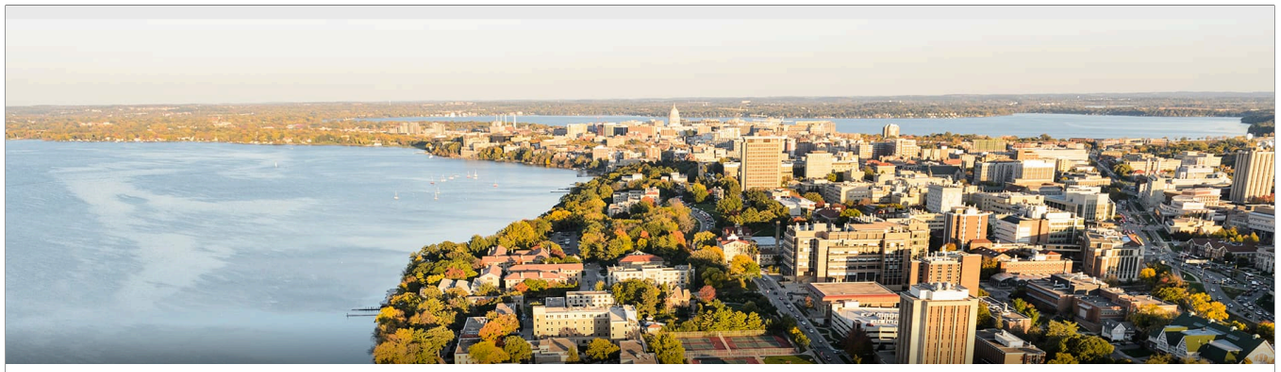


CS 540 Introduction to Artificial Intelligence

Perceptron

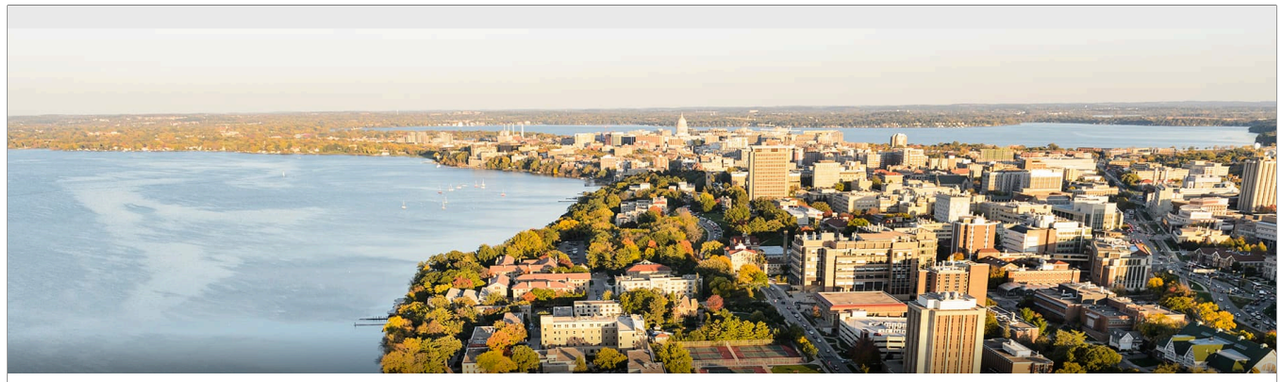
Yingyu Liang
University of Wisconsin-Madison

Oct 19, 2021

Slides created by Sharon Li [modified by Yingyu Liang]

Today's outline

- Naive Bayes (cont.)
- Single-layer Neural Network (Perceptron)



Part I: Naïve Bayes (cont.)

Recall the stages:

1. Formulate the decision making (ie, discrete prediction label for y) into a conditional probability problem p(y|x): the distribution over all possible labels given x.
2. Apply Bayes' rule, turn the problem into maximizing the product of class conditionals p(x|y) and class priors p(y)
3. Use the training data to estimate p(x|y) and p(y), and plug in the Bayes' rule to make the prediction.

Naive Bayes is using Naive Bayes assumption on p(x|y), to get p(x|y) = \prod_i p(x_i|y) where x_i is the i-th feature of the input x. Then use MLE to estimate p(x_i|y) and p(y). For discrete x, MLE is essentially counting.

Example 1: Play outside or not?

- If weather is sunny, would you likely to play outside?

Posterior probability p(Yes | ☀️) vs. p(No | ☀️)

Stage 1: formulate the decision making problem (play outside or not) into a conditional probability problem: p(Play|sunny) for two labels Play=Yes and Play=No.

Example 1: Play outside or not?

- If weather is sunny, would you likely to play outside?

Posterior probability p(Yes | ☀️) vs. p(No | ☀️)

- Weather = {Sunny, Rainy, Overcast}
- Play = {Yes, No}
- Observed data {Weather, play on day *m*}, m={1,2,…,N}

Example 1: Play outside or not?

- If weather is sunny, would you likely to play outside?

Posterior probability p(Yes | ☀) vs. p(No | ☀)

- Weather = {Sunny, Rainy, Overcast}
- Play = {Yes, No}
- Observed data {Weather, play on day *m*}, m={1,2,…,N}

$$p(\text{Play} \mid ☀) = \frac{p(☀ \mid \text{Play})\, p(\text{Play})}{p(☀)}$$

Bayes rule

Stage 2: apply Bayes rule. Need to estimate the terms p(sunny|Play) and p(Play).

Example 1: Play outside or not?

- **Step 1**: Convert the data to a frequency table of Weather and Play

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

Stage 3: MLE to estimate the terms. Essentially counting (we have proved this for Bernoulli distribution in the coin-flipping example; a similar proof holds for the multinomial distribution.) Then plug in Bayes' rule to make the prediction

Example 1: Play outside or not?

