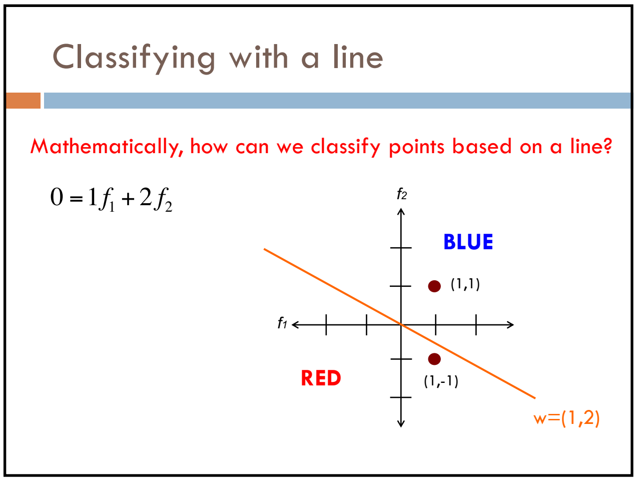
$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

w=(1,2)

We can also view it as the line perpendicular to the *weight vector*

33

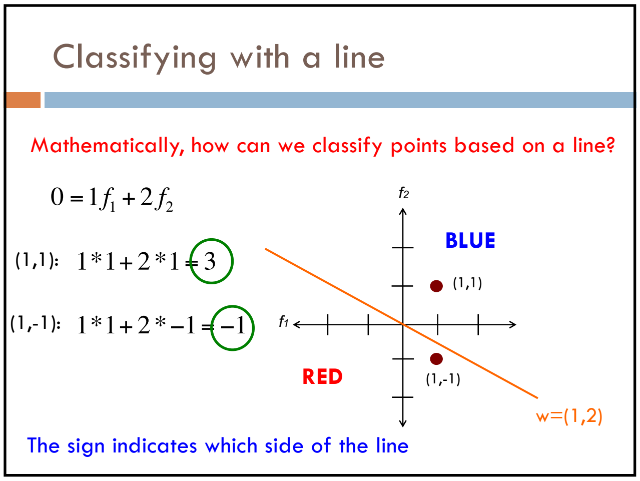


Classifying with a line

Mathematically, how can we classify points based on a line?

$$0 = 1 f_1 + 2 f_2$$

34



Classifying with a line

Mathematically, how can we classify points based on a line?

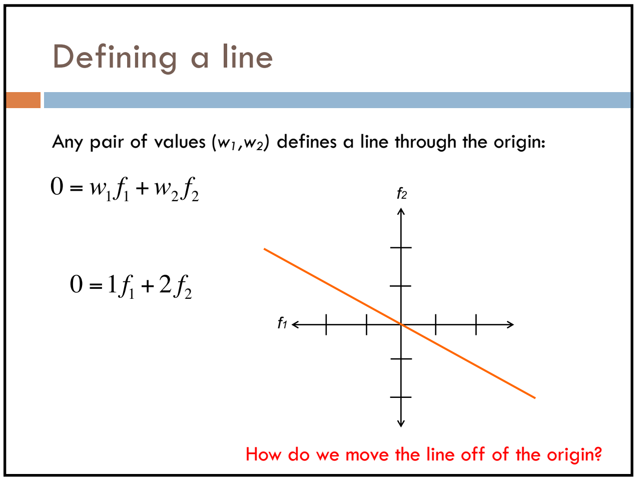
$$0 = 1 f_1 + 2 f_2$$

(1,1): $1*1+2*1 = \boxed{3}$

(1,-1): $1*1+2*-1 = \boxed{-1}$

The sign indicates which side of the line

35



Defining a line

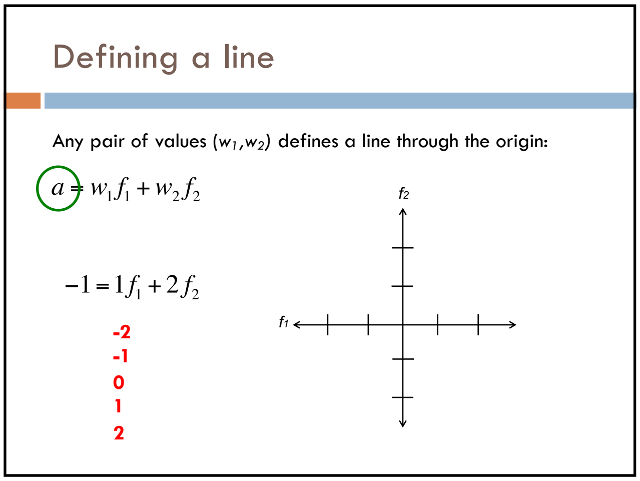
Any pair of values (w_1,w_2) defines a line through the origin:

$$0 = w_1 f_1 + w_2 f_2$$

$$0 = 1 f_1 + 2 f_2$$

How do we move the line off of the origin?

36



Defining a line

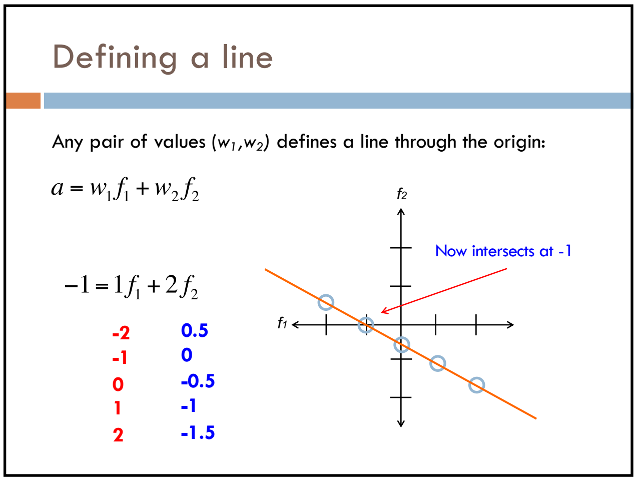
Any pair of values (w_1,w_2) defines a line through the origin:

$$a = w_1 f_1 + w_2 f_2$$

$$-1 = 1 f_1 + 2 f_2$$

-2
-1
0
1
2

37



Defining a line

Any pair of values (w_1,w_2) defines a line through the origin:

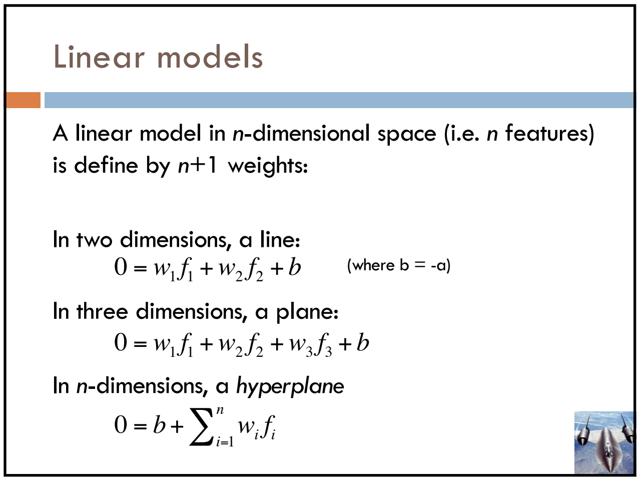
$$a = w_1 f_1 + w_2 f_2$$

$$-1 = 1 f_1 + 2 f_2$$

-2	0.5
-1	0
0	-0.5
1	-1
2	-1.5

f2
f1
Now intersects at -1

38



Linear models

A linear model in *n*-dimensional space (i.e. *n* features) is define by *n*+1 weights:

In two dimensions, a line:

$$0 = w_1 f_1 + w_2 f_2 + b$$ (where b = -a)

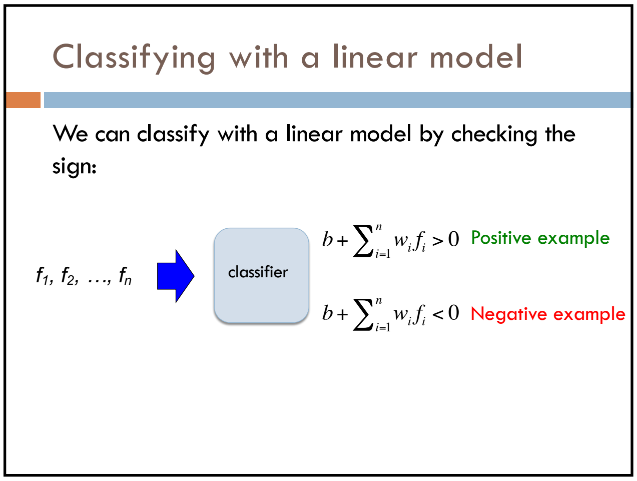
In three dimensions, a plane:

$$0 = w_1 f_1 + w_2 f_2 + w_3 f_3 + b$$

In *n*-dimensions, a *hyperplane*

$$0 = b + \sum_{i=1}^{n} w_i f_i$$

39



Classifying with a linear model

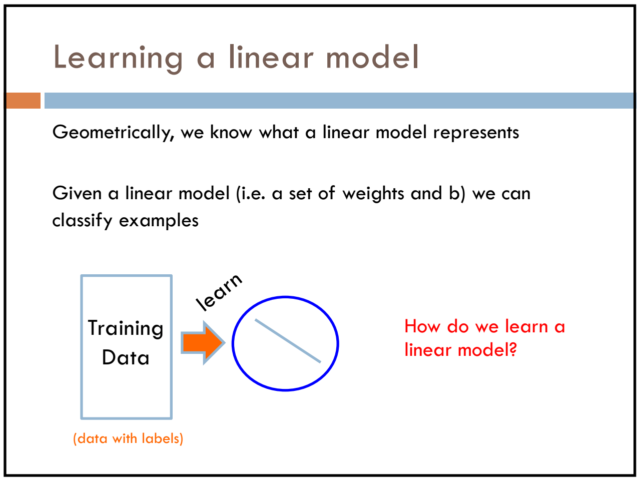
We can classify with a linear model by checking the sign:

$f_1, f_2, \ldots, f_n$
classifier

$b + \sum_{i=1}^{n} w_i f_i > 0$ Positive example

$b + \sum_{i=1}^{n} w_i f_i < 0$ Negative example

40



Learning a linear model

Geometrically, we know what a linear model represents

Given a linear model (i.e. a set of weights and b) we can classify examples

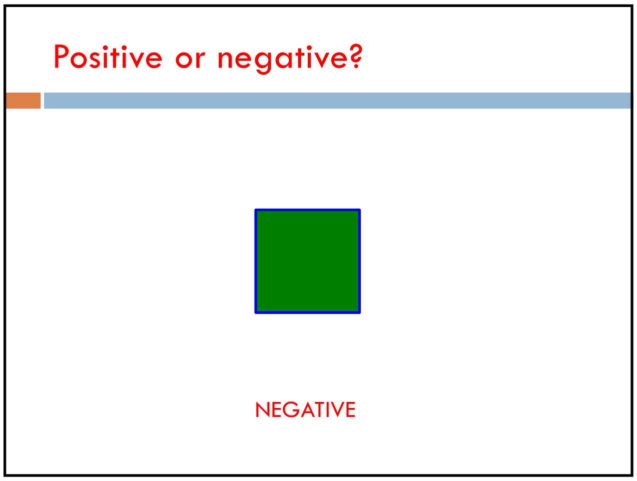
Training Data

(data with labels)

learn

How do we learn a linear model?

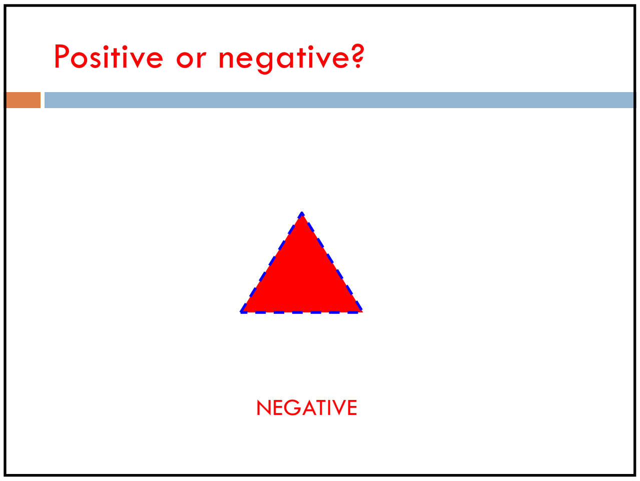
41



Positive or negative?

