1.5em 0pt

Harvard CS 182: Artificial Intelligence November 10, 2022

Section 10: Decision Trees and Linear Classification

Lecturer: Ariel Procaccia *Authors: Janani Sekar and Catherine Cui*

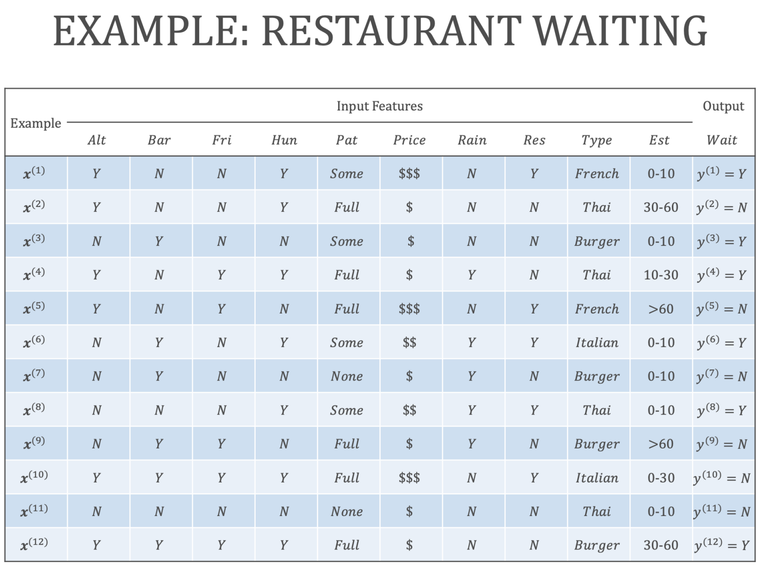
Decision Tree

Classification is the task of learning a **classifier** function f whose range is a discrete, finite set. When the cardinality of the range is 2, the task is a binary classification task; otherwise, it's a multi-class classification.

An example of binary classification is the spam filter in email. An example of a multi-class classification problem is identifying sorting pictures of animals (snakes, lions, etc.)

We can represent any **classifier** using a decision tree, although some may require a large tree. Specifically, a decision tree reaches an output (in the leaves) through a sequence of tests on the input attributes (in the internal nodes). When building a decision tree, the big question(s) are: **what features should we split on and in what order?**

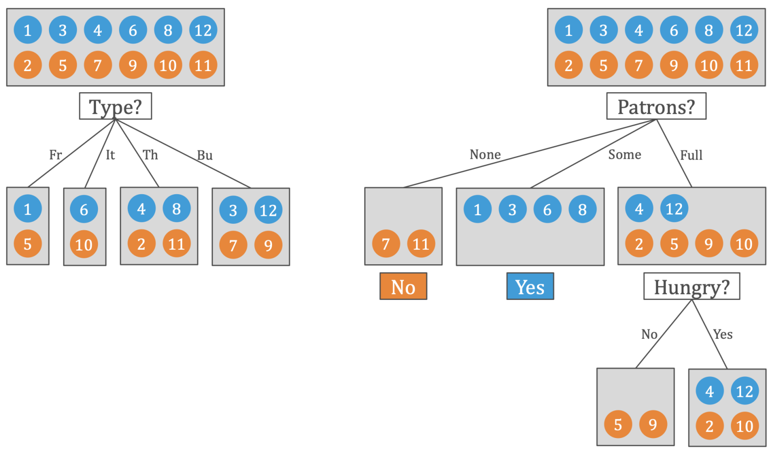
Consider the following data set:



