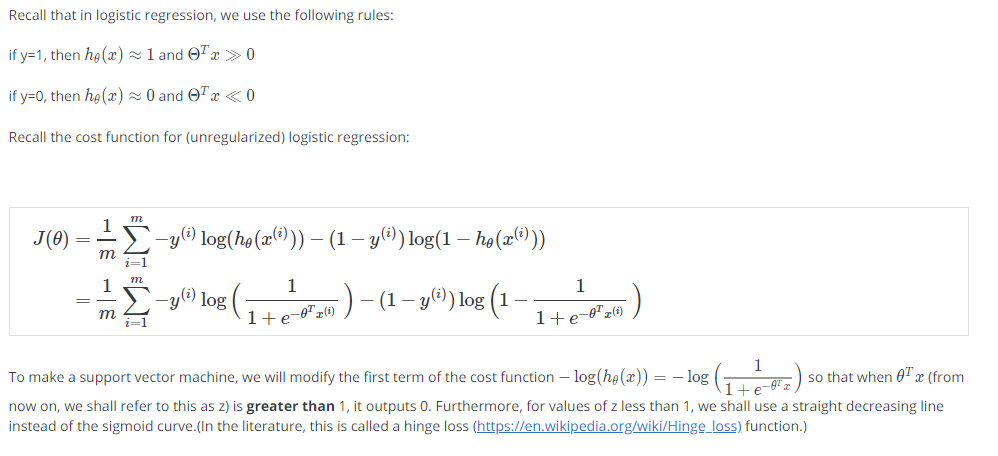
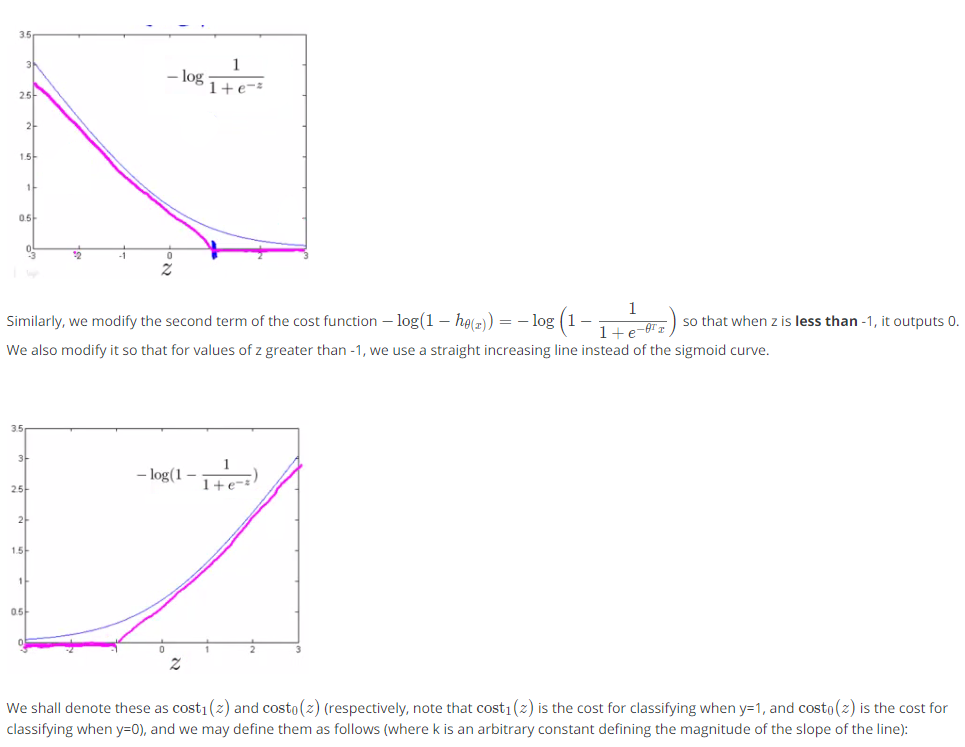
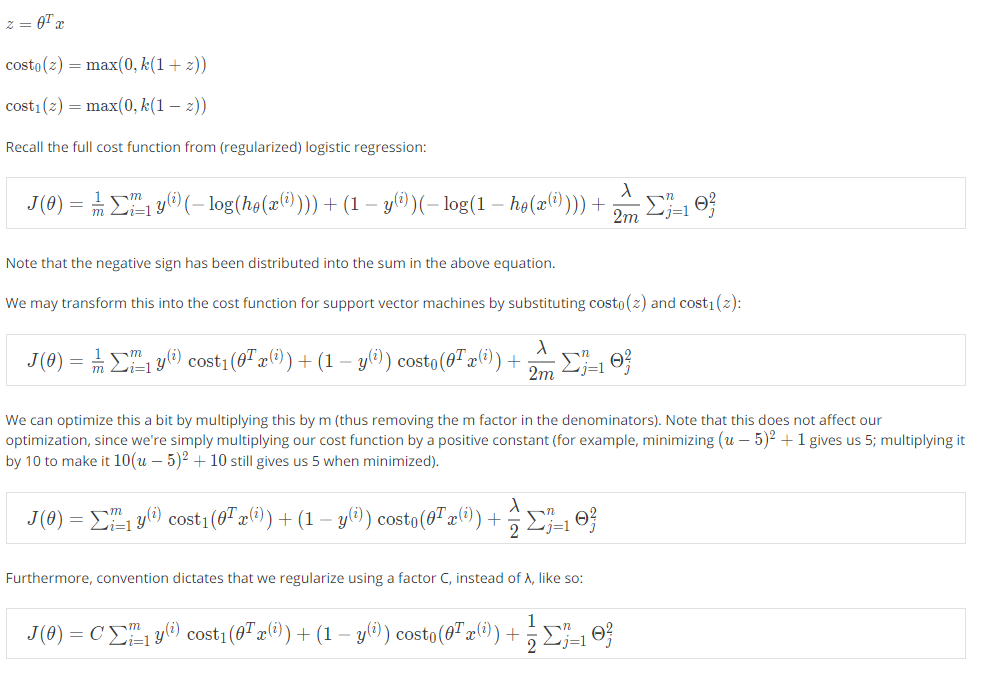
# Support Vector Machines