NEGATIVE

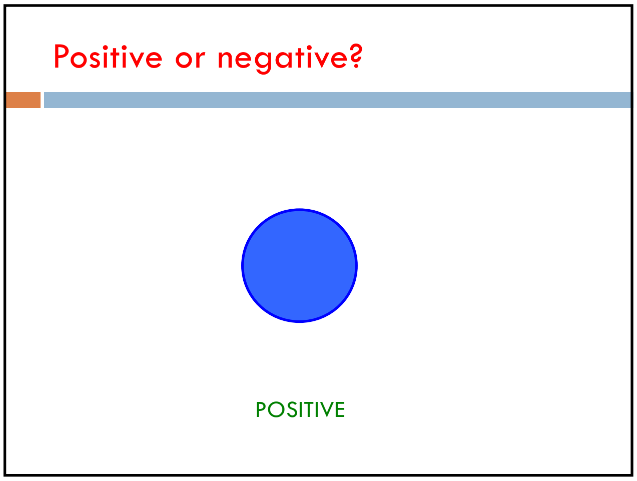
42



Positive or negative?

NEGATIVE

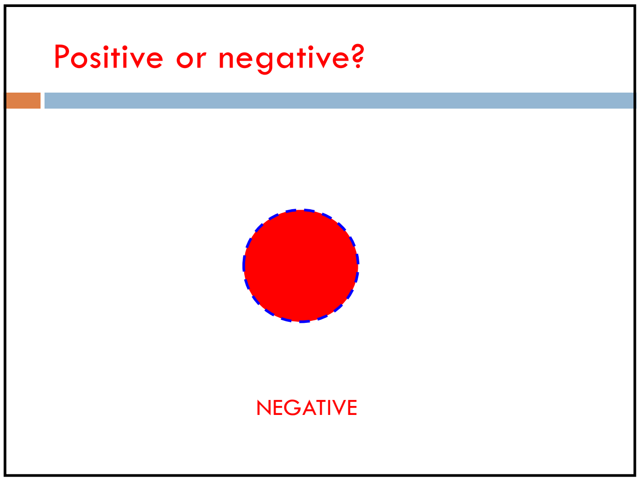
43



Positive or negative?

POSITIVE

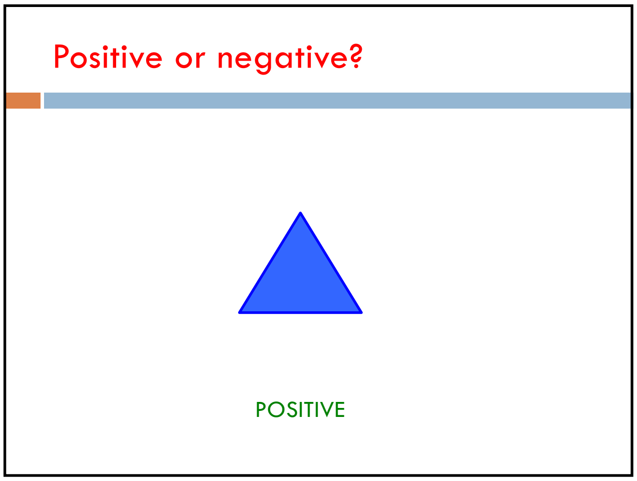
44



Positive or negative?

NEGATIVE

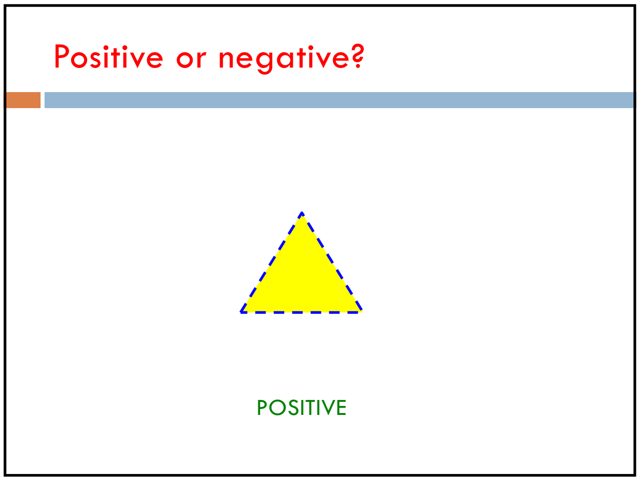
45



Positive or negative?

POSITIVE

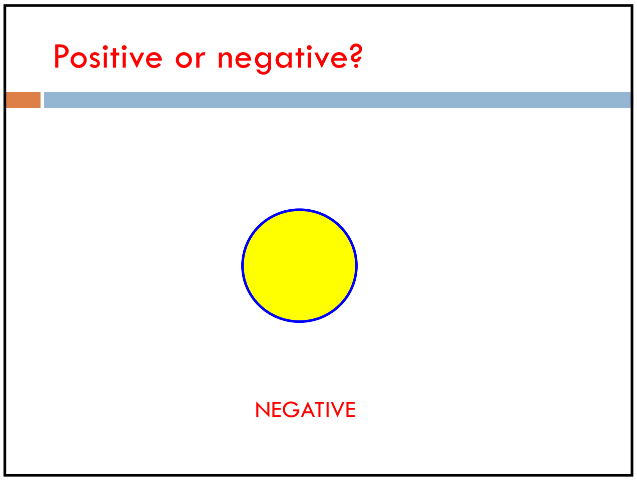
46



Positive or negative?

POSITIVE

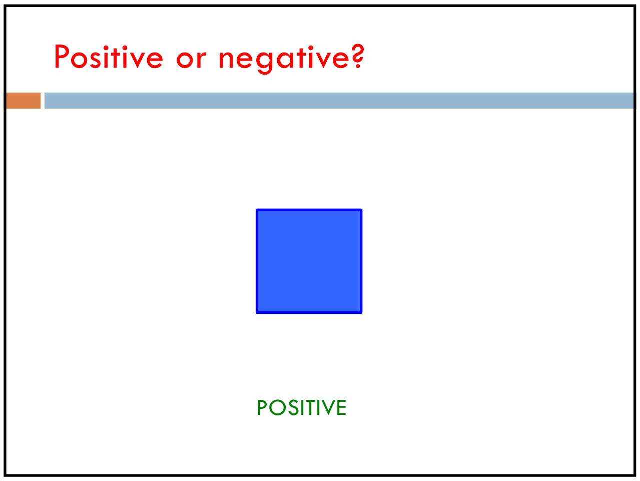
47



Positive or negative?

NEGATIVE

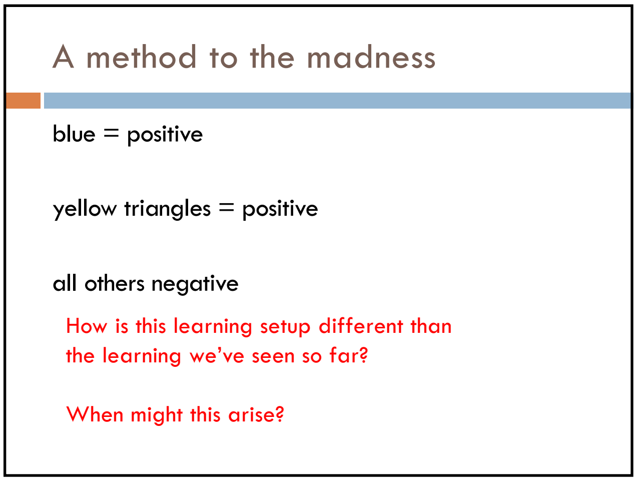
48



Positive or negative?

POSITIVE

49



A method to the madness

blue = positive

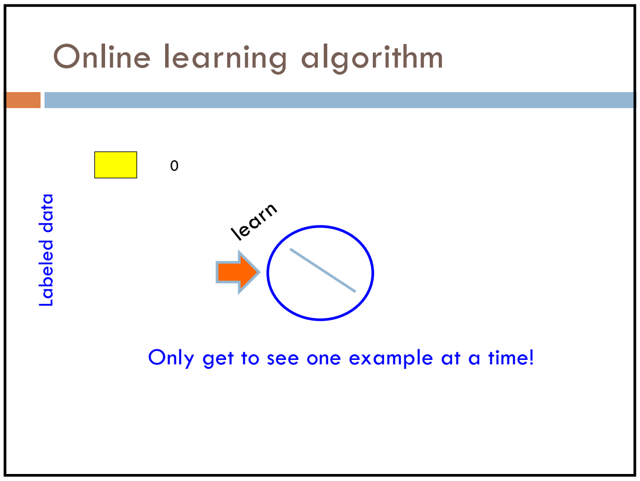
yellow triangles = positive

all others negative

How is this learning setup different than the learning we've seen so far?

When might this arise?

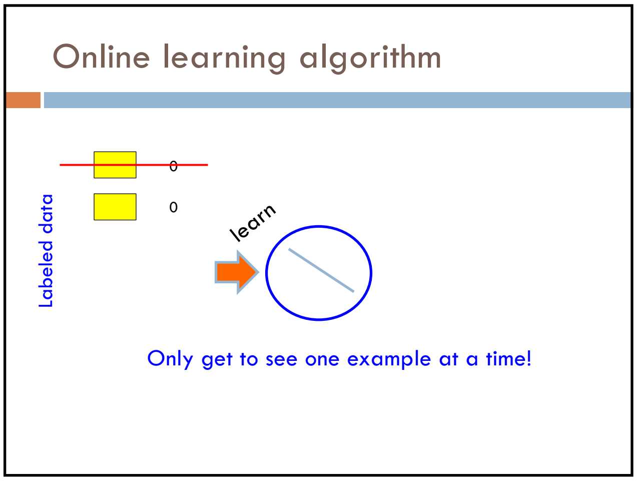
50



Online learning algorithm

Only get to see one example at a time!

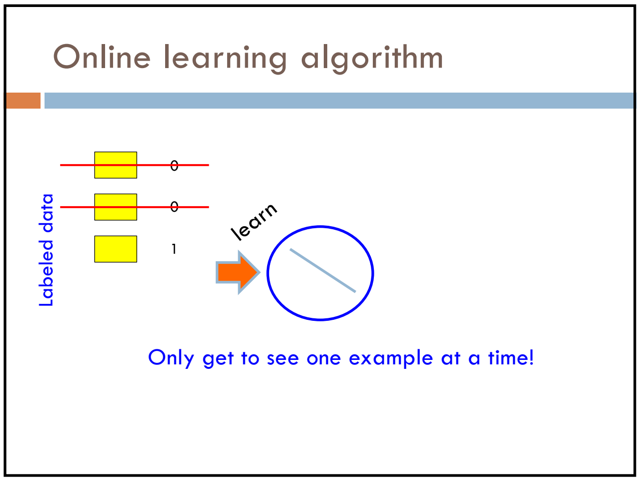
51



Online learning algorithm

Only get to see one example at a time!

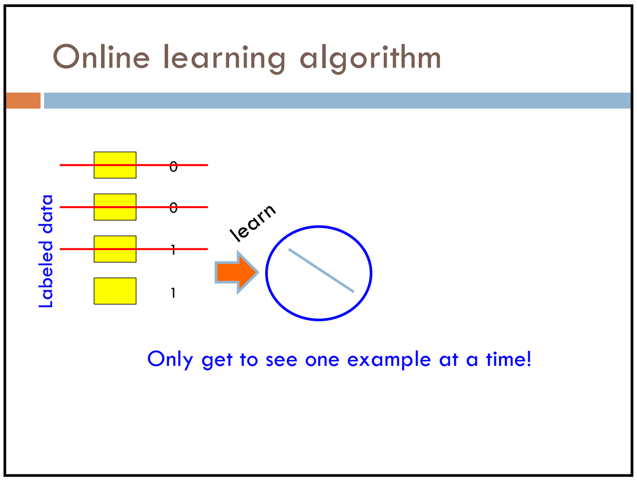
52



Online learning algorithm

Only get to see one example at a time!

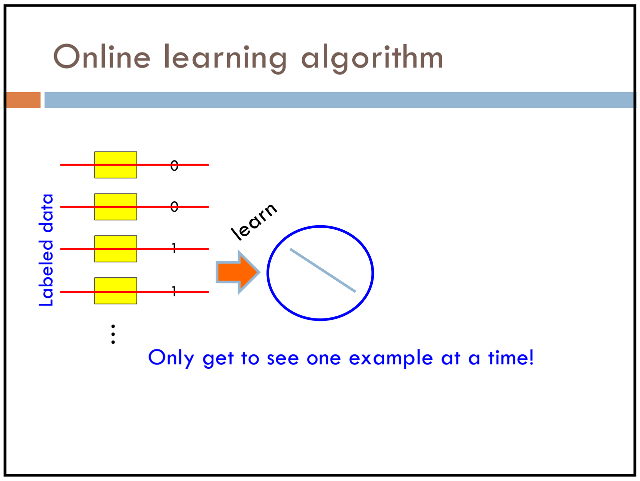
53



Online learning algorithm

Only get to see one example at a time!