EXAMPLE: RESTAURANT WAITING

Example	Input Features										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
$x^{(1)}$	*Y*	*N*	*N*	*Y*	*Some*	$$$	*N*	*Y*	*French*	0-10	$y^{(1)} = Y$
$x^{(2)}$	*Y*	*N*	*N*	*Y*	*Full*	$	*N*	*N*	*Thai*	30-60	$y^{(2)} = N$
$x^{(3)}$	*N*	*Y*	*N*	*N*	*Some*	$	*N*	*N*	*Burger*	0-10	$y^{(3)} = Y$
$x^{(4)}$	*Y*	*N*	*Y*	*Y*	*Full*	$	*Y*	*N*	*Thai*	10-30	$y^{(4)} = Y$
$x^{(5)}$	*Y*	*N*	*Y*	*N*	*Full*	$$$	*N*	*Y*	*French*	>60	$y^{(5)} = N$
$x^{(6)}$	*N*	*Y*	*N*	*Y*	*Some*	$$	*Y*	*Y*	*Italian*	0-10	$y^{(6)} = Y$
$x^{(7)}$	*N*	*Y*	*N*	*N*	*None*	$	*Y*	*N*	*Burger*	0-10	$y^{(7)} = N$
$x^{(8)}$	*N*	*N*	*N*	*Y*	*Some*	$$	*Y*	*Y*	*Thai*	0-10	$y^{(8)} = Y$
$x^{(9)}$	*N*	*Y*	*Y*	*N*	*Full*	$	*Y*	*N*	*Burger*	>60	$y^{(9)} = N$
$x^{(10)}$	*Y*	*Y*	*Y*	*Y*	*Full*	$$$	*N*	*Y*	*Italian*	0-30	$y^{(10)} = N$
$x^{(11)}$	*N*	*N*	*N*	*N*	*None*	$	*N*	*N*	*Thai*	0-10	$y^{(11)} = N$
$x^{(12)}$	*Y*	*Y*	*Y*	*Y*	*Full*	$	*N*	*N*	*Burger*	30-60	$y^{(12)} = Y$

On the left, we split on the feature **type of restaurant** and on the right, we split on the feature **Patrons?**. Note that intuitively, splitting on **Patrons?** seems better because now, if the answer is **no patrons**, all examples are negative; if the answer is **some**, all examples are positive.



We hope to encode how good splitting on a certain feature is using a function; for now, we take this function $IMPORTANCE$ as a black box.

Below, we present an algorithm for learning a decision tree.

Algorithm 1 Learning Decision Trees

```
Require: $examples, features, parent\_examples$
  if $examples = \emptyset$ then
      return PLURALITY-VALUE($parent\_examples$)
  else if all $examples$ have the same label then
      return that label
  else if $features = \emptyset$ then
      return PLURALITY-VALUE($examples$)
  else
      $A \leftarrow argmax_{a \in features}$IMPORTANCE($a, examples$)    $tree \leftarrow$ new decision tree with
  root test $A$
      for each value $v$ of $A$ do
          $new\_examples \rightarrow \{$e$\in examples : e.A = v\}$
          $subtree \rightarrow$ LEARN-DT($new\_examples, features \setminus \{A\}, examples$)
          add branch to $tree$ with label $A = v$ and $subtree$
      end for
      return $tree$
  end if
```

First, examine the **else** branch. In this branch, we use our IMPORTANCE function to discover the most "important" feature **A** for some definition of important we discuss more later on. Then we create a new decision tree with the root as feature **A**. For each value v of **A**, the function creates a new set of examples $new\_examples$ consisting of all the examples where of value of **A** is v. Then we repeat the process by calling LEARN_DT on this new set of examples, the set of all features except **A** since it's already been split on, and *examples* (which now becomes the set of parent examples). This returns to us a *subtree.* We add a branch to our overall *tree* with label $\mathbf{A} = v$ and *subtree.*

In the **if** statement, we deal with the case in which our *examples* set is empty; the heuristic is to return the plurality value of the parent examples. Similarly, in the second **else if** statement, we deal with the case in which our $features$ set is empty; the heuristic is to return the plurality value of the examples. Lastly, in the first **else if** statement, we've reached our goal.