## Optimization Objective

The **Support Vector Machine** (SVM) is yet another type of supervised machine learning algorithm. It is sometimes cleaner and more powerful.

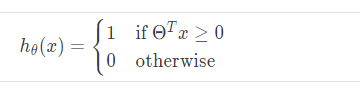




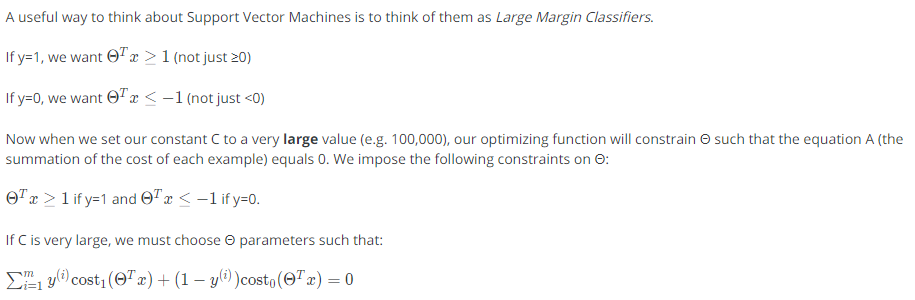


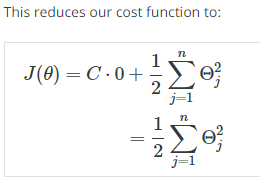
This is equivalent to multiplying the equation by C = 1/ λ and thus results in the same values when optimized. Now, when we wish to regularize more (that is, reduce overfitting), we decrease C, and when we wish to regularize less (that is, reduce underfitting), we increase C.

Finally, note that the hypothesis of the Support Vector Machine is not interpreted as the probability of y being 1 or 0 (as it is for the hypothesis of logistic regression). Instead, it outputs either 1 or 0. (In technical terms, it is a discriminant function.)



## Large Margin Intuition





Recall the decision boundary from logistic regression (the line separating the positive and negative examples). In SVMs, the decision boundary has the special property that it is **as far away as possible** from both the positive and the negative examples.

The distance of the decision boundary to the nearest example is called the **margin**. Since SVMs maximize this margin, it is often called a **Large Margin Classifier.**

The SVM will separate the negative and positive examples by a **large margin**.