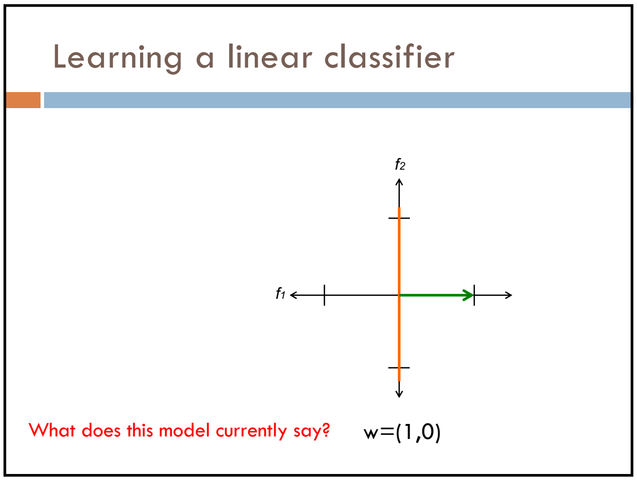
54



Online learning algorithm

Only get to see one example at a time!

55

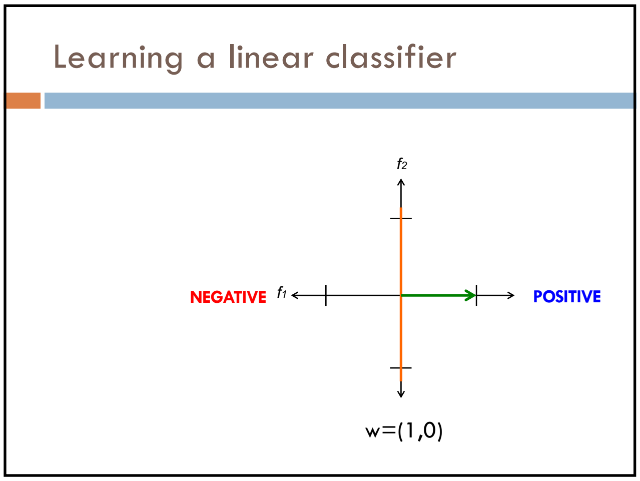


Learning a linear classifier

What does this model currently say?

w=(1,0)

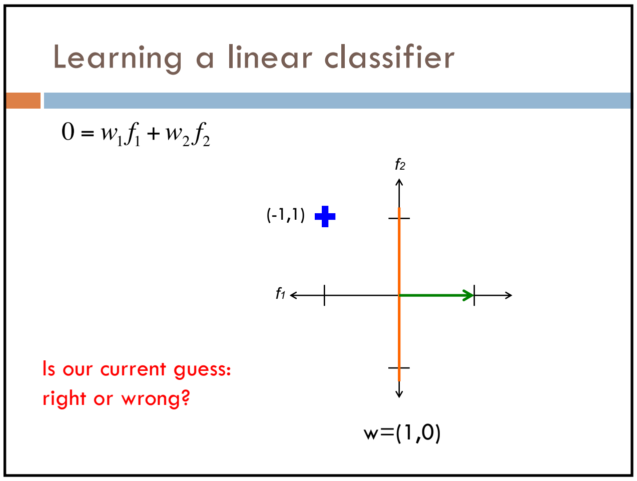
56



Learning a linear classifier

w=(1,0)

57



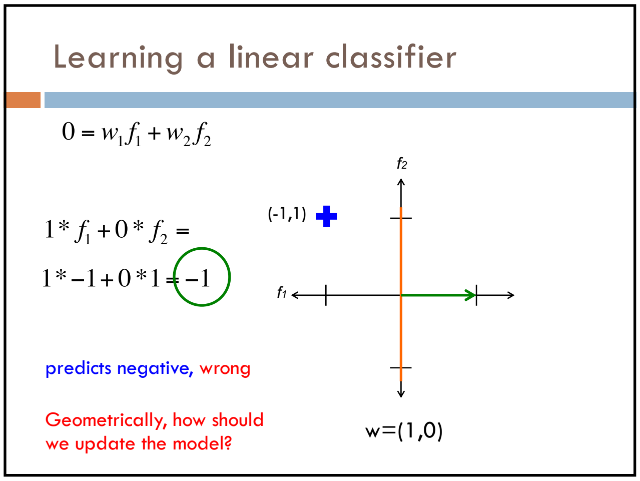
Learning a linear classifier

$$0 = w_1 f_1 + w_2 f_2$$

Is our current guess: right or wrong?

w=(1,0)

58



Learning a linear classifier

$$0 = w_1 f_1 + w_2 f_2$$

$$1 * f_1 + 0 * f_2 =$$

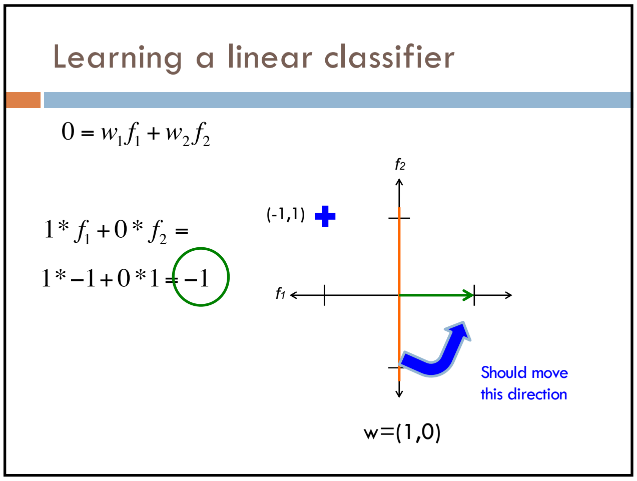
$$1 * -1 + 0 * 1 = -1$$

predicts negative, wrong

Geometrically, how should we update the model?

w=(1,0)

59



Learning a linear classifier

$$0 = w_1 f_1 + w_2 f_2$$

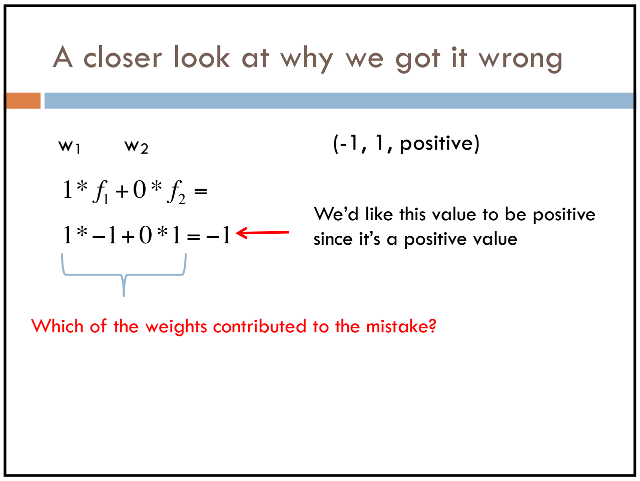
$$1 * f_1 + 0 * f_2 =$$

$$1 * -1 + 0 * 1 = -1$$

Should move this direction

w=(1,0)

60



A closer look at why we got it wrong

w1 w2

(-1, 1, positive)

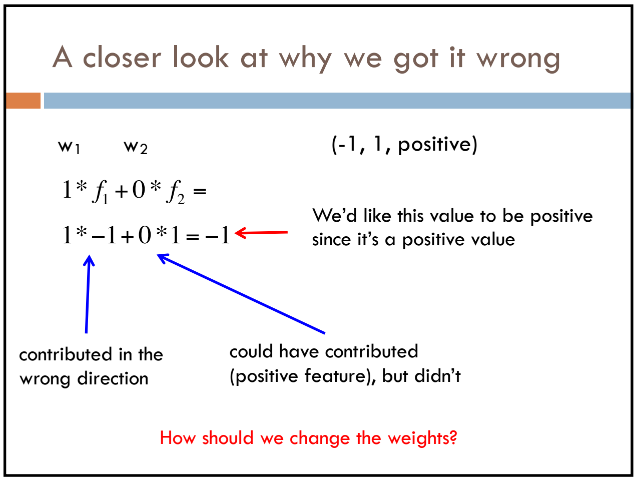
$1 * f_1 + 0 * f_2 =$

$1 * -1 + 0 * 1 = -1$

We'd like this value to be positive since it's a positive value

Which of the weights contributed to the mistake?

61



A closer look at why we got it wrong

w1 w2

(-1, 1, positive)

$1 * f_1 + 0 * f_2 =$

$1 * -1 + 0 * 1 = -1$

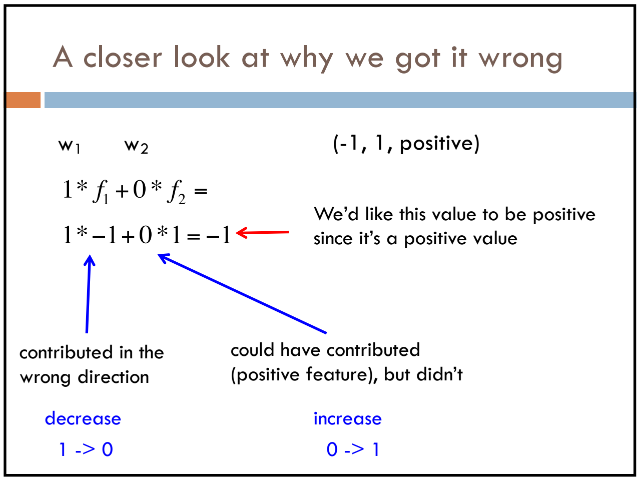
We'd like this value to be positive since it's a positive value

contributed in the wrong direction

could have contributed (positive feature), but didn't

How should we change the weights?

62



A closer look at why we got it wrong

w1 w2

(-1, 1, positive)

$1 * f_1 + 0 * f_2 =$

$1 * -1 + 0 * 1 = -1$

We'd like this value to be positive since it's a positive value

contributed in the wrong direction

could have contributed (positive feature), but didn't

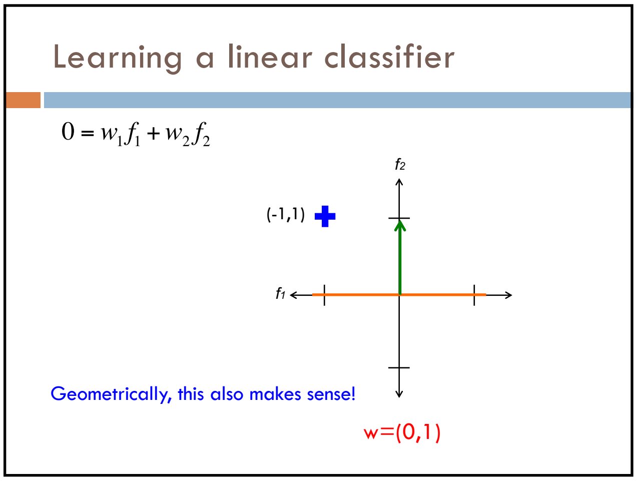
decrease

1 -> 0

increase

0 -> 1

63



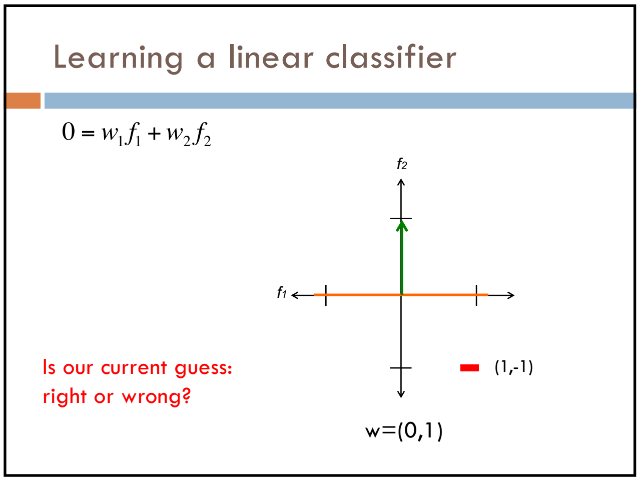
Learning a linear classifier

$0 = w_1 f_1 + w_2 f_2$

Geometrically, this also makes sense!

w=(0,1)

64



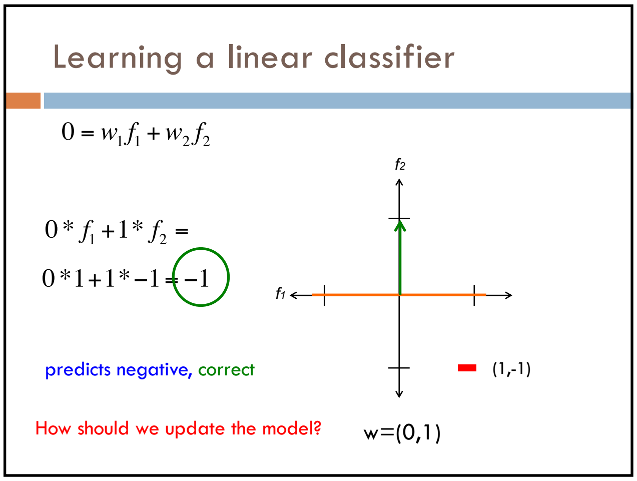
Learning a linear classifier

$0 = w_1 f_1 + w_2 f_2$

Is our current guess: right or wrong?

w=(0,1)