Information Gain

Now we discuss the IMPORTANCE function. We use the notion of **entropy**, which is measured in bits. The entropy of random variable V that takes each value v with probability $P(v)$ is

$$HH(V) = \sum_v P(v) \log \frac{1}{P(v)} = -\sum_v P(v) \log P(v)$$

For example, the entropy of a biased coin with 99% heads is:

$$H(Biased) = -(0.99 \log 0.99 + 0.01 \log 0.01) \approx 0.08$$

Denote the entropy of a Bernoulli random variable that is true with probability w by

$$B(q) = -(q \log q + (1-q)\log(1-q))$$

If a training set contains p positive eamples and n negative, the entropy of the output variable is

$$H(Output) = B\left(\frac{p}{p+n}\right)$$

Feature A with d values divides the training set into d subsets, each with p_k positive examples and n_k negative. The entropy after testing A is

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B\left(\frac{p_k}{p_k + n_k}\right)$$

The **information gain** from testing A is

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

In LEARN-DT, our IMPORTANCE function is based on information gain.

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Practice Problems using Decision Trees

Problem 1 *Below, we have a table of data we will be working with for the next few problems. What is the information gain from splitting on* ***weather****? What is the information gain from splitting on* ***temperature****?*

1. We find (1) the information gain from splitting on **weather** and (2) the information gain from splitting on **temperature**.

1. Splitting on **weather** gives three different subsets of examples: *Sunny* with probability $\frac{3}{10}$, *Cloudy* with probability $\frac{3}{10}$, and *Rainy* with probability $\frac{4}{10}$. Note that if it's *Sunny*, we play $\frac{1}{3}$ of the time; if it's *Cloudy* we play $\frac{3}{3}$ of the time; if it's *Rainy*, we play $\frac{1}{4}$ of the time. Therefore

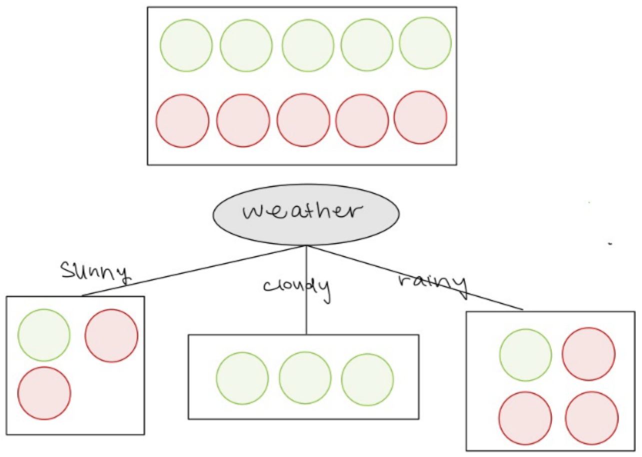
$$Gain(weather) = 1 - \left[\frac{3}{10} \cdot B\left(\frac{1}{3}\right) + \frac{3}{10} \cdot B\left(\frac{3}{3}\right) + \frac{4}{10} \cdot B\left(\frac{1}{4}\right)\right]$$

2. Splitting on **temperature** gives three different subsets of examples: *Hot* with probability $\frac{4}{10}$, *Mild* with probability $\frac{5}{10}$, *Cool* with probability $\frac{1}{10}$. Note that if it's *Hot*, we play $\frac{2}{4}$ of the time; if it's *Mild* we play $\frac{3}{5}$ of the time; if it's *Cool*, we play $\frac{0}{1}$ of the time. Therefore

$$Gain(weather) = 1 - \left[\frac{4}{10} \cdot B\left(\frac{2}{4}\right) + \frac{5}{10} \cdot B\left(\frac{3}{5}\right) + \frac{1}{10} \cdot B\left(\frac{0}{1}\right)\right]$$

Problem 2 *Draw how a decision tree might look like after splitting on* ***weather****.*

2.



Problem 3 *What are some of the issues with Decision Trees? How can we combat these problems?*

3. If we split on too many features, we are at risk of overfitting. The risk is that you'd construct a very detailed tree to perfectly classify a small number of examples, some of which happen to be outliers. Such a tree wouldn't do well on the underlying distribution.