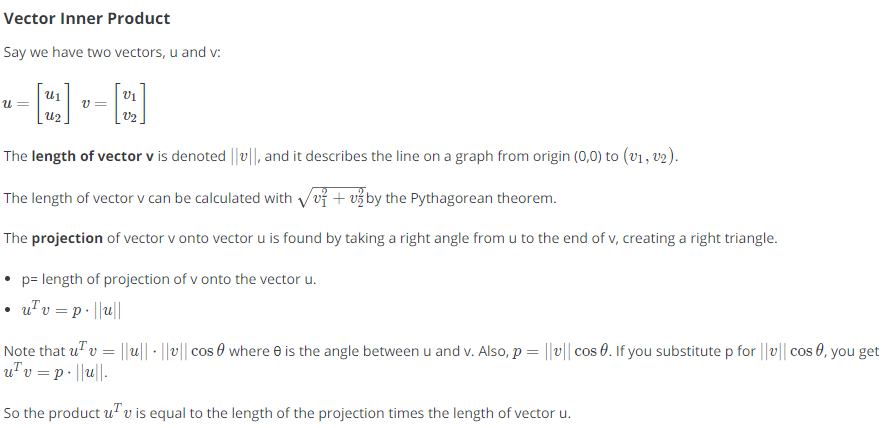
This large margin is only achieved when **C is very large**.

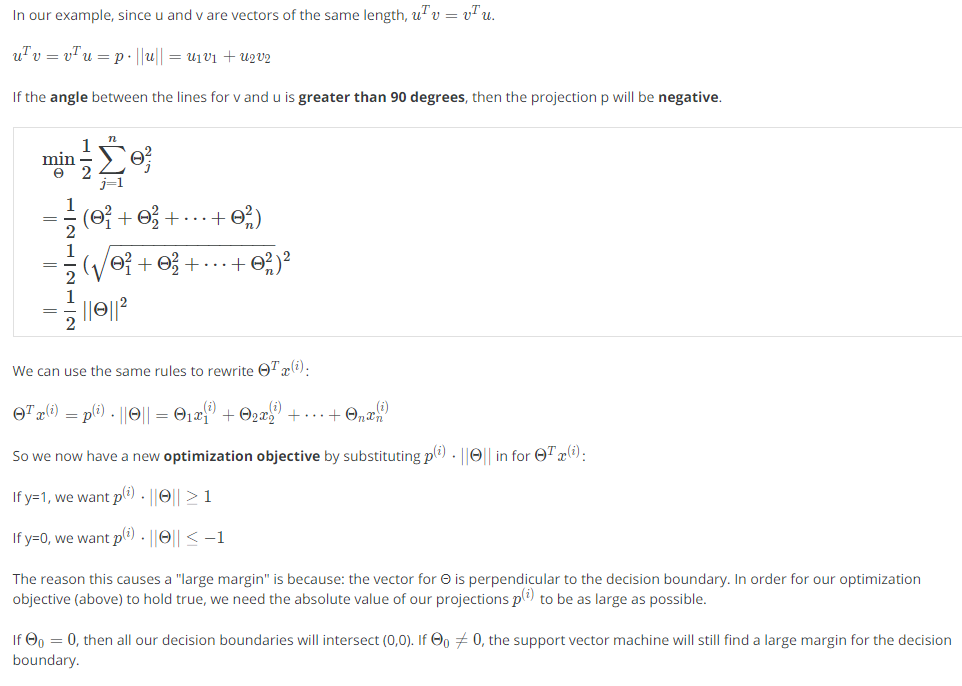
Data is **linearly separable** when a **straight line** can separate the positive and negative examples.

If we have **outlier examples that we don't want to affect the decision boundary, then we can reduce C.**