65



Learning a linear classifier

$0 = w_1 f_1 + w_2 f_2$

$0 * f_1 + 1 * f_2 =$

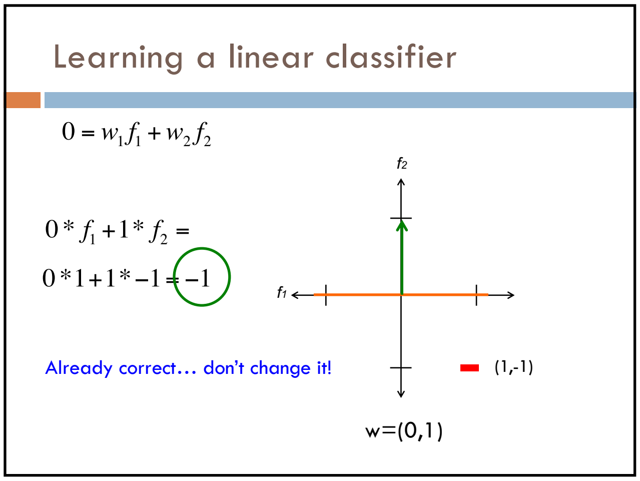
$0 * 1 + 1 * -1 = -1$

predicts negative, correct

How should we update the model?

w=(0,1)

66



Learning a linear classifier

$0 = w_1 f_1 + w_2 f_2$

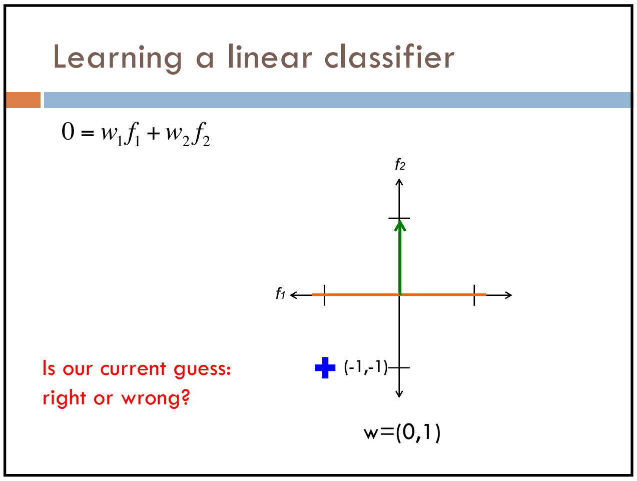
$0 * f_1 + 1 * f_2 =$

$0 * 1 + 1 * -1 = -1$

Already correct… don't change it!

w=(0,1)

67



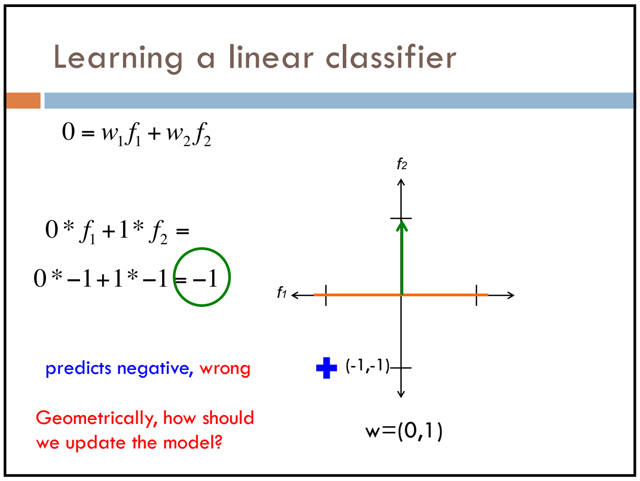
Learning a linear classifier

$0 = w_1 f_1 + w_2 f_2$

Is our current guess: right or wrong?

w=(0,1)

68



Learning a linear classifier

$0 = w_1 f_1 + w_2 f_2$

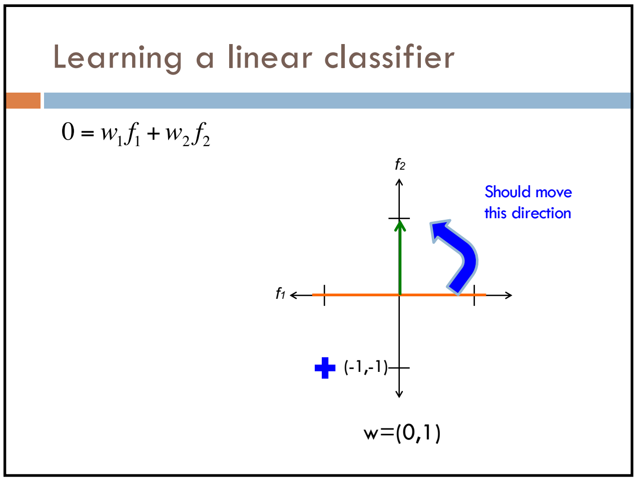
$0 * f_1 + 1 * f_2 =$

$0 * -1 + 1 * -1 \boxed{= -1}$

predicts negative, wrong

Geometrically, how should we update the model?

69

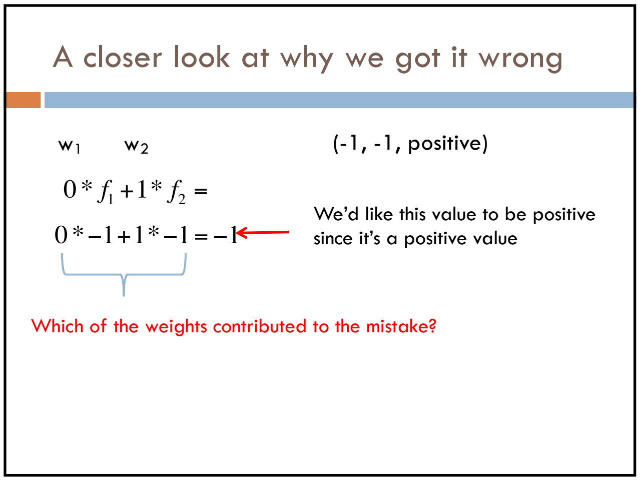


Learning a linear classifier

$0 = w_1 f_1 + w_2 f_2$

Should move this direction

70



A closer look at why we got it wrong

w_1 w_2

(-1, -1, positive)

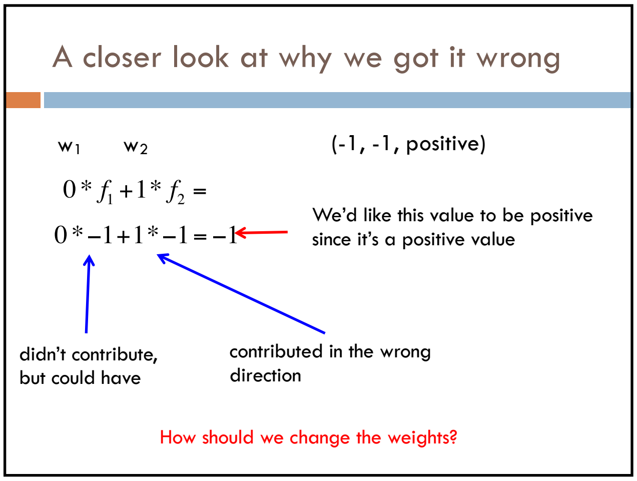
$0 * f_1 + 1 * f_2 =$

$\underbrace{0 * -1 + 1 * -1} = -1 \leftarrow$

We'd like this value to be positive since it's a positive value

Which of the weights contributed to the mistake?

71



A closer look at why we got it wrong

w_1 w_2

(-1, -1, positive)

$0 * f_1 + 1 * f_2 =$

$0 * -1 + 1 * -1 = -1 \leftarrow$

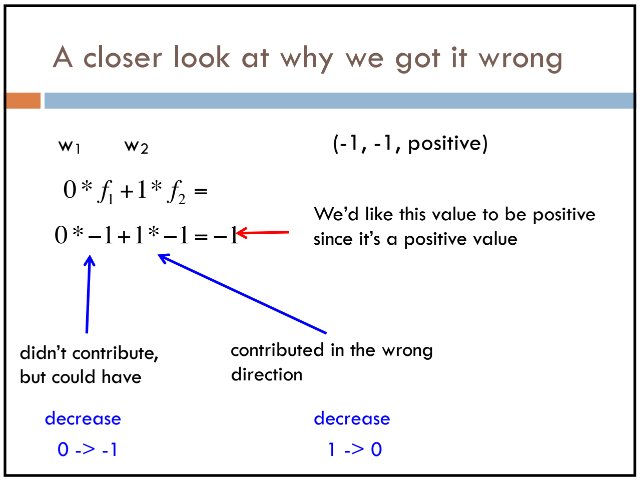
We'd like this value to be positive since it's a positive value

didn't contribute, but could have

contributed in the wrong direction

How should we change the weights?

72



A closer look at why we got it wrong

w1 w2

(-1, -1, positive)

$0 * f_1 + 1 * f_2 =$

$0 * -1 + 1 * -1 = -1$

We'd like this value to be positive since it's a positive value

didn't contribute, but could have

contributed in the wrong direction

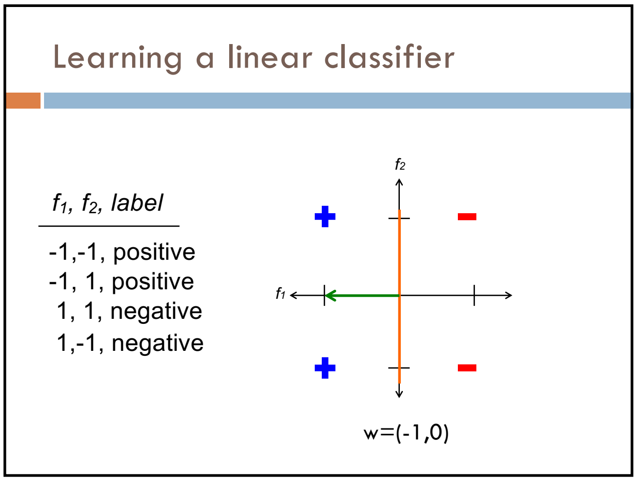
decrease

0 -> -1

decrease

1 -> 0

73

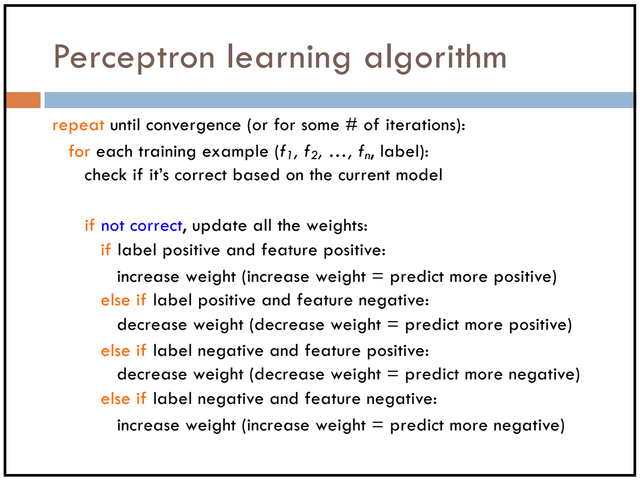


Learning a linear classifier

f_1, f_2, label
-1,-1, positive
-1, 1, positive
1, 1, negative
1,-1, negative

w=(-1,0)