Linear Classification

Linear classification is classification using linear functions. We are trying to find a linear function that can separate our data into classes, e.g. we have positive and negative examples and want to separate them. We define our hypothesis as the following:

$$h_{\mathbf{w}}(\mathbf{x}) = \text{Threshold}(\mathbf{w} \cdot \mathbf{x})$$

Where $\mathbf{x}$ is a data point, and $\mathbf{w}$ is a weight parameter. The threshold function is a sign function, defined as the following:

$$\text{Threshold}(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

We find a linear separator using a linear program to find w such that:

$$\forall \mathbf{x^{(i)}} \in \mathcal{D}^+, \mathbf{w} \cdot \mathbf{x^{(i)}} \geq 0$$

$$\forall \mathbf{x^{(i)}} \in \mathcal{D}^-, \mathbf{w} \cdot \mathbf{x^{(i)}} \leq -\epsilon$$

In other words, we find a **w** such that for positive examples of **x**, $\mathbf{w} \cdot \mathbf{x}$ positive, and for negative examples of **x**, $\mathbf{w} \cdot \mathbf{x}$ is negative.

Problem 4 *We want to find a linear classifiers to represent the binary AND, OR, and XOR operators that separate True examples from False examples. In other words, Threshold(True) is positive, while Threshold(False) is negative. Below, we have the truth tables for these binary operators using data points* $\mathbf{x} = (x_1, x_2)$ *where True = 1 and False = 0.*

AND:

x_1	x_2	x_1 *and* x_2
1	*1*	*1*
1	*0*	*0*
0	*1*	*0*
0	*0*	*0*

OR:

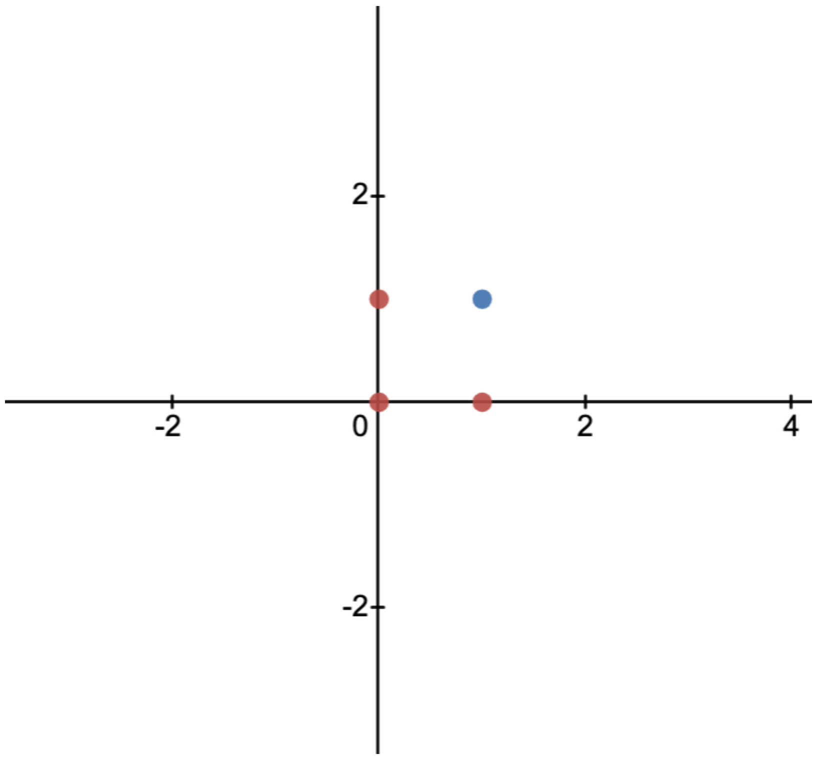
x_1	x_2	x_1 *or* x_2
1	*1*	*1*
1	*0*	*1*
0	*1*	*1*
0	*0*	*0*