Step 1: Convert the data to a frequency table of Weather and Play

Step 2: Based on the frequency table, calculate **likelihoods** and **priors**

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

p(Play = Yes) = 0.64

p(☀ | Yes) = 3/9 = 0.33

https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

Example 1: Play outside or not?

Step 3: Based on the likelihoods and priors, calculate posteriors

P(Yes| ☀)
=P(☀ |Yes)*P(Yes)/P(☀) **?**

P(No| ☀)
=P(☀ |No)*P(No)/P(☀) **?**

Example 1: Play outside or not?

Step 3: Based on the likelihoods and priors, calculate posteriors

P(Yes| ☀)
=P(☀ |Yes)*P(Yes)/P(☀)
=0.33*0.64/0.36
=0.6

P(No| ☀)
=P(☀ |No)*P(No)/P(☀)
=0.4*0.36/0.36
=0.4

P(Yes| ☀) > P(No| ☀) go outside and play!

Bayesian classification

$$\hat{y} = \arg\max p(y \mid \mathbf{x})$$

(Posterior)

(Prediction)

$$= \arg\max \frac{p(\mathbf{x} \mid y) \cdot p(y)}{p(\mathbf{x})}$$

(by Bayes' rule)

$$= \arg\max \; p(\mathbf{x} \mid y) p(y)$$

Bayesian classification

What if **x** has multiple attributes $\mathbf{x} = \{X_1, \ldots, X_k\}$

$$\hat{y} = \arg\max_y p(y \mid X_1, \ldots, X_k)$$

(Prediction) (Posterior)

Bayesian classification

What if **x** has multiple attributes $\mathbf{x} = \{X_1, \ldots, X_k\}$

$$\hat{y} = \arg\max_y p(y \mid X_1, \ldots, X_k)$$

(Prediction) (Posterior)

$$= \arg\max_y \frac{p(X_1, \ldots, X_k \mid y) \cdot p(y)}{p(X_1, \ldots, X_k)}$$

(by Bayes' rule)

Independent of y

Bayesian classification

What if **x** has multiple attributes $\mathbf{x} = \{X_1, \ldots, X_k\}$

$$\hat{y} = \arg\max_y p(y \mid X_1, \ldots, X_k)$$ (Posterior)

(Prediction)

$$= \arg\max_y \frac{p(X_1, \ldots, X_k \mid y) \cdot p(y)}{p(X_1, \ldots, X_k)}$$ (by Bayes' rule)

$$= \arg\max_y \; p(X_1, \ldots, X_k \mid y)\; p(y)$$

Class conditional likelihood

Class prior

Recall the stages:

1. Formulate the decision making (ie, discrete prediction label for y) into a conditional probability problem p(y|x): the distribution over all possible labels given x.
2. Apply Bayes' rule, turn the problem into maximizing the product of class conditionals p(x|y) and class priors p(y)
3. Use the training data to estimate p(x|y) and p(y), and plug in the Bayes' rule to make the prediction.

Naïve Bayes Assumption

Conditional independence of feature attributes

$$p(X_1, \ldots, X_k \mid y)p(y) = \Pi_{i=1}^{k} p(X_i \mid y)p(y)$$

Easier to estimate

(using MLE!)

Naive Bayes is using Naive Bayes (ie conditional indolence) assumption on p(x|y), to get p(x|y) = \prod_i p(x_i|y) where x_i is the i-th feature of the input x. Then use MLE to estimate each p(x_i|y) and p(y). For discrete x, MLE is essentially counting.

Quiz break

Q3-1: Which of the following about Naive Bayes is incorrect?

- A Attributes can be nominal or numeric
- B Attributes are equally important
- C Attributes are statistically dependent of one another given the class value
- D Attributes are statistically independent of one another given the class value
- E All of above

Quiz break

Q3-1: Which of the following about Naive Bayes is incorrect?

- A Attributes can be nominal or numeric
- B Attributes are equally important
- C Attributes are statistically dependent of one another given the class value
- D Attributes are statistically independent of one another given the class value
- E All of above

Naive Bayes assumption: Attributes are statistically **independent** of one another given the class value.

Quiz break

Q3-2: Consider a classification problem with two binary features, x_1, $x_2 \in \{0,1\}$, and $y \in \{1,2,\ldots,32\}$. Suppose $P(Y = y) = 1/32$, $P(x_1 = 1 | Y = y) = y/46$, $P(x_2 = 1 | Y = y) = y/62$. Which class will naive Bayes classifier produce on a test item with $x_1 = 1$ and $x_2 = 0$?

- A 16
- B 26
- C 31
- D 32

Quiz break

Q3-2: Consider a classification problem with two binary features, x_1, $x_2 \in \{0,1\}$, and $y \in \{1,2,\ldots,32\}$. Suppose $P(Y = y) = 1/32$, $P(x_1 = 1 | Y = y) = y/46$, $P(x_2 = 1 | Y = y) = y/62$. Which class will naive Bayes classifier produce on a test item with $x_1 = 1$ and $x_2 = 0$?

- A 16
- B 26
- C 31
- D 32

Stage 1: need to estimate P(Y=y|x1=1, x2=0) for different y's.

Stage 2: Apply Bayes' rule and get
Prediction= \argmax_y p(x1=1,x2=0|Y=y)P(Y=y)

Stage 3: estimate the terms and plug in the Bayes' rule to make the prediction.
Apply Naive Bayes assumption:
p(x1=1,x2=0|Y=y)=p(x1=1|Y=y) p(x2=0|Y=y)
Then we have:
Prediction= \argmax_y p(x1=1|Y=y) p(x2=0|Y=y) p(Y=y)
= \argmax_y y/46 * (1-y/62) * 1/32
= \argmax_y y * (62-y)
= 31

Quiz break

Q3-3: Consider the following dataset showing the result whether a person has passed or failed the exam based on various factors. Suppose the factors are independent to each other. We want to classify a new instance with Confident=Yes, Studied=Yes, and Sick=No.