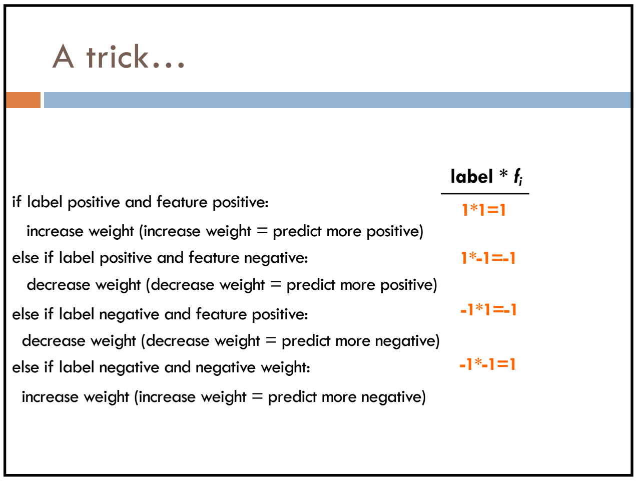
74



Perceptron learning algorithm

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    check if it's correct based on the current model

    if not correct, update all the weights:
      if label positive and feature positive:
        increase weight (increase weight = predict more positive)
      else if label positive and feature negative:
        decrease weight (decrease weight = predict more positive)
      else if label negative and feature positive:
        decrease weight (decrease weight = predict more negative)
      else if label negative and feature negative:
        increase weight (increase weight = predict more negative)
```

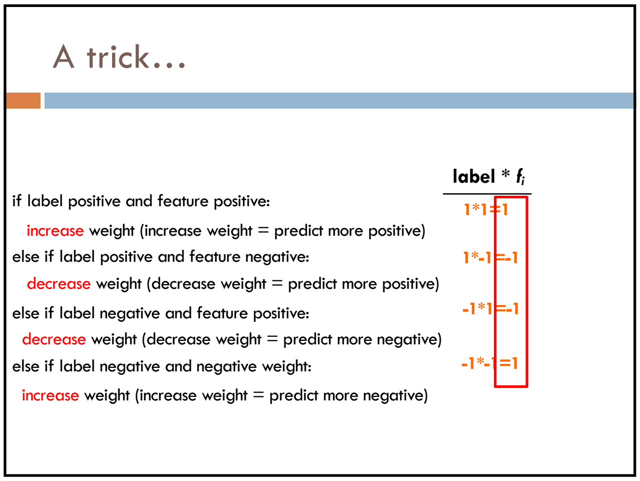
75



A trick…

	label * f_i
if label positive and feature positive:	1*1=1
increase weight (increase weight = predict more positive)	
else if label positive and feature negative:	1*-1=-1
decrease weight (decrease weight = predict more positive)	
else if label negative and feature positive:	-1*1=-1
decrease weight (decrease weight = predict more negative)	
else if label negative and negative weight:	-1*-1=1
increase weight (increase weight = predict more negative)	

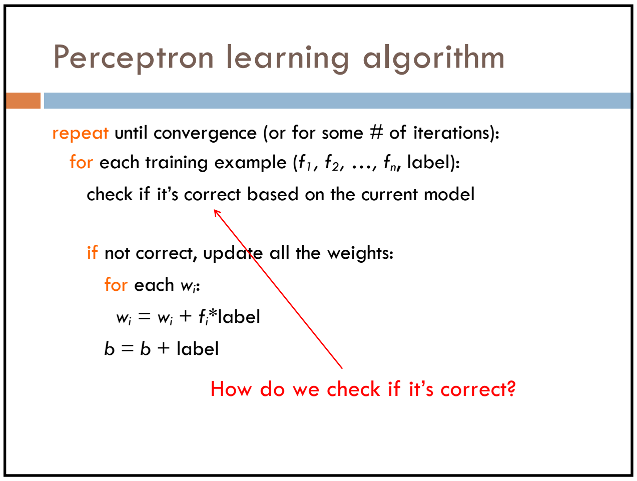
76



A trick…

	label * f_i
if label positive and feature positive:	1*1=1
increase weight (increase weight = predict more positive)	
else if label positive and feature negative:	1*-1=-1
decrease weight (decrease weight = predict more positive)	
else if label negative and feature positive:	-1*1=-1
decrease weight (decrease weight = predict more negative)	
else if label negative and negative weight:	-1*-1=1
increase weight (increase weight = predict more negative)	

77

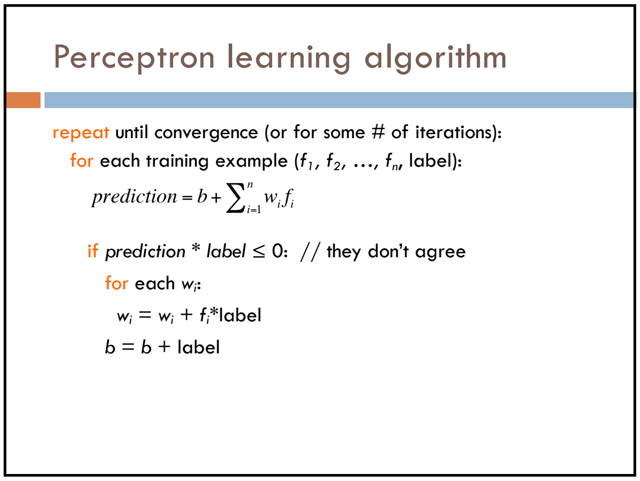


Perceptron learning algorithm

```
repeat until convergence (or for some # of iterations):
   for each training example ($f_1$, $f_2$, …, $f_n$, label):
      check if it's correct based on the current model

      if not correct, update all the weights:
         for each $w_i$:
            $w_i = w_i + f_i$*label
         $b = b$ + label
```

How do we check if it's correct?

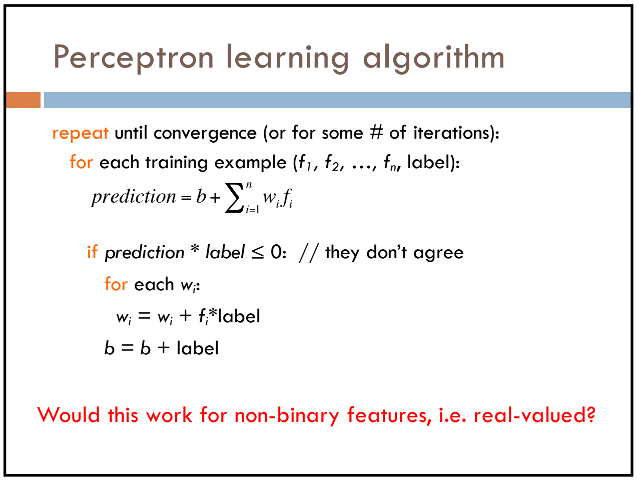
78



Perceptron learning algorithm

```
repeat until convergence (or for some # of iterations):
   for each training example ($f_1$, $f_2$, …, $f_n$, label):
      $prediction = b + \sum_{i=1}^{n} w_i f_i$

      if $prediction$ * $label \leq 0$:  // they don't agree
         for each $w_i$:
            $w_i = w_i + f_i$*label
         $b = b$ + label
```

79

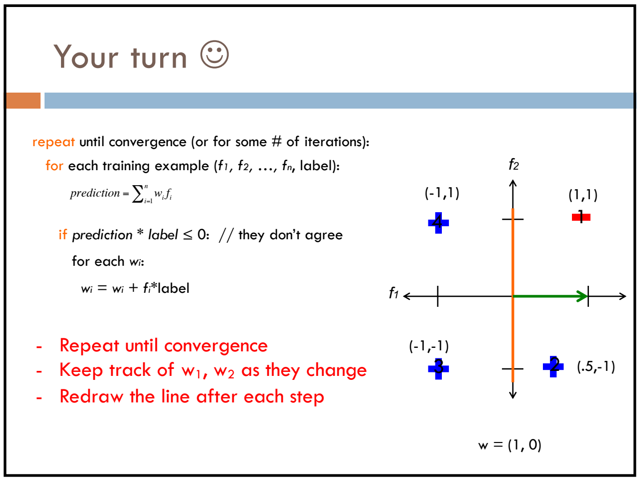


Perceptron learning algorithm

```
repeat until convergence (or for some # of iterations):
   for each training example ($f_1$, $f_2$, …, $f_n$, label):
      $prediction = b + \sum_{i=1}^{n} w_i f_i$

      if $prediction$ * $label \leq 0$:  // they don't agree
         for each $w_i$:
            $w_i = w_i + f_i$*label
         $b = b$ + label
```

Would this work for non-binary features, i.e. real-valued?

80



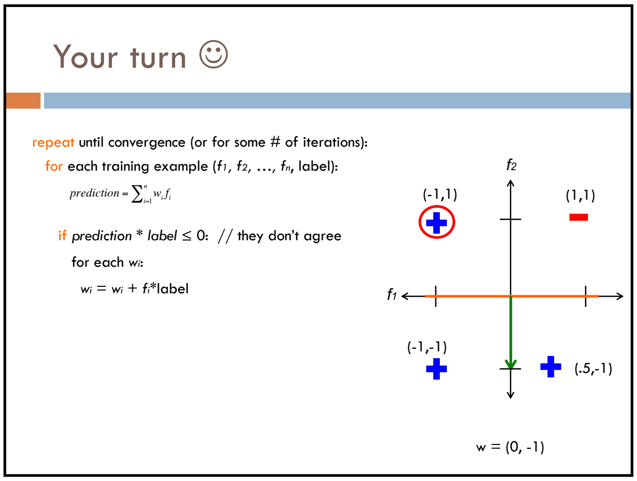
Your turn ☺

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1, f_2, …, f_n$, label):
    $prediction = \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
```

- Repeat until convergence
- Keep track of w_1, w_2 as they change
- Redraw the line after each step

w = (1, 0)

81

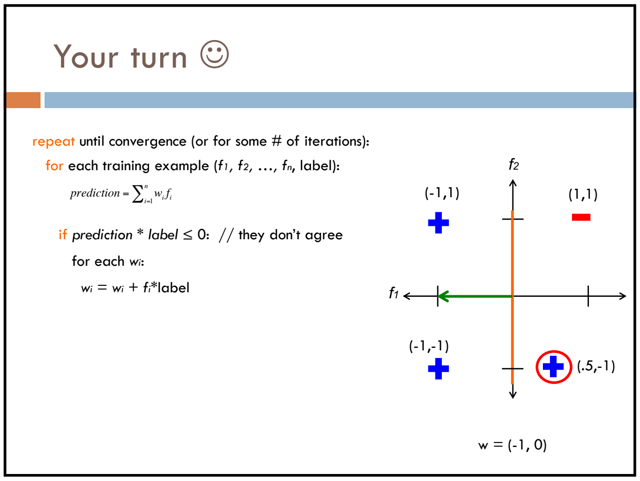


Your turn ☺

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1, f_2, …, f_n$, label):
    $prediction = \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
```

w = (0, -1)

82

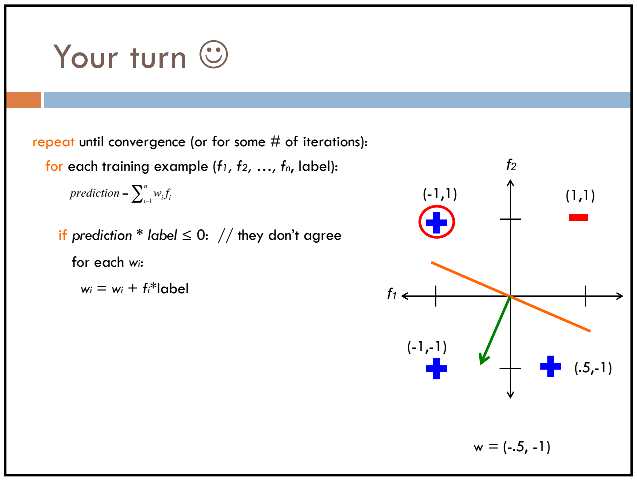


Your turn ☺

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1, f_2, …, f_n$, label):
    $prediction = \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
```

w = (-1, 0)

83

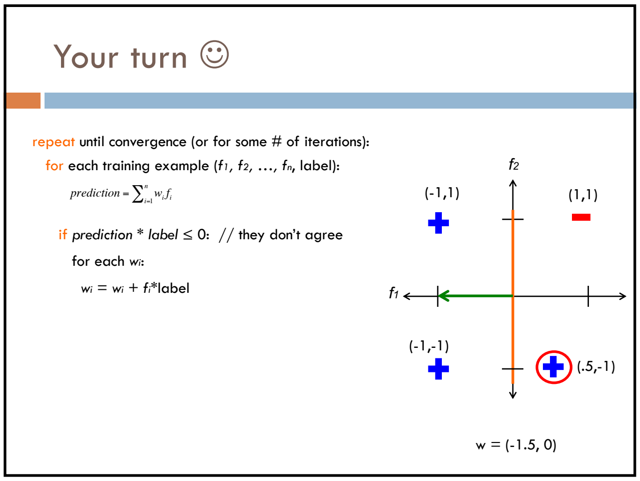


Your turn ☺

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1, f_2, …, f_n$, label):
    $prediction = \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
```

w = (-.5, -1)

84

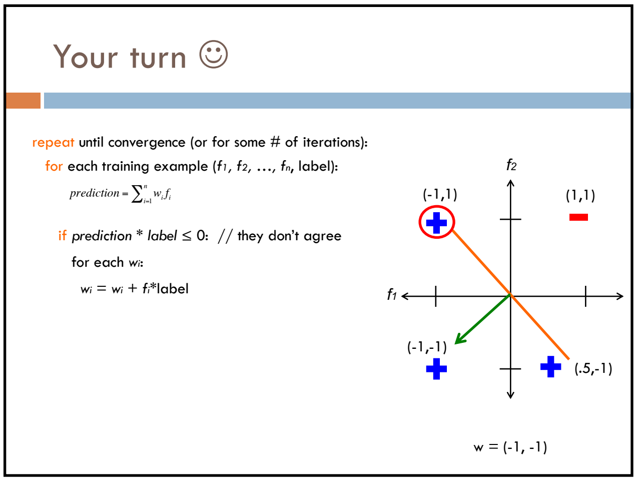


Your turn ☺

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    $prediction = \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
```

w = (-1.5, 0)