XOR:

x_1	x_2	x_1 *xor* x_2
1	*1*	*0*
1	*0*	*1*
0	*1*	*1*
0	*0*	*0*

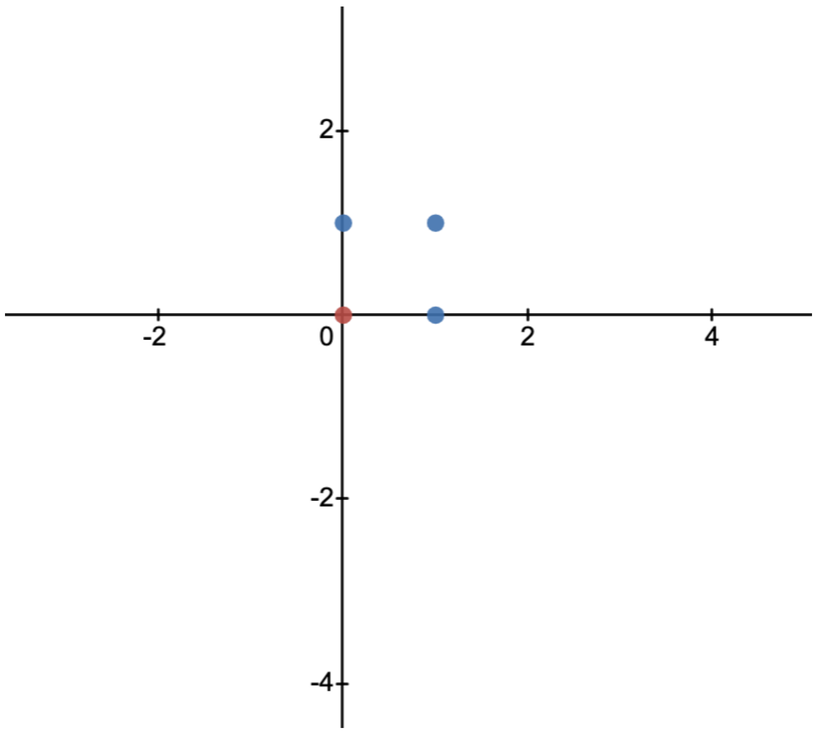
In other words, for each operator AND, OR, XOR, find a $\mathbf{w} = (w_1, w_2)$ *and "bias term"* w_0 *such that* $\mathbf{w} \cdot \mathbf{x_i} + w_0 \geq 0$ *when the result of the binary operation is True, and* $\mathbf{w} \cdot \mathbf{x_i} + w_0 < 0$ *when the result is False, or explain why there no such* **w** *exists in 2 dimensions. (It may help to plot the values in the truth tables and visualize what a separator would look like for each of these cases.)*

4. AND:



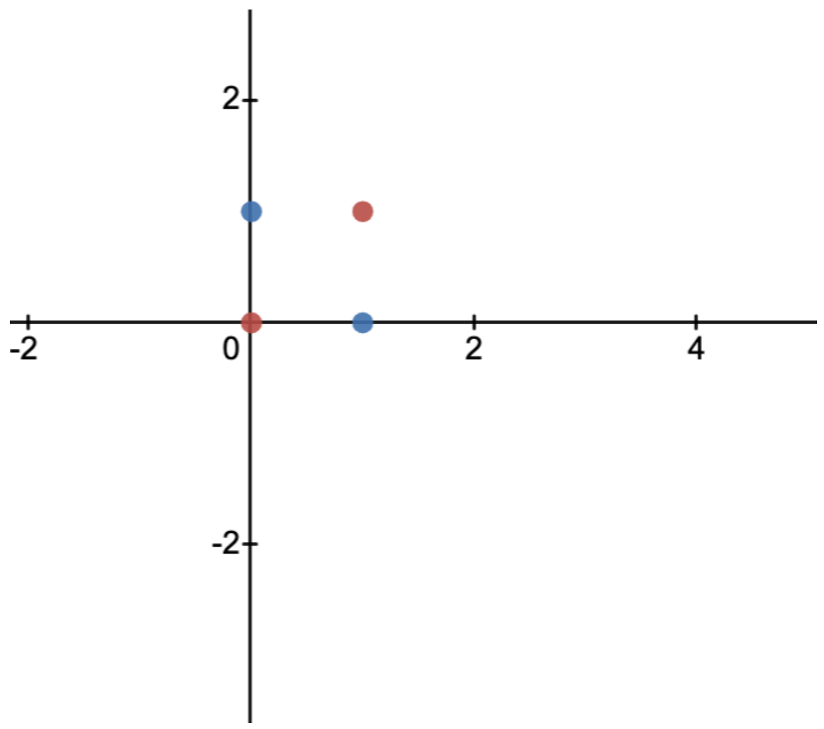
One set of possible values for w_0, w_1, w_2 is $(-\frac{3}{2}, 1, 1)$. Then, for (1,1), $\mathbf{w} \cdot \mathbf{x} + w_0 = \frac{1}{2}$. For (0,1) and (1,0), $\mathbf{w} \cdot \mathbf{x} + w_0 = -\frac{1}{2}$. And for (0,0), $\mathbf{w} \cdot \mathbf{x} + w_0 = -\frac{3}{2}$.