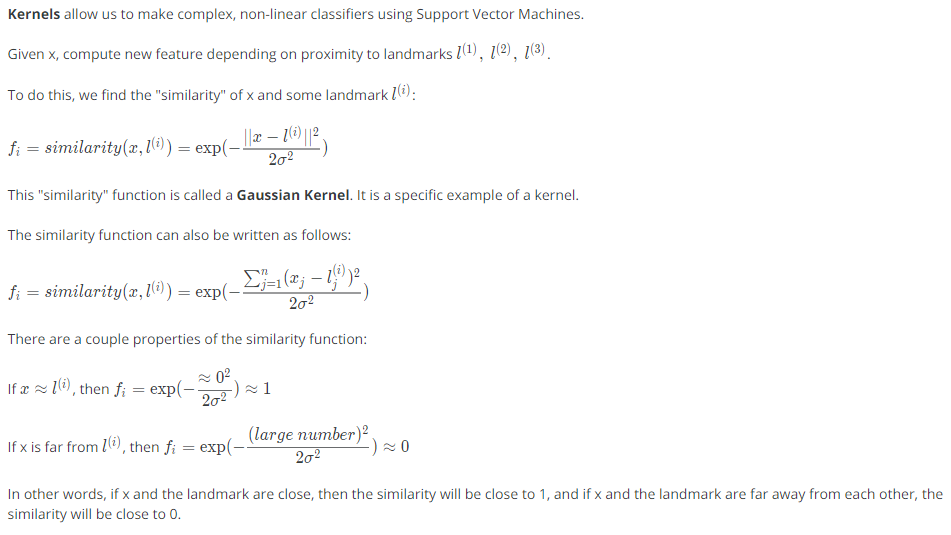
Increasing and decreasing C is similar to respectively decreasing and increasing λ, and can simplify our decision boundary.

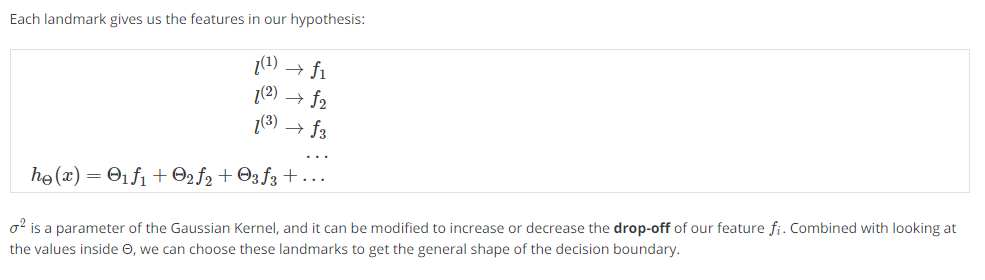
## Mathematics behind Large Margin Classifier