85

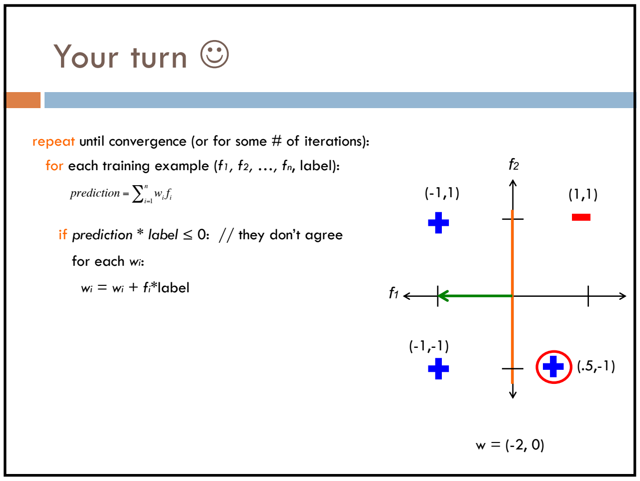


Your turn ☺

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    $prediction = \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
```

w = (-1, -1)

86

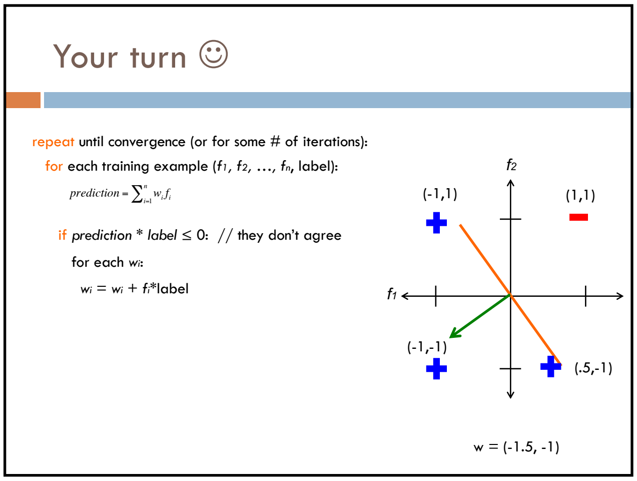


Your turn ☺

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    $prediction = \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
```

w = (-2, 0)

87



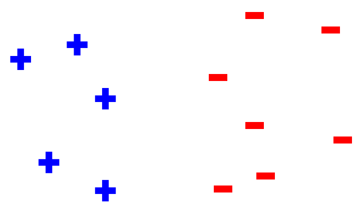
Your turn ☺

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    $prediction = \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
```

w = (-1.5, -1)

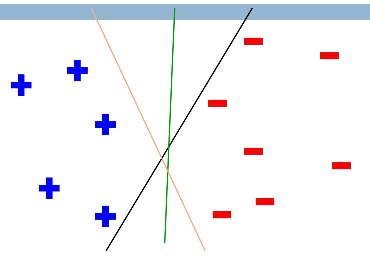
88

Which line will it find?



89

Which line will it find?



Only guaranteed to find ***some*** line that separates the data

90

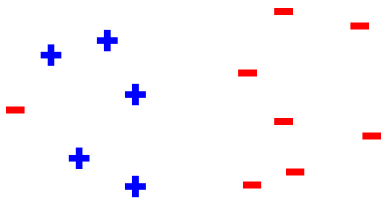
Convergence

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    $prediction = b + \sum_{i=1}^{n} w_i f_i$
    if prediction * label ≤ 0:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
      b = b + label
```

Why do we also have the "some # iterations" check?

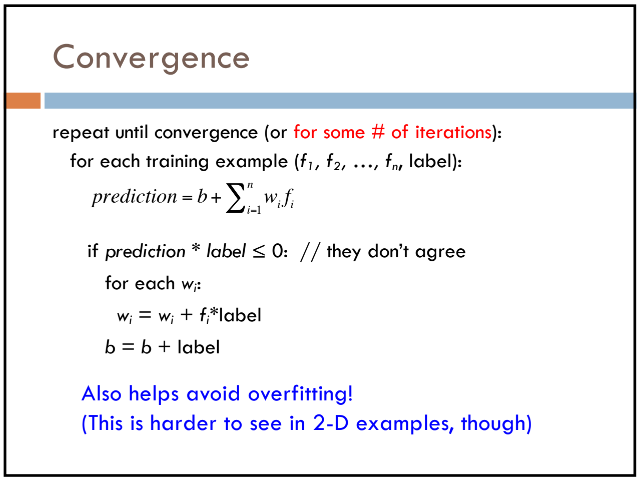
91

Handling non-separable data



If we ran the algorithm on this it would never converge!

92

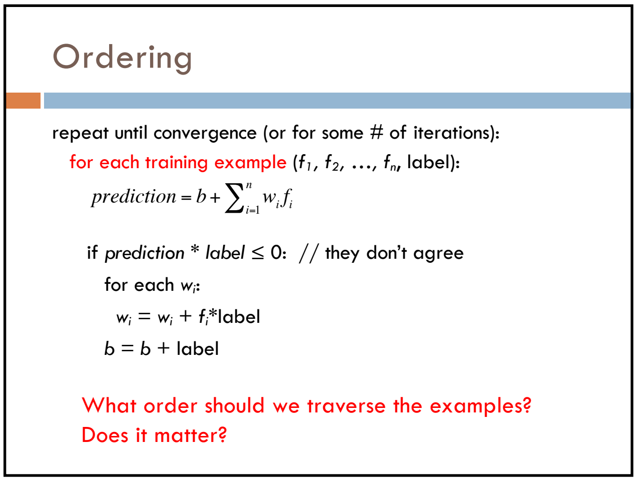


Convergence

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    $prediction = b + \sum_{i=1}^{n} w_i f_i$
    if $prediction$ * $label \leq 0$:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
      $b = b$ + label
```

Also helps avoid overfitting!
(This is harder to see in 2-D examples, though)

93

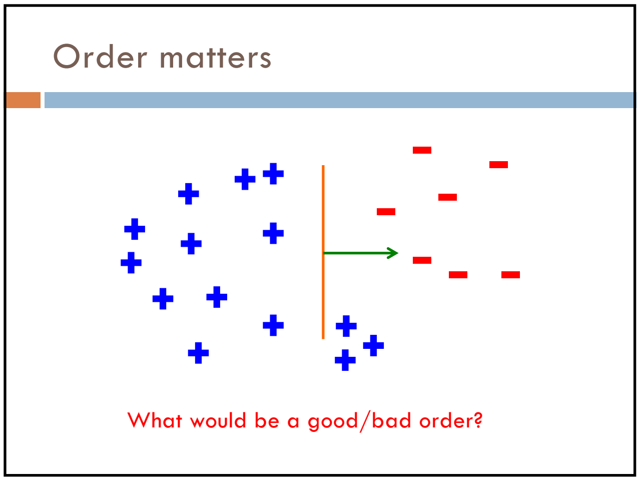


Ordering

```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    $prediction = b + \sum_{i=1}^{n} w_i f_i$
    if $prediction$ * $label \leq 0$:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
      $b = b$ + label
```

What order should we traverse the examples?
Does it matter?

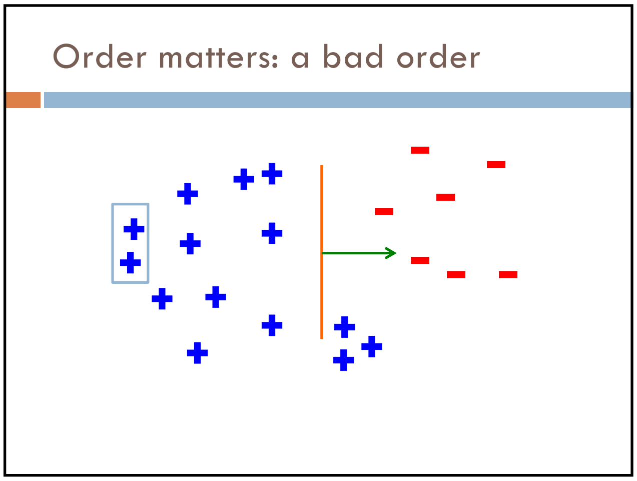
94



Order matters

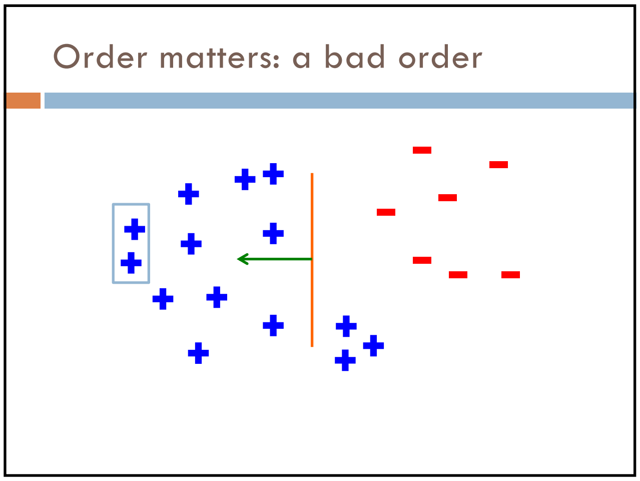
What would be a good/bad order?

95



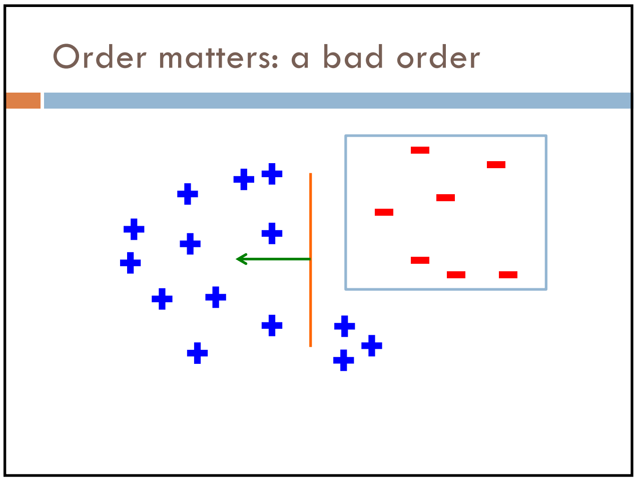
Order matters: a bad order

96



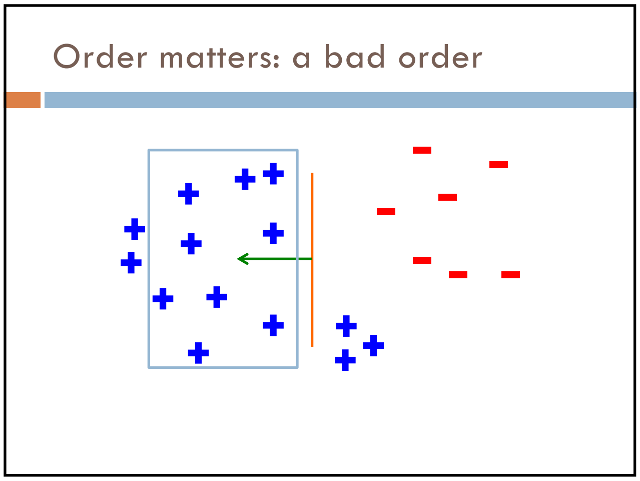
Order matters: a bad order

97



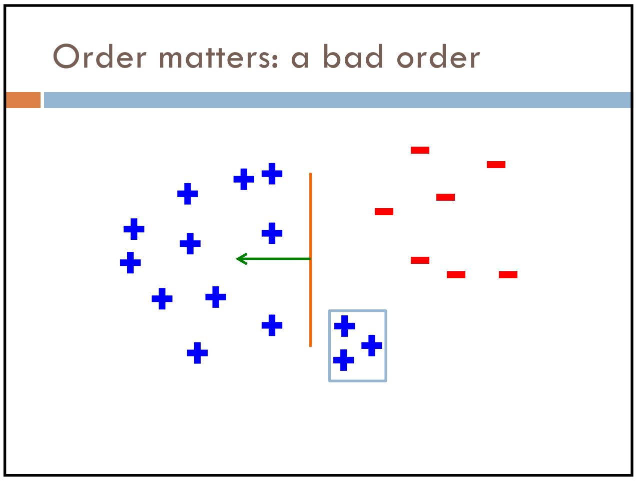
Order matters: a bad order

98



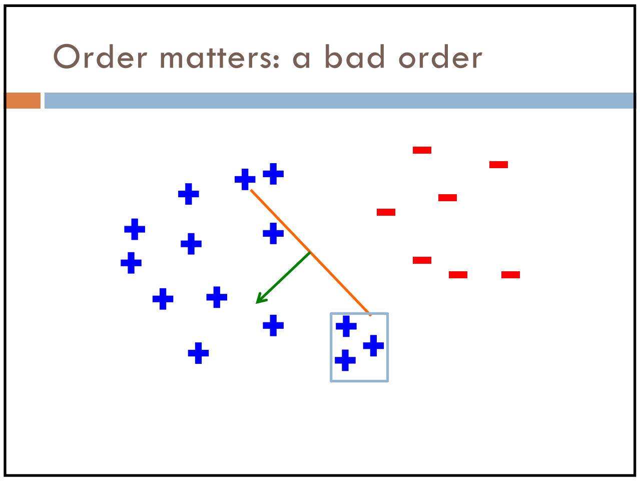
Order matters: a bad order

99



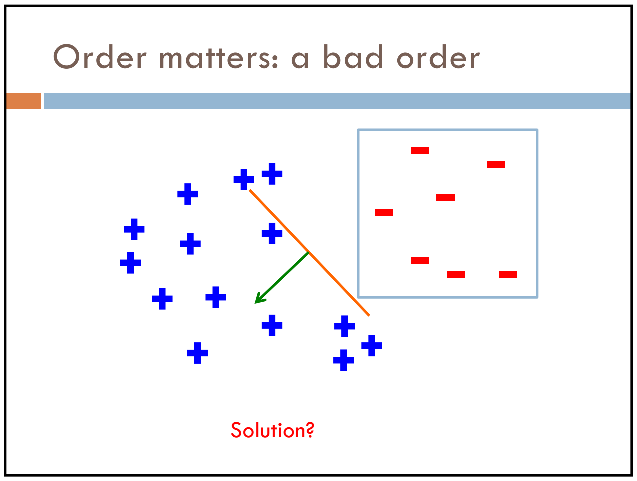
Order matters: a bad order

100



Order matters: a bad order

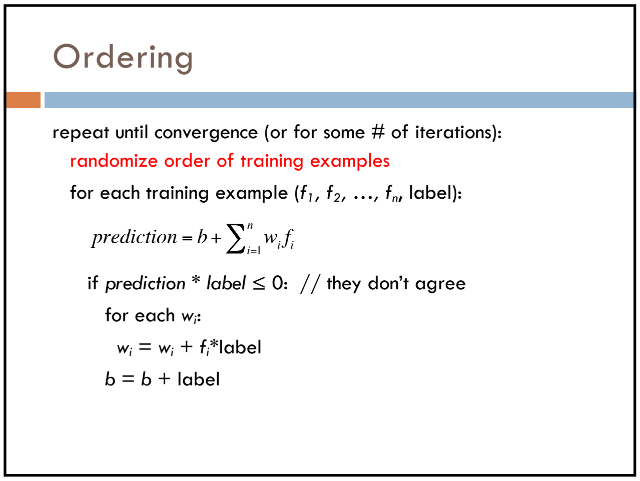
101



Order matters: a bad order

Solution?

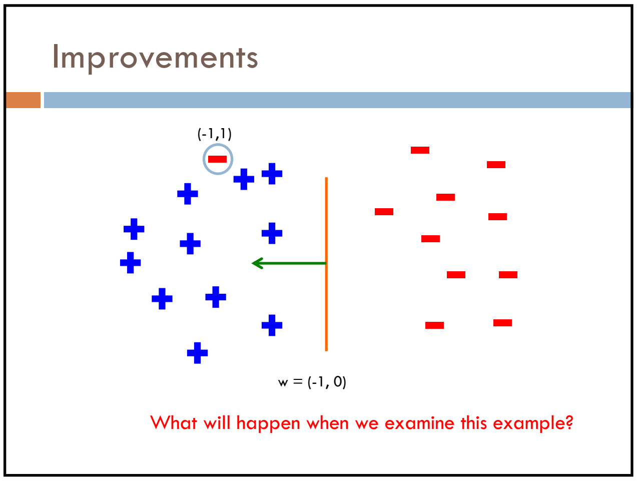
102



Ordering