OR:



One set of possible values for w_0, w_1, w_2 is $(-\frac{1}{2}, 1, 1)$. Then, for (1,1), $\mathbf{w} \cdot \mathbf{x} + w_0 = \frac{3}{2}$. For (0,1) and (1,0), $\mathbf{w} \cdot \mathbf{x} + w_0 = \frac{1}{2}$. And for (0,0), $\mathbf{w} \cdot \mathbf{x} + w_0 = -\frac{1}{2}$.

XOR:



For binary data, XOR is not linearly separable.

Perceptron

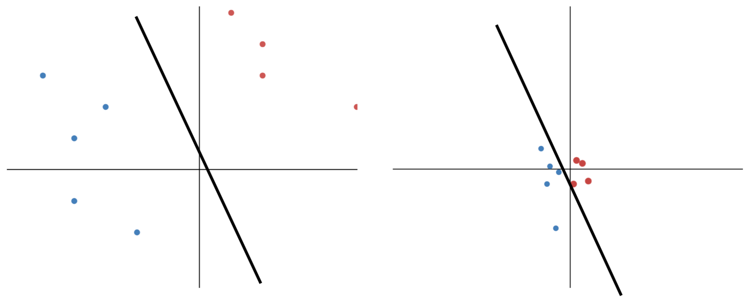
The Perceptron Learning Rule gives us a way to find a linear separator.

Perceptron learning rule: $\forall(x, y)$, classify $\hat{y} = \text{Threshold}(w \cdot x)$. If $\hat{y} \neq y$, update $w = w + y \cdot x$.

In other words, if a point is not correctly classified, we update w to take into account the correct label for that point.

Theorem 1 (Perceptron Mistake Bound) *Given a dataset* $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$*, if* $||x^{(i)}|| \leq R$*,* $\forall i, w^*$ *such that* $||w^*|| = 1$ *and,* $\forall i$*,* $y^{(i)}(w * x(i)) \geq \gamma$*, the number of mistakes made by Perceptron is at most* $\frac{R}{\gamma}^2$

For linearly separable data, geometrically γ is the absolute value of the minimum distance of an example from the separator, or the margin. So if the data can be separated by a margin γ, we can also bound the number of mistakes that Perceptron makes. If the margin is very very small, it is more difficult to separate the data, so it makes sense that Perceptron makes more mistakes for a small γ and finds a solution faster with a large γ.



Problem 5 *Suppose we have the following data. We randomly initialize our weights* **w** *to (1, 1). Iterate through the data in the table (use all the points exactly once) using the Perceptron Learning Rule to update* **w**. *Does the weight vector at the end of this process give us an accurate linear classifier?*

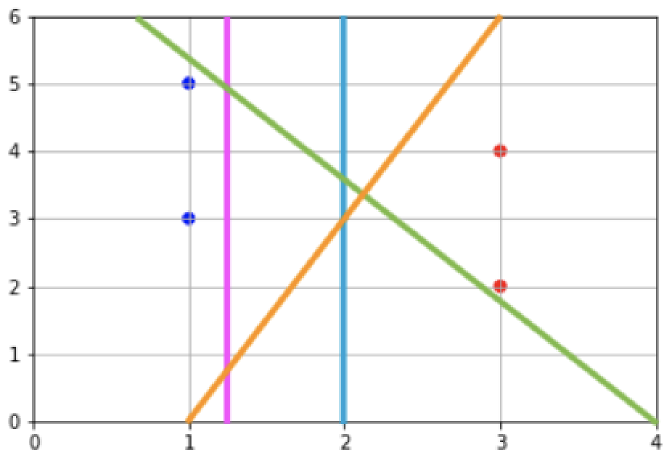
x_1	x_2	y
4	*3*	*1*
1	*3*	*-1*
-2	*0*	*-1*
2	*2*	*1*