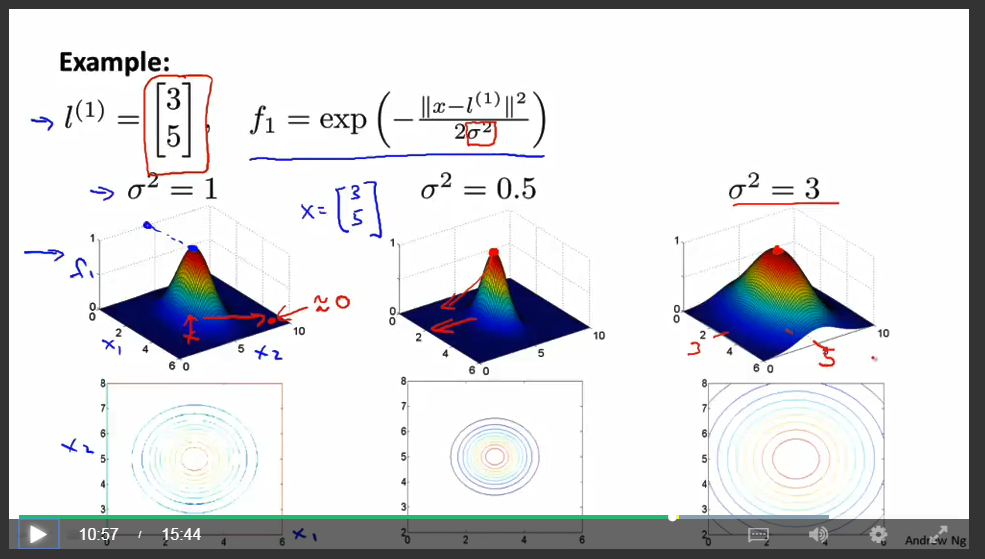
## Kernels I





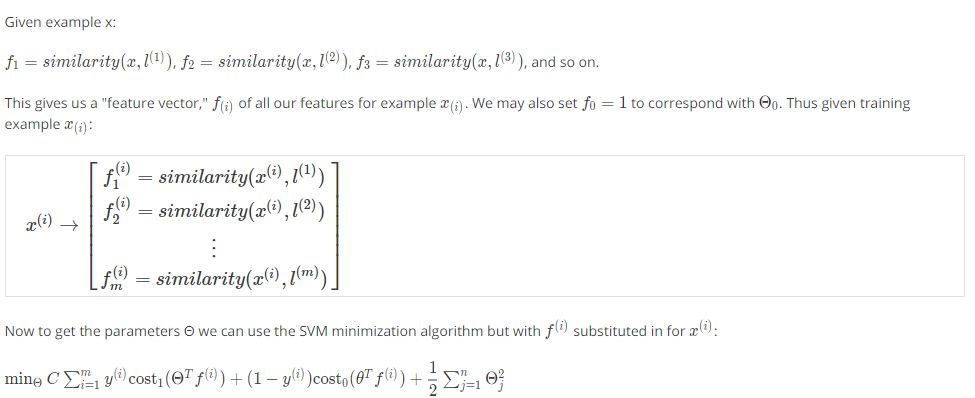
The value of sigma in the denominator decides how rapidly the value of feature f(i) falls to zero. If it’s small, then it falls rapidly and if it’s large, it falls slowly as we move away from the respective landmark.

The same can be explained in the below image.



## Kernels II

One way to get the landmarks is to put them in the **exact same** locations as all the training examples. This gives us m landmarks, with one landmark per training example.



Using kernels to generate f(i) is not exclusive to SVMs and may also be applied to logistic regression. However, because of computational optimizations on SVMs, kernels combined with SVMs is much faster than with other algorithms, so kernels are almost always found combined only with SVMs.

**Choosing SVM parameters**

Choosing C (recall that C = 1/λ

* If C is large, then we get higher variance/lower bias
* If C is small, then we get lower variance/higher bias

The other parameter we must choose is σ2 from the Gaussian Kernel function:

With a large σ2, the features fi vary more smoothly, causing higher bias and lower variance.

With a small σ2, the features fi vary less smoothly, causing lower bias and higher variance.

## SVMs in practice

**Using an SVM**

There are lots of good SVM libraries already written. A. Ng often uses 'liblinear' and 'libsvm'. In practical application, you should use one of these libraries rather than rewrite the functions.

In practical application, the choices you do need to make are:

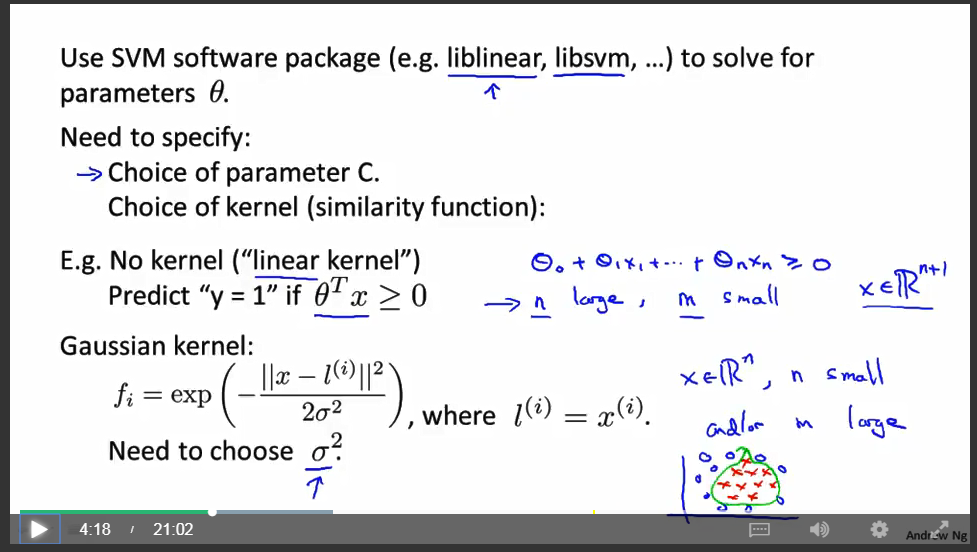
* Choice of parameter C
* Choice of kernel (similarity function)
* No kernel ("linear" kernel) -- gives standard linear classifier
* Choose when n is large and when m is small
* Gaussian Kernel (above) -- need to choose σ2
* Choose when n is small and m is large

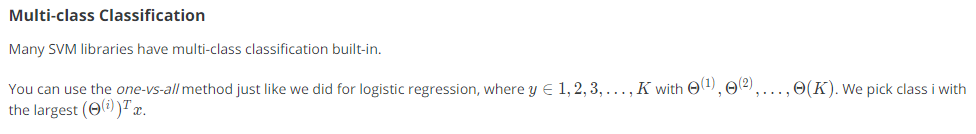
The library may ask you to provide the kernel function.