```
repeat until convergence (or for some # of iterations):
   randomize order of training examples
   for each training example ($f_1$, $f_2$, …, $f_n$, label):
      $prediction = b + \sum_{i=1}^{n} w_i f_i$
      if prediction * label ≤ 0:  // they don't agree
         for each $w_i$:
            $w_i = w_i + f_i$*label
         b = b + label
```

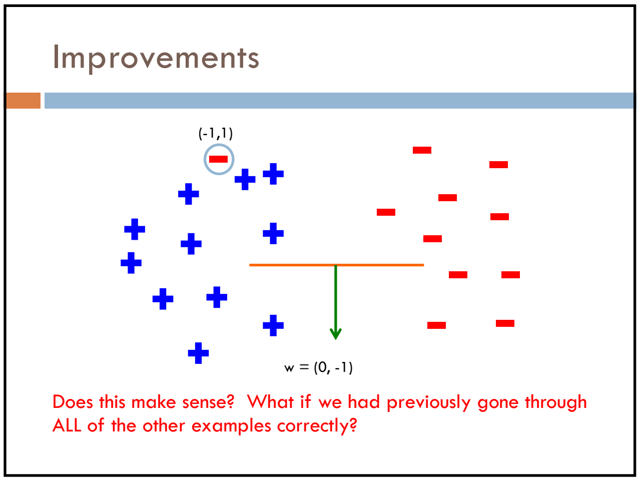
103



Improvements

What will happen when we examine this example?

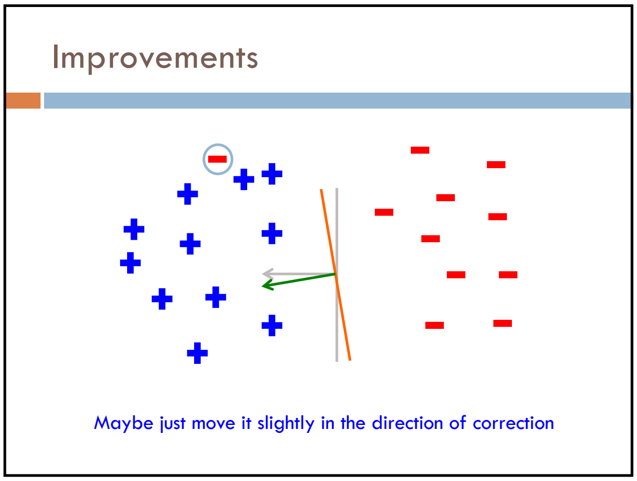
104



Improvements

Does this make sense? What if we had previously gone through ALL of the other examples correctly?

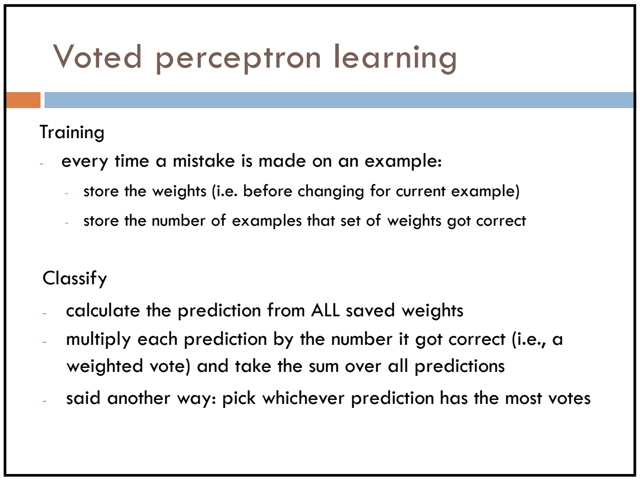
105



Improvements

Maybe just move it slightly in the direction of correction

106



Voted perceptron learning

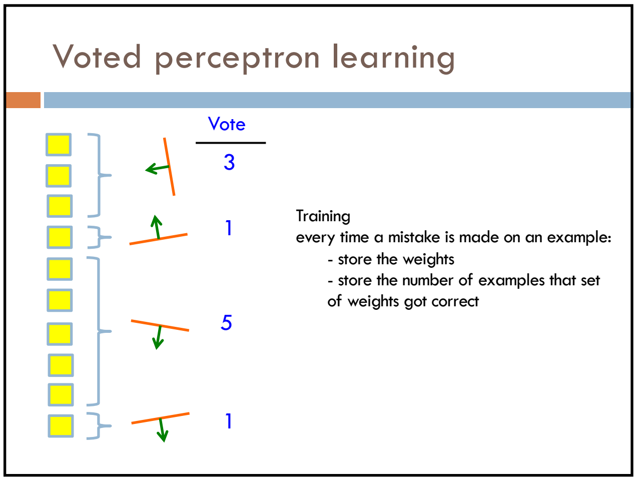
Training

- every time a mistake is made on an example:
  - store the weights (i.e. before changing for current example)
  - store the number of examples that set of weights got correct

Classify

- calculate the prediction from ALL saved weights
- multiply each prediction by the number it got correct (i.e., a weighted vote) and take the sum over all predictions
- said another way: pick whichever prediction has the most votes

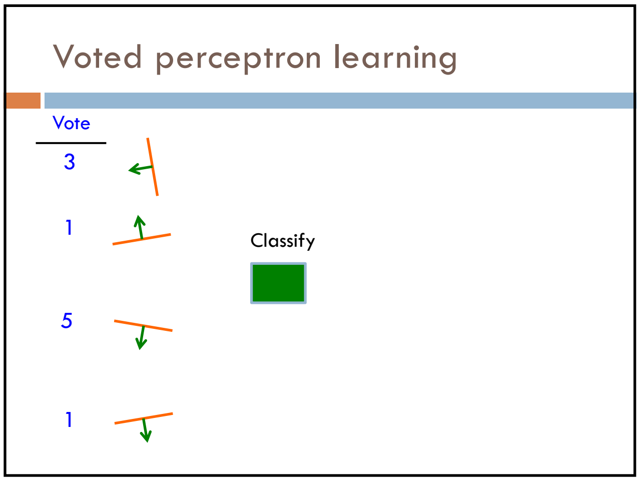
107



Voted perceptron learning

Training
every time a mistake is made on an example:
- store the weights
- store the number of examples that set of weights got correct

108



Voted perceptron learning

Vote

3

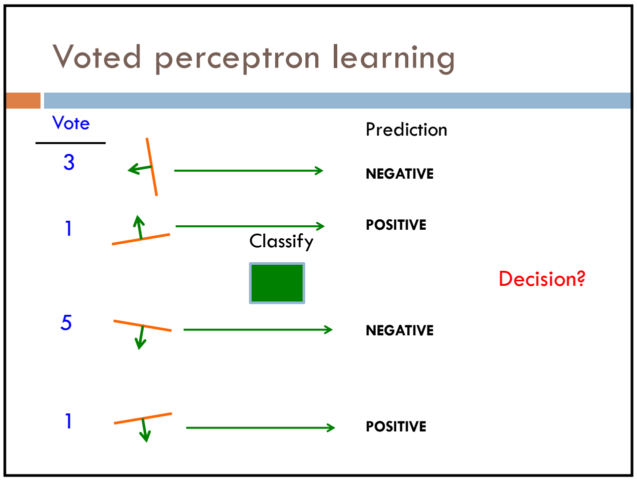
1

Classify

5

1

109



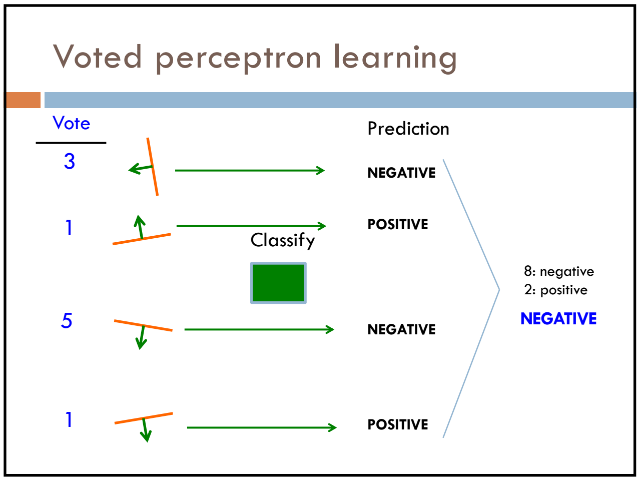
Voted perceptron learning

Vote	Prediction
3	NEGATIVE
1	POSITIVE
5	NEGATIVE
1	POSITIVE

Classify

Decision?

110



Voted perceptron learning

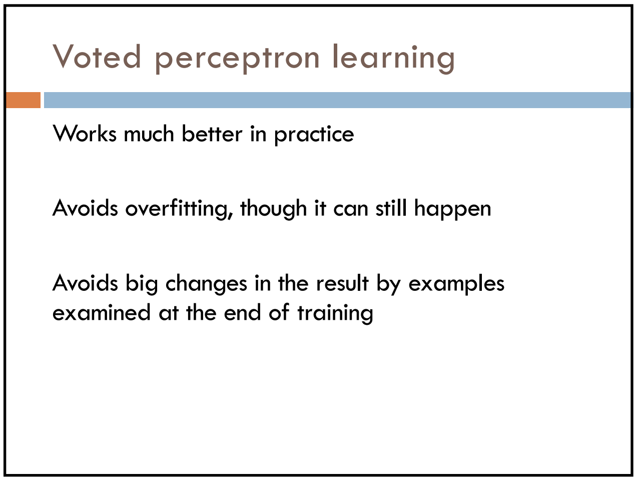
Vote	Prediction
3	NEGATIVE
1	POSITIVE
5	NEGATIVE
1	POSITIVE

Classify

8: negative
2: positive

NEGATIVE

111



Voted perceptron learning

Works much better in practice

Avoids overfitting, though it can still happen

Avoids big changes in the result by examples examined at the end of training

112

Voted perceptron learning

Training

- every time a mistake is made on an example:
  - store the weights (i.e. before changing for current example)
  - store the number of examples that set of weights got correct

Classify

- calculate the prediction from ALL saved weights
- multiply each prediction by the number it got correct (i.e a weighted vote) and take the sum over all predictions
- said another way: pick whichever prediction has the most votes

Any issues/concerns?

113

Voted perceptron learning

Training

- every time a mistake is made on an example:
  - store the weights (i.e. before changing for current example)
  - store the number of examples that set of weights got correct

Classify

- calculate the prediction from ALL saved weights
- multiply each prediction by the number it got correct (i.e a weighted vote) and take the sum over all predictions
- said another way: pick whichever prediction has the most votes

1. Can require a lot of storage
2. Classifying becomes very, very expensive

114

Average perceptron

Vote	
3	$w_1^1, w_2^1, \ldots, w_n^1, b^1$
1	$w_1^2, w_2^2, \ldots, w_n^2, b^2$
5	$w_1^3, w_2^3, \ldots, w_n^3, b^3$
1	$w_1^4, w_2^4, \ldots, w_n^4, b^4$

$$\overline{w_i} = \frac{3w_i^1 + 1w_i^2 + 5w_i^3 + 1w_i^4}{10}$$

The final weights are the *weighted average* of the previous weights

How does this help us?

115

Average perceptron

Vote	
3	$w_1^1, w_2^1, \ldots, w_n^1, b^1$
1	$w_1^2, w_2^2, \ldots, w_n^2, b^2$
5	$w_1^3, w_2^3, \ldots, w_n^3, b^3$
1	$w_1^4, w_2^4, \ldots, w_n^4, b^4$

$$\overline{w_i} = \frac{3w_i^1 + 1w_i^2 + 5w_i^3 + 1w_i^4}{10}$$

The final weights are the *weighted average* of the previous weights

Can just keep a running average!

116

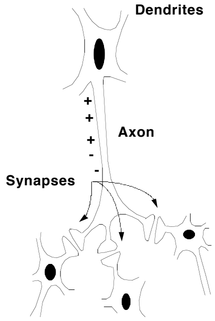
Perceptron learning algorithm

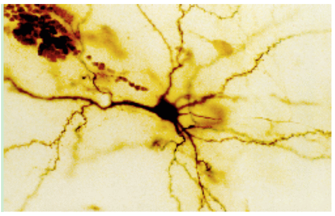
```
repeat until convergence (or for some # of iterations):
  for each training example ($f_1$, $f_2$, …, $f_n$, label):
    $prediction = b + \sum_{i=1}^{n} w_i f_i$