Confident	Studied	Sick	Result
Yes	No	No	Fail
Yes	No	Yes	Pass
No	Yes	Yes	Fail
No	Yes	No	Pass
Yes	Yes	Yes	Pass

- A Pass
- B Fail

Quiz break

Q3-3: Consider the following dataset showing the result whether a person has passed or failed the exam based on various factors. Suppose the factors are independent to each other. We want to classify a new instance with Confident=Yes, Studied=Yes, and Sick=No.

Confident	Studied	Sick	Result
Yes	No	No	Fail
Yes	No	Yes	Pass
No	Yes	Yes	Fail
No	Yes	No	Pass
Yes	Yes	Yes	Pass

- A Pass
- B Fail

Stage 1: need to estimate P(Y=y|Confident=Yes, Studied=Yes, Sick=No) for y in {Pass, Fail}.

Stage 2: Apply Bayes' rule and get
Prediction= \argmax_y p(Confident=Yes, Studied=Yes, Sick=No|Y=y)P(Y=y)

Stage 3: estimate the terms and plug in the Bayes' rule to make the prediction.

Apply Naive Bayes assumption:
p(Confident=Yes, Studied=Yes, Sick=No|Y=y)=p(Confident=Yes|Y=y) p(Studied=Yes|Y=y) p(Sick=No|Y=y)

Apply MLE on the training data (ie, counting):
1) For Y=Pass
p(Confident=Yes|Y=Pass) = 2/3,
p(Studied=Yes|Y=Pass) = 2/3,
p(Sick=No|Y=Pass) = 1/3,
p(Y=Pass) = 3/5

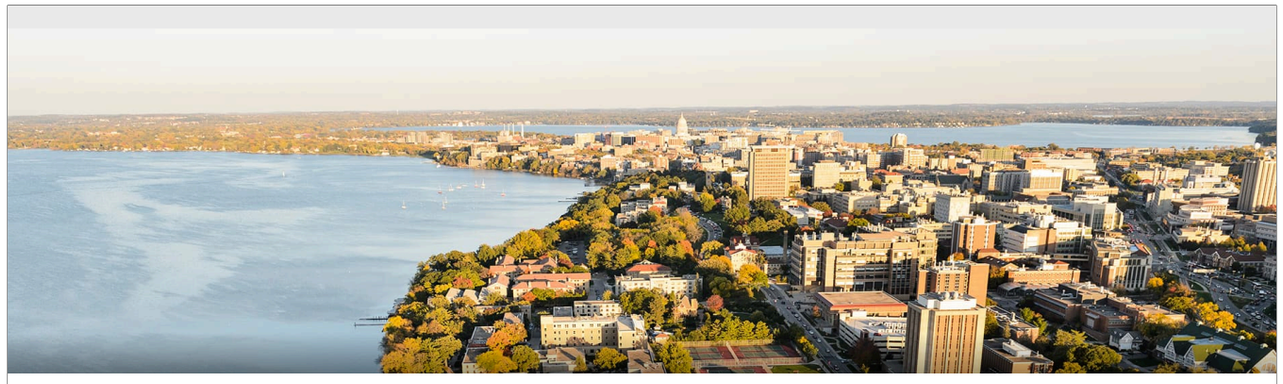
2) For Y=Fail
p(Confident=Yes|Y=Fail) = 1/2,
p(Studied=Yes|Y=Fail) = 1/2,

p(Sick=No|Y=Fail) = 1/2,
p(Y=Fail) = 2/5

Then we have:
p(Confident=Yes, Studied=Yes, Sick=No|Y=Pass)P(Y=Pass) = 2/3 * 2/3 * 1/3 * 3/5 = 4/9 * 1/5
p(Confident=Yes, Studied=Yes, Sick=No|Y=Fail)P(Y=Fail) = 1/2 * 1/2 * 1/2 * 2/5 = 1/4 * 1/5
The former is larger than the latter, so:
Prediction= \argmax_y p(Confident=Yes, Studied=Yes, Sick=No|Y=y)P(Y=y) = Pass

What we've learned today…

- K-Nearest Neighbors
- Maximum likelihood estimation
  - Bernoulli model
  - Gaussian model
- Naive Bayes
  - Conditional independence assumption



Part I: Single-layer Neural Networks

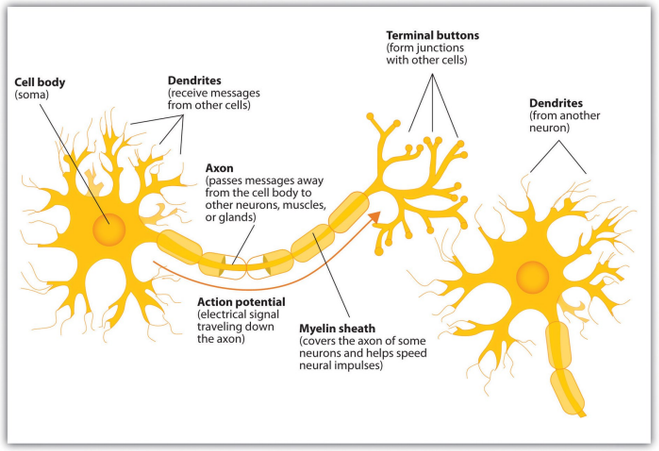
How to classify

Cats vs. dogs?