**Note**: do perform feature scaling before using the Gaussian Kernel.

**Note**: not all similarity functions are valid kernels. They must satisfy "Mercer's Theorem" which guarantees that the SVM package's optimizations run correctly and do not diverge.

You want to train C and the parameters for the kernel function using the training and cross-validation datasets.

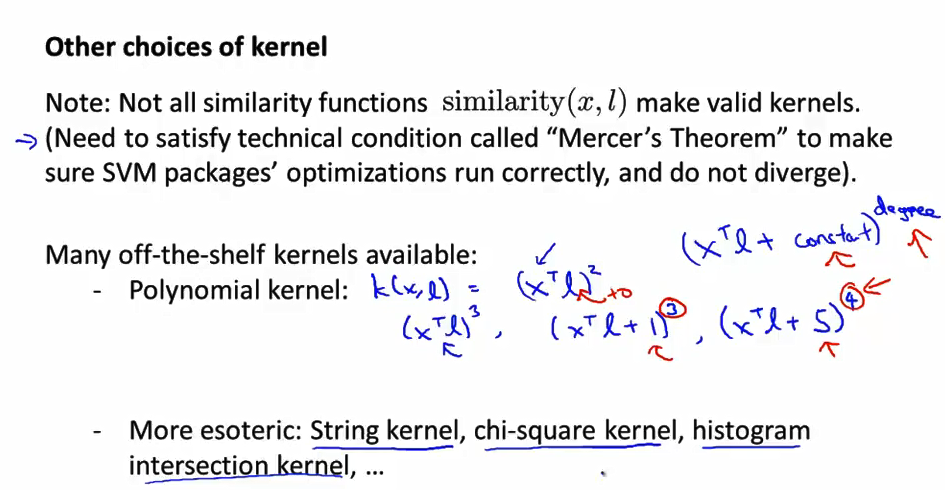


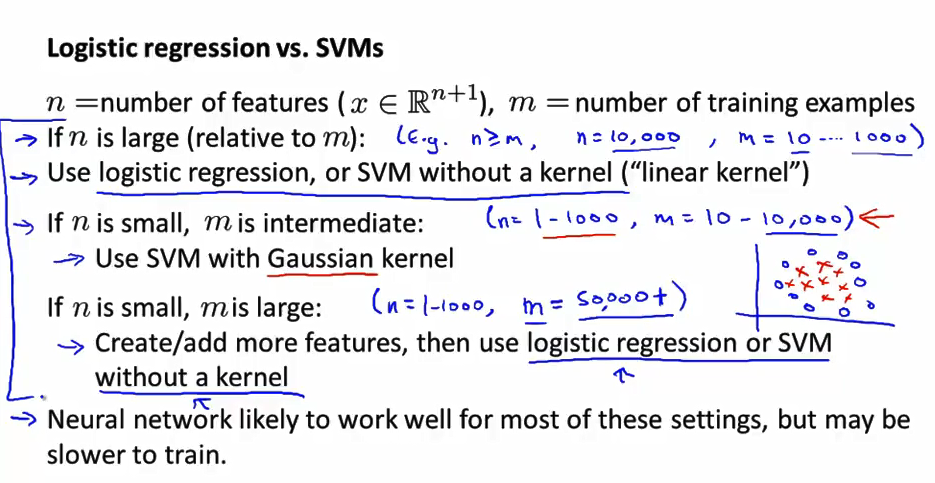


**Logistic Regression vs. SVMs**

* If n is large (relative to m), then use logistic regression, or SVM without a kernel (the "linear kernel")
* If n is small and m is intermediate, then use SVM with a Gaussian Kernel
* If n is small and m is large, then manually create/add more features, then use logistic regression or SVM without a kernel.
* In the first case, we don't have enough examples to need a complicated polynomial hypothesis. In the second example, we have enough examples that we may need a complex non-linear hypothesis. In the last case, we want to increase our features so that logistic regression becomes applicable.

**Note**: a neural network is likely to work well for any of these situations, but may be slower to train