    if $prediction * label \le 0$:  // they don't agree
      for each $w_i$:
        $w_i = w_i + f_i$*label
      b = b + label
```

Why is it called the "perceptron" learning algorithm if what it learns is a line? Why not "line learning" algorithm?

117

Our Nervous System

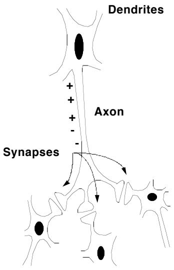




Neuron

118

Our nervous system: *the computer science view*

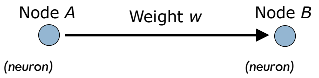


the human brain is a large collection of interconnected neurons

a NEURON is a brain cell

- collect, process, and disseminate electrical signals
- Neurons are connected via synapses
- They FIRE depending on the conditions of the neighboring neurons

119



w is the strength of signal sent between A and B.

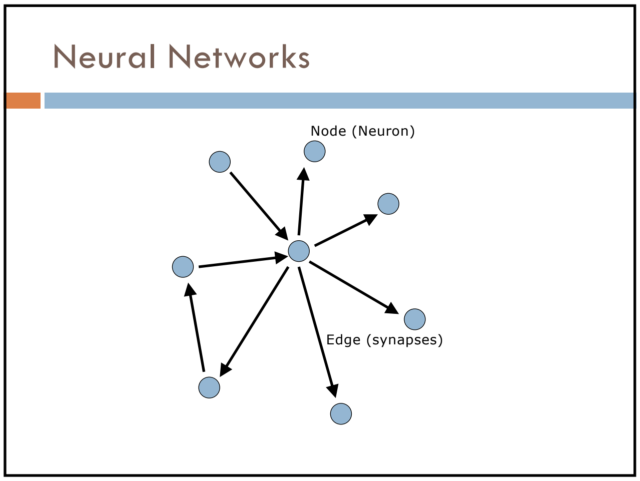
If *A* fires and *w* is **positive**, then *A* **stimulates** *B*.

If *A fires* and *w* is **negative**, then *A* **inhibits** *B*.

If a node is stimulated enough, then it also fires.

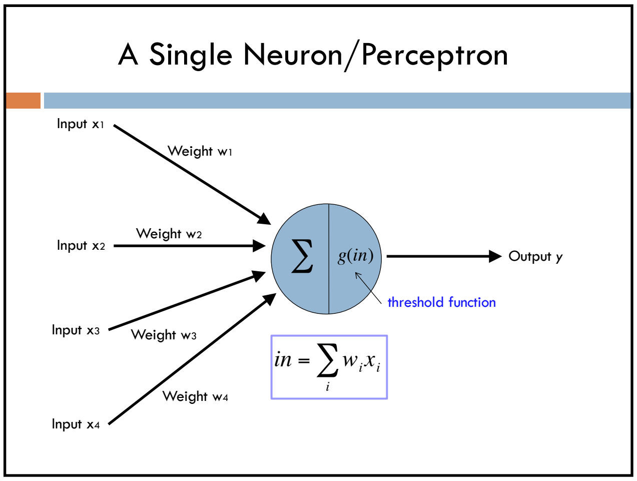
How much stimulation is required is determined by its **threshold**.

120



Neural Networks

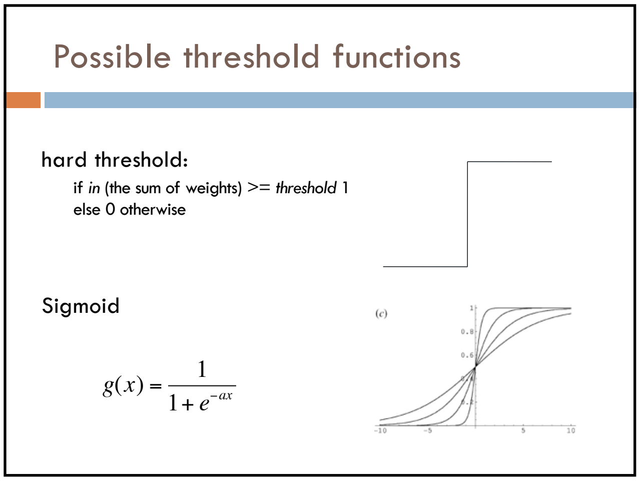
121



A Single Neuron/Perceptron

$$in = \sum_i w_i x_i$$

122



Possible threshold functions

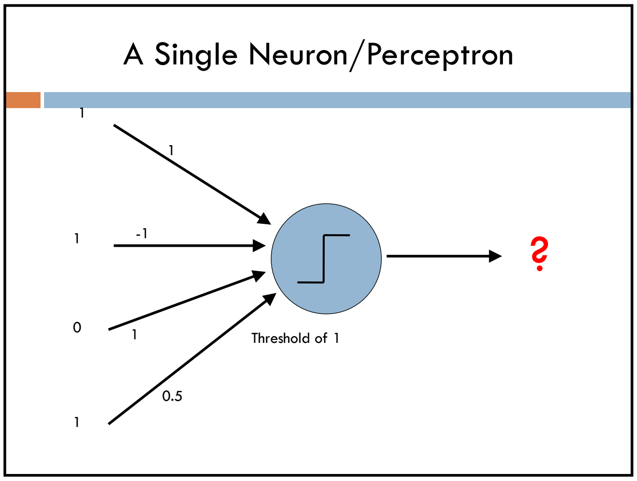
hard threshold:

if *in* (the sum of weights) >= *threshold* 1
else 0 otherwise

Sigmoid

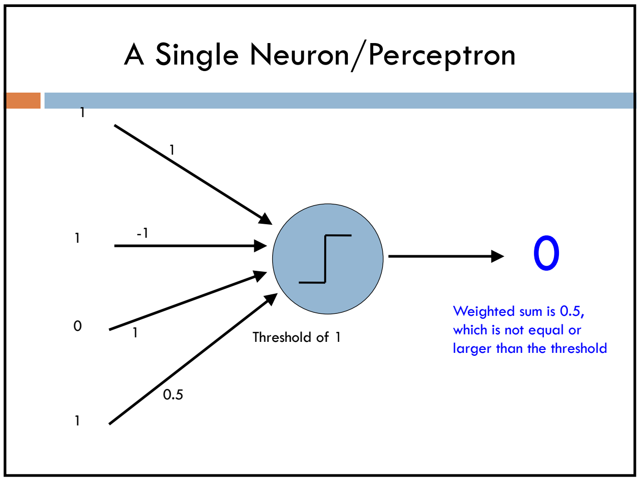
$$g(x) = \frac{1}{1+e^{-ax}}$$

123



A Single Neuron/Perceptron

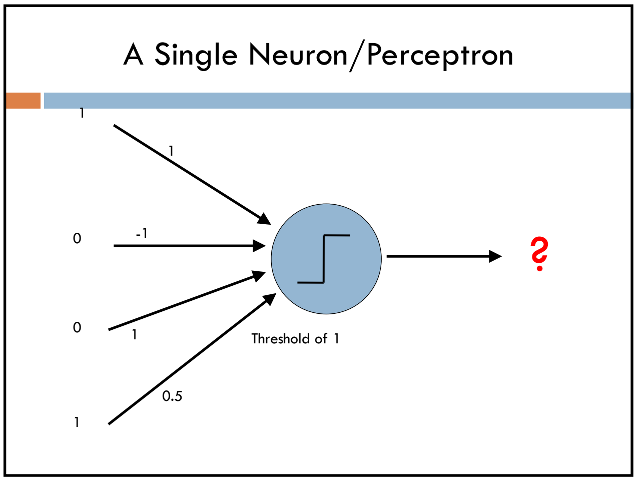
124



A Single Neuron/Perceptron

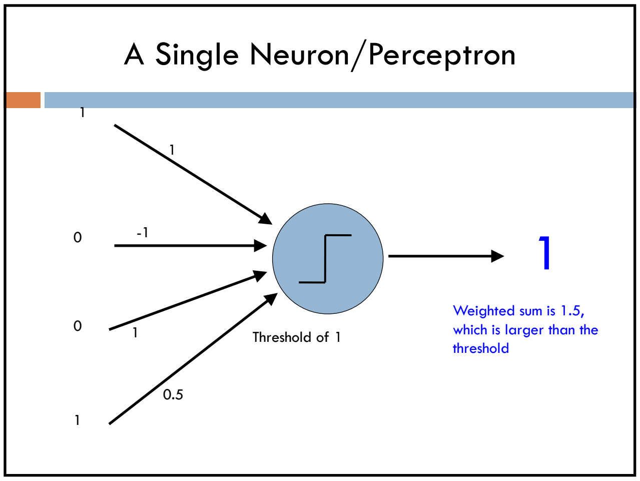
Weighted sum is 0.5, which is not equal or larger than the threshold

125



A Single Neuron/Perceptron

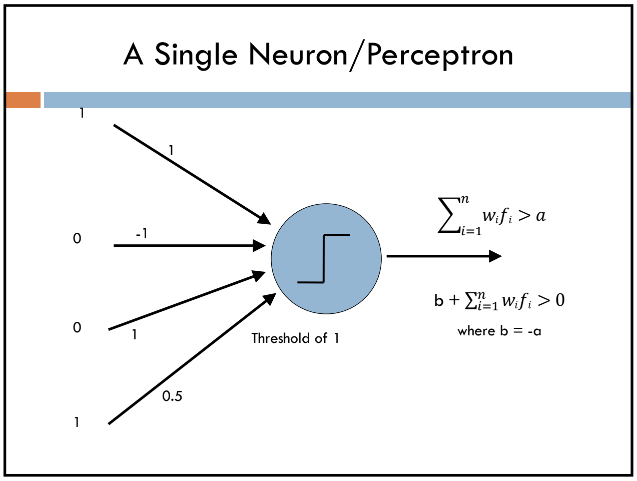
126



A Single Neuron/Perceptron

Weighted sum is 1.5, which is larger than the threshold

127



A Single Neuron/Perceptron

$\sum_{i=1}^{n} w_i f_i > a$

$b + \sum_{i=1}^{n} w_i f_i > 0$

where b = -a

128