Inspiration from neuroscience

- Inspirations from human brains
- Networks of simple and homogenous units

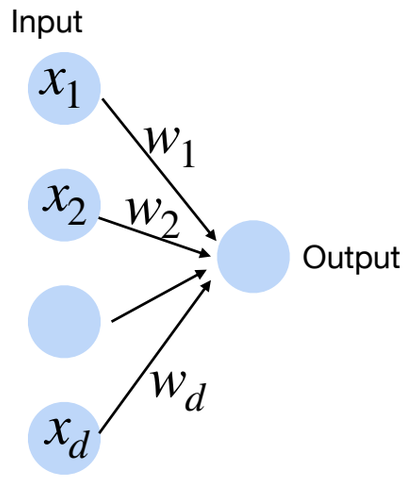
(wikipedia)



Artificial neural networks are inspired by human neural networks. Though we don’t know much about human brains, we do know that they are networks of simple and homogenous units called neurons. Human neurons can be divided into a few types, and neurons of the same type are very similar to each other; each neuron performs simple operations. These facts lead to the design of artificial neural networks.

Perceptron

Cats vs. dogs?

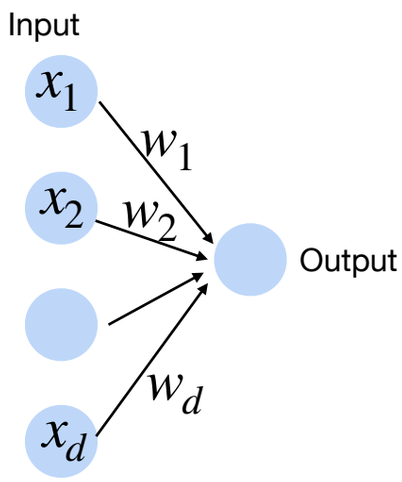


Linear Perceptron

- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b, perceptron outputs:

$$f = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

Cats vs. dogs?



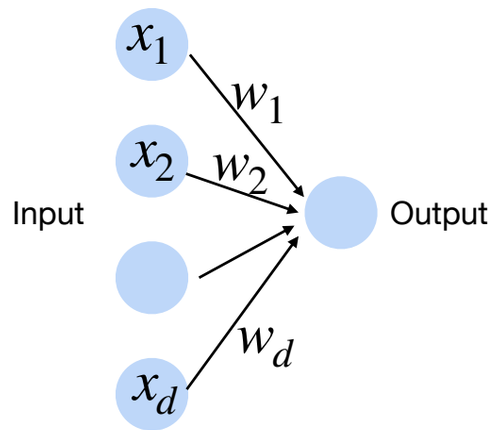
Perceptron

- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b, perceptron outputs:

$$o = \sigma\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \qquad \sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Activation function

Cats vs. dogs?



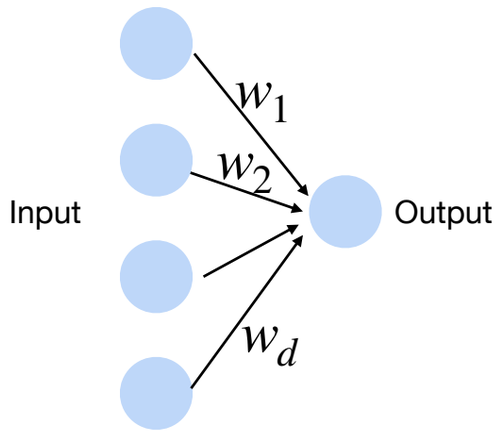
The standard perceptron is simply linear transformation followed by the activation function (the step function). The output 0 or 1 can be viewed as classes.

Using different activation functions leads to different variants of the standard perceptron. E.g., if we use the identity function \sigma(x) = x, then it's the linear perceptron (i.e., the linear function).

Perceptron

- Goal: learn parameters $\mathbf{w} = \{w_1, w_2, \ldots, w_d\}$ and b to minimize the classification error

Cats vs. dogs?



Training the Perceptron

Perceptron Algorithm

Initialize $\vec{w} = \vec{0}$ // Initialize $\vec{w}$. $\vec{w} = \vec{0}$ misclassifies everything.
while TRUE **do** // Keep looping
  $m = 0$ // Count the number of misclassifications, m
  for $(x_i, y_i) \in D$ **do** // Loop over each (data, label) pair in the dataset, D
    if $o_i \neq y_i$ **then** // If the pair $(\vec{x_i}, y_i)$ is misclassified
      $\vec{w} \leftarrow \vec{w} + x_i$ if $y_i = 1$, $\vec{w} \leftarrow \vec{w} - x_i$ if $y_i = 0$
      $m \leftarrow m + 1$ // Counter the number of misclassification
    end if
  end for
  if $m = 0$ **then** // If the most recent $\vec{w}$ gave 0 misclassifications
    break // Break out of the while-loop
  end if
end while // Otherwise, keep looping!

For simplicity, the weight vector and input vector are extended vectors (including the bias or the constant 1).

This perceptron was proposed in the early days of AI, which has inspired many other methods.

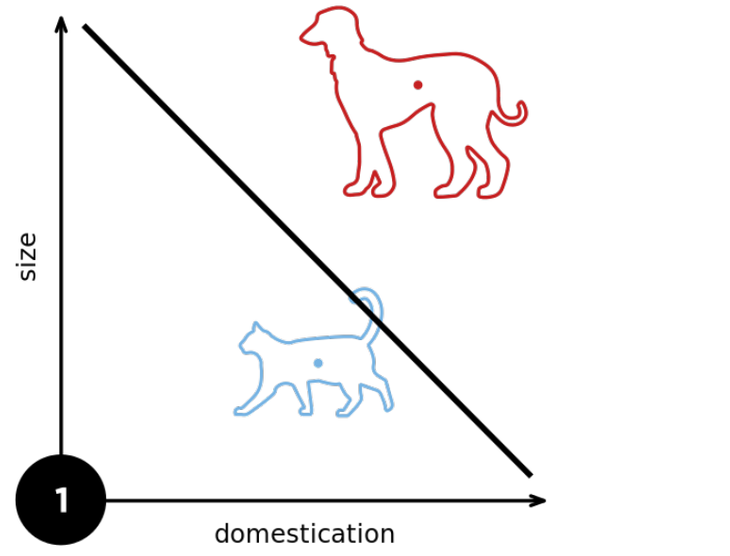
The method follows the intuition of correcting mistakes.
For simplicity of presentation, usually we concatenate the weight vector with the bias to get an extended weight vector \vec{w}. We also concatenate the input point x with a constant 1. In this weight, the linear transformation before the step activation function is simply the inner product between the extended weight vector and the extended input point.

The training algorithm goes over the training dataset; each pass is called an epoch. In an epoch, it goes over the data points one by one. For the current data point, it uses the current perceptron to predict. If no mistake, just do nothing; if there is a mistake then update the weight vector. In the case when the true class label is 1 but we predict class 0, then it means the linear transformation is too small, we should increase that, and we do so by increasing the weight vector by adding the extended input point. In the other case when the true class label is 0 but we predict class 1, then it means the linear transformation is too large, we should decrease that, and we do so by decreasing the weight vector by subtracting the extended input point.

If in an epoch we make no mistakes, it means we cannot update the weight vector anymore. The method just stop.

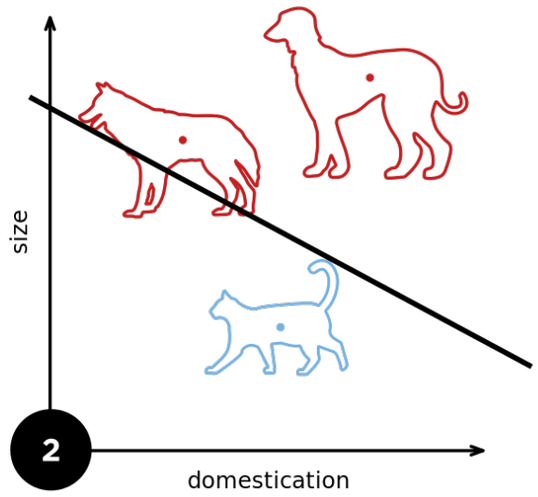
The method can succeed when there is indeed a ground truth linear hyperplane that can separates the two classes in the training data. If the two classes in the training data are not linearly separable, then the method will loop for ever since there is always some mistake.

Perceptron



From wikipedia

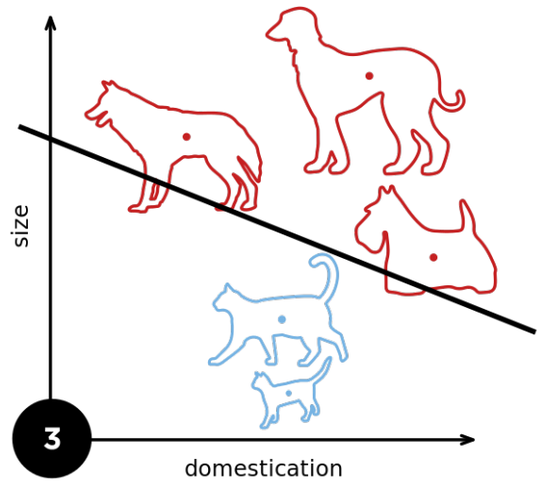
Perceptron



From wikipedia

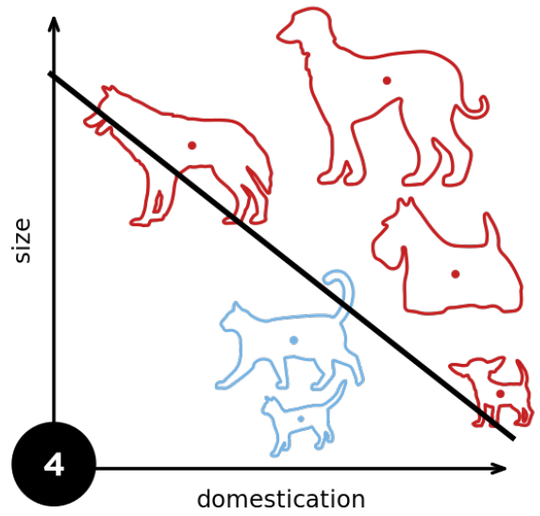
Made mistake on the third data points, so update the model to correct the mistake.

Perceptron



From wikipedia

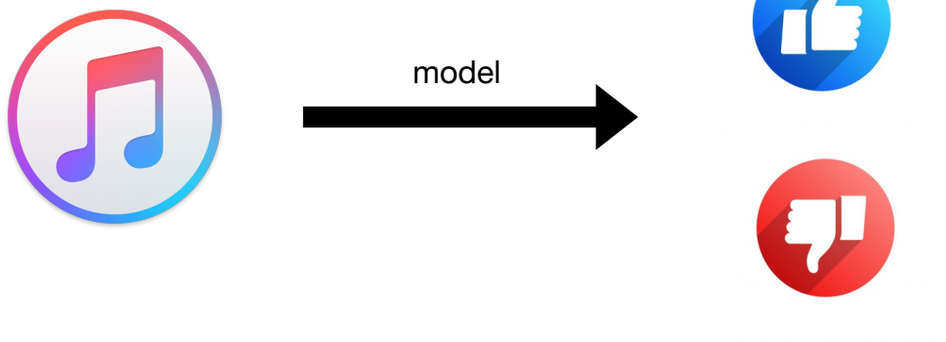
Perceptron



From wikipedia

Keep correcting the mistakes to arrive at a final solution.

Example 2: Predict whether a user likes a song or not

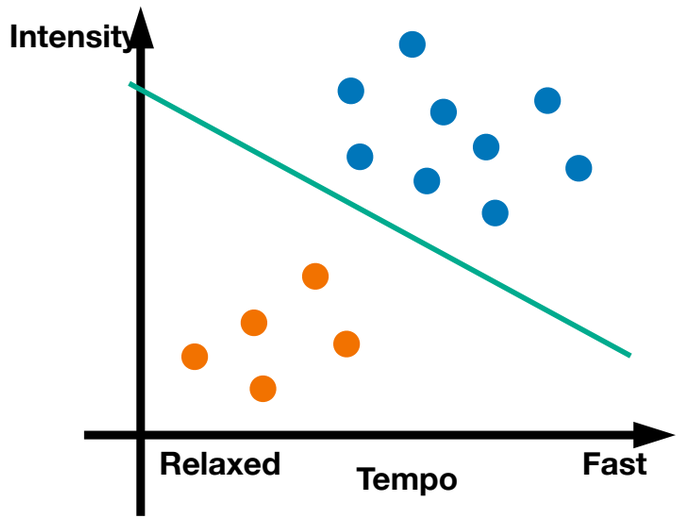


Example 2: Predict whether a user likes a song or not Using Perceptron

User Sharon

DisLike

Like



Learning AND function using perceptron

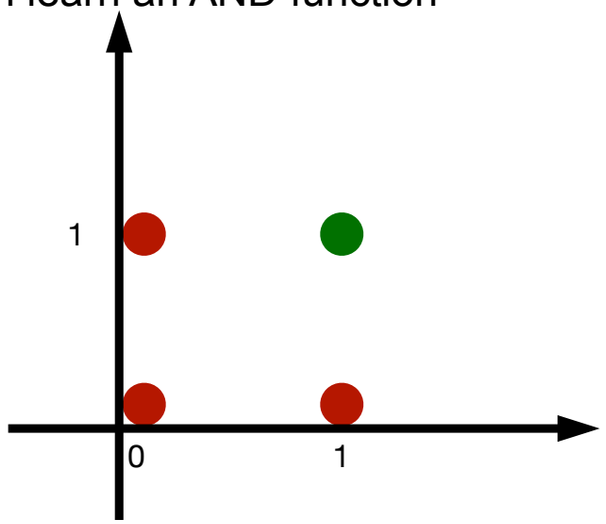
The perceptron can learn an AND function

$x_1 = 1, x_2 = 1, y = 1$

$x_1 = 1, x_2 = 0, y = 0$

$x_1 = 0, x_2 = 1, y = 0$

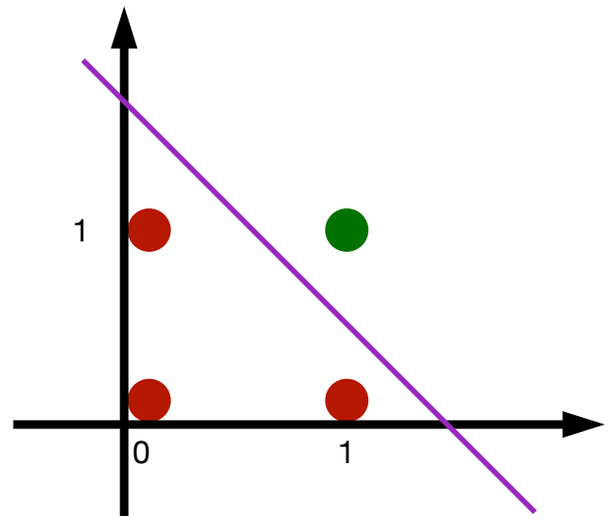
$x_1 = 0, x_2 = 0, y = 0$



The AND function has a linear decision boundary

Learning AND function using perceptron

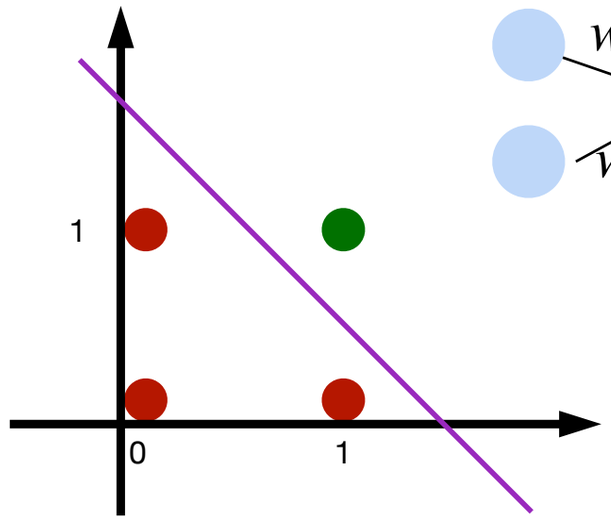
The perceptron can learn an AND function

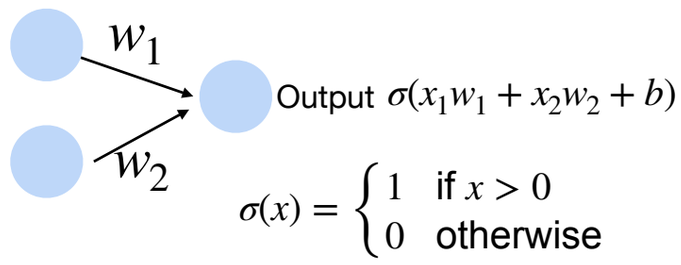


The linear decision boundary can be realized by a perceptron.

Learning AND function using perceptron

The perceptron can learn an AND function



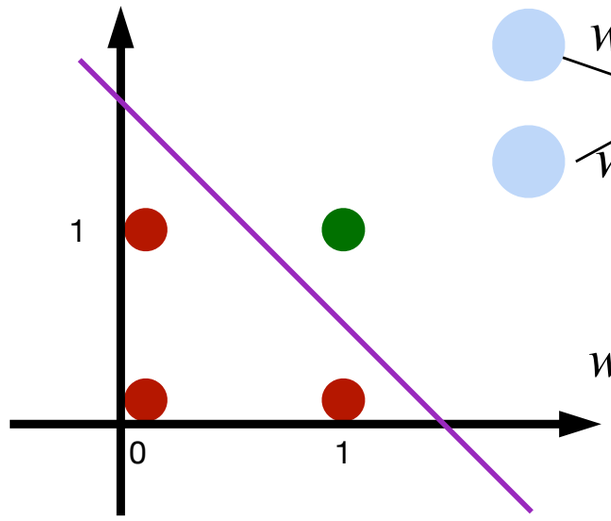


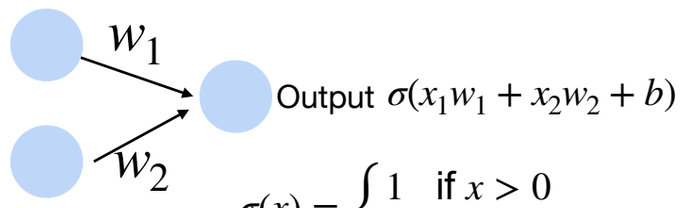
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

What's w and b?

Learning AND function using perceptron

The perceptron can learn an AND function





$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 1, w_2 = 1, b = -1.5$$

Learning OR function using perceptron

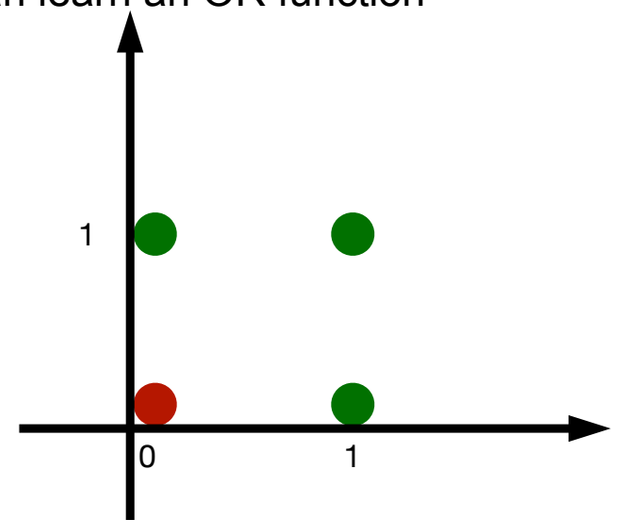
The perceptron can learn an OR function

$x_1 = 1, x_2 = 1, y = 1$

$x_1 = 1, x_2 = 0, y = 1$

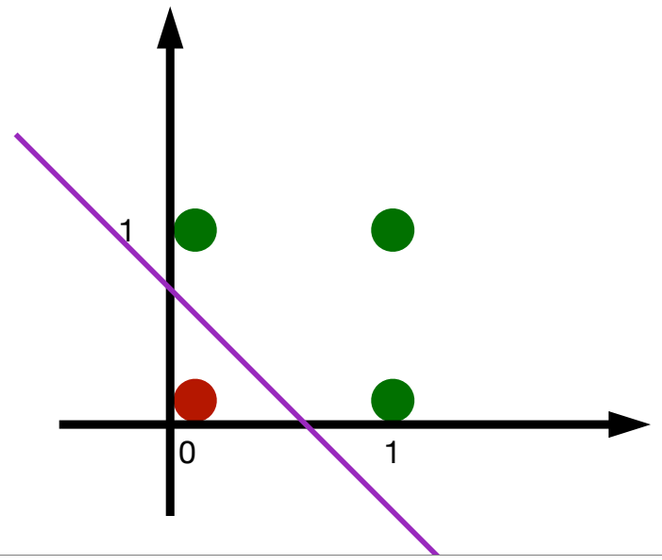
$x_1 = 0, x_2 = 1, y = 1$

$x_1 = 0, x_2 = 0, y = 0$



Learning OR function using perceptron

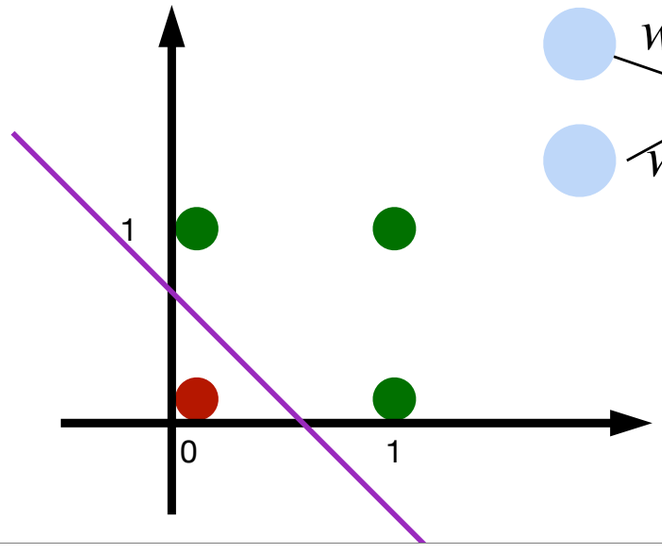
The perceptron can learn an OR function

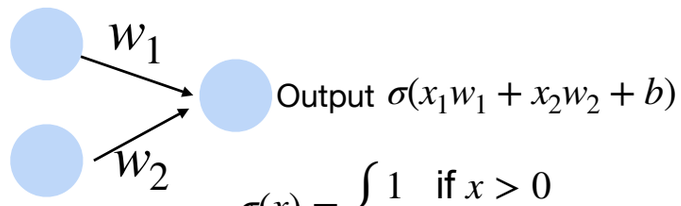


Similarly, the OR function has a linear decision boundary, which can be realized by a perceptron.

Learning OR function using perceptron

The perceptron can learn an OR function



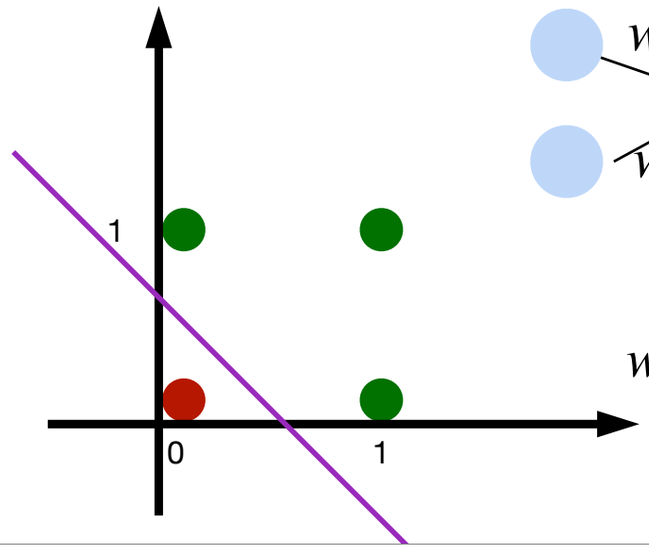


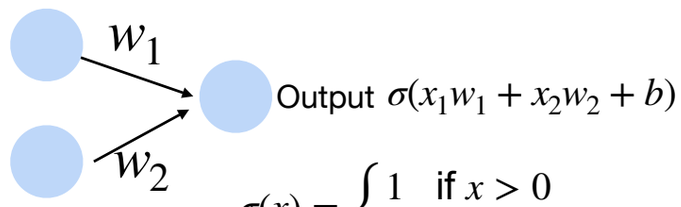
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

What's w and b?

Learning OR function using perceptron

The perceptron can learn an OR function



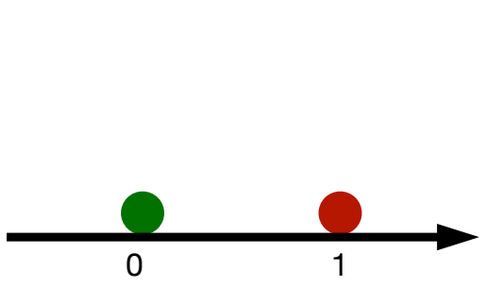


$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 1, w_2 = 1, b = -0.5$$

Learning NOT function using perceptron

The perceptron can learn NOT function (single input)



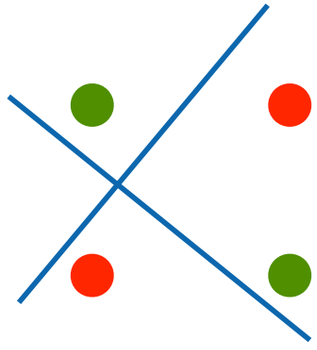
$$\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = -1, b = 0.5$$

Similarly for the NOT function.

XOR Problem (Minsky & Papert, 1969)

The perceptron cannot learn an XOR function
(neurons can only generate linear separators)



This contributed to the first AI winter

However, the XOR function has a nonlinear decision boundary. While any perceptron has a linear decision boundary. Therefore, no perceptron can represent the XOR function. This is the limitation of perceptron: it can only represent linear decision boundaries.

Quiz Break

Consider the linear perceptron with x as the input. Which function can the linear perceptron compute?
(1) $y = \text{ax} + \text{b}$
(2) $y = \text{ax}^2 + bx + c$

A. (1)
B. (2)
C. (1)(2)
D. None of the above

Quiz Break

Consider the linear perceptron with x as the input. Which function can the linear perceptron compute?
(1) $y = ax + b$
(2) $y = ax^2 + bx + c$

A. (1)
B. (2)
C. (1)(2)
D. None of the above

Answer: A. All units in a linear perceptron are linear. Thus, the model can not present non-linear functions.

Linear perceptron is using the identity function as the activation function, so it’s just a linear function.

Quiz Break

Perceptron can be used for representing:

A. AND function
B. OR function
C. XOR function
D. Both AND and OR function

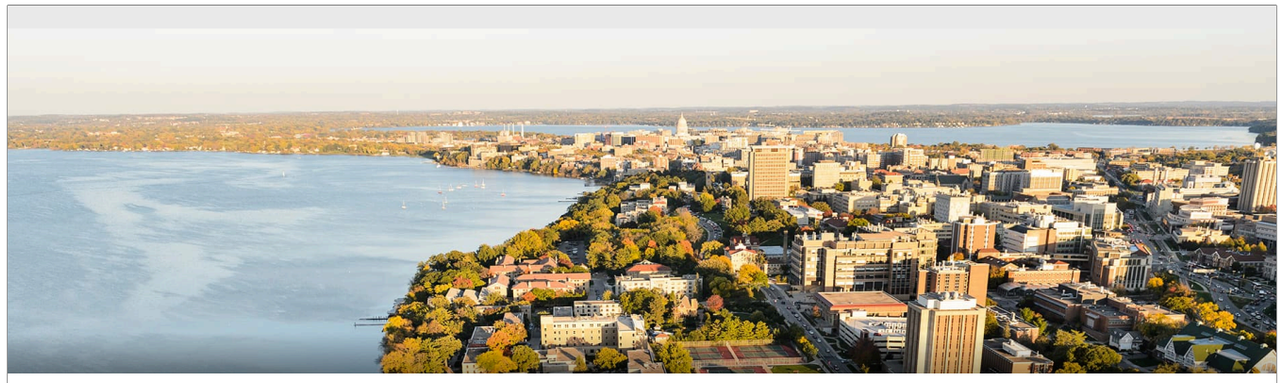
Quiz Break

Perceptron can be used for representing:

A. AND function
B. OR function
C. XOR function
D. Both AND and OR function

What we've learned today…

- Single-layer Perceptron
  - Motivation
  - Activation function
  - Representing AND, OR, NOT



Thanks!