5. The initial $\mathbf{w} = (1, 1)$. We take the first data point in our table, $(x, y) = ((4, 3), 1)$. Applying the Perceptron Learning Rule, $(\mathbf{w} \cdot \mathbf{x}) = \mathbf{4} + \mathbf{3} = \mathbf{7}$. Because Threshold(7) $= +1$ which is the same as y for this point, $\mathbf{w}$ remains unchanged. Now we take the next point in our data set $(x, y) = ((1, 3), -1)$. $(\mathbf{w} \cdot \mathbf{x}) = \mathbf{1} + \mathbf{3} = \mathbf{4}$. Because Threshold(4) $= +1$, but the corresponding $y = -1$, we have to update the weight. $\mathbf{w} = (1, 1) - (1, 3) = (0, -2)$. The next point is $(x, y) = ((-2, 0), -1)$. $(\mathbf{w} \cdot \mathbf{x}) = \mathbf{0} + \mathbf{0} = \mathbf{0}$. Threshold(0) $= +1$ (because this is how we break ties for values on the boundary), but the label on this point is actually negative, so we update $\mathbf{w}$ once more. $\mathbf{w} = (0, -2) + (2, 0) = (2, -2)$. Finally, we take the last point $((2, 2), 1)$. $(\mathbf{w} \cdot \mathbf{x}) = \mathbf{4} + -\mathbf{4} = \mathbf{0}$. This is correctly classified (again on the boundary) so we don't update $\mathbf{w}$. At this point, every data point should be correctly classified.

Support Vector Machines

SVM is an algorithm that works by finding the maximum margin separator for a set of data. The maximum margin is defined by support vectors, which are the examples that that lie on the boundary of the margin.

Problem 6 *Consider the following data, where the blue points are positive examples, and the red points are negative examples. Which of the following separators is the maximum margin separator for this data?*



What are the weights w_2, w_1, w_0 for the maximum margin separator? Again, w_0 is is called a bias term (which basically allows us to construct a separator that does not have to go through 0). The decision boundary equation for the separator should be of the form:

$$w_2x_2 + w_1x_1 + w_0 = 0$$

If you scaled this separator by a positive constant k (i.e., replace w_2, w_1, w_0 with kw_2, kw_1, kw_0), would it still be a maximum margin separator?

6. The maximum margin separator is the blue line. In order to find the maximum margin separator, one approach we can take is considering the positive and negative examples that are closest together, and find a separator that is directly between them, to maximize the distance from both points. In this case, we can take either (1,5) and (3,4) or (1,3) and (3,2) since the distance between these pairs of points is the same. Let's just take the first pair. The point directly between these two points is (2, 4.5), so we know that our maximum margin separator has to go through this point. To maximize the distance between the next closest pair of points, we also have to make sure our separator goes through (2, 2.5). The line that accomplishes this is simply $x_1 = 2$, or $x_1 - 2 = 0$. Then, in order to satisfy $w_0 + w_1x_1 + w_2x_2 = 0$, we get $w_1 = 1, w_2 = 0, w_0 = -2$.

Scaling the separator by some factor k would still result in a valid separator. $kx_1 = 2k$ still satisfies $w_0 + w_1x_1 + w_2x_2 = 0$.

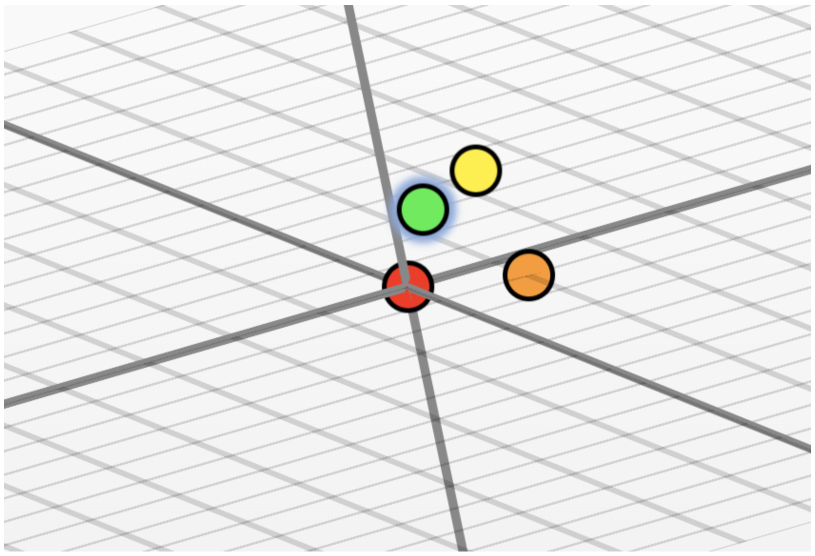
Higher Dimensions

Given a set of data that is not linearly separable in the dimension that it is currently in, we have a couple of different strategies we can use to find a linear separator.

The first is transforming the data to a higher dimension. In lecture, we saw that we could transform data that was not linearly separable in 1-dimension by mapping it to a 2-dimensional space.

Problem 7 *As we saw earlier in section, the XOR operator in 2 dimensions is not separable in 2 dimensions. Is there a transformation we can make to the 2-dimensional data that would allow us to separate the data in 3 dimensions? You do not have to actually find the linear separator. Just find a transformation that would make the data separable and reason why it would work.*

7. A transformation we could consider is mapping (x_1, x_2) to $(x_1, x_2, (x_1 - x_2)^2)$. Then (1, 1) and (0,0) would map to a point with $z = 0$, while (0,1) and (1,0) would map to a point with $z = 1$. To make this a little more intuitive, here's a 3D plot of the transformed XOR, with (1,0) and (0,1) higher along the z-axis. Any plane that is located below these points but above the other two would be a valid linear separator in 3 dimensions.



Logistic Regression

At a conceptual level, a second approach to handling non separable data is through soft margins. We can enable soft margins through different activation functions.

Kind of like we did with the binary logic operators, in order to do this, we take negative labels to be 0 and positive labels to be 1.

In contrast to using the threshold function which labels anything below 0 as 0 and anything above zero as 1, we can use the logistic function. The logistic function is defined as

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

This function is continuous and less "decisive" e.g. instead of assigning a label of 0 or 1 to a data point, it will give us some value on the continuous interval [0,1]. In logistic regression, instead of plugging $\mathbf{w} \cdot \mathbf{x}$ into the Threshold function like we did before, we plug it into the logistic function, and interpret the output as the probability of having a positive label.

For some $\mathbf{w}$, the probability of observing $\{(\mathbf{x^{(i)}}, y^{(i)})\} =$

$$\prod_i \sigma(\mathbf{w} \cdot \mathbf{x^{(i)}})^{y(i)} \cdot (1 - \sigma(\mathbf{w} \cdot \mathbf{x^{(i)}}))^{1-y^{(i)}}$$

This is the product over all the points in the data set of the probability of observing 1 when the label is 1, or 0 when the label is 0 (i.e. the probability of the the example matching its given label). We derive the log-likelihood function by taking the log of this expression:

$$LL(\mathbf{w}) = \sum_i y^{(i)} \log \sigma(\mathbf{w} \cdot \mathbf{x^{(i)}}) + (1 - y^{(i)}) \log (1 - \sigma(\mathbf{w} \cdot \mathbf{x^{(i)}}))$$

We want to find the $\mathbf{w}$ that makes the observed labels as likely as possible. We can do this by finding the that maximizes the log-likelihood. We can maximize the log-likelihood function using gradient ascent, or taking steps in the direction of the gradient until a maximum is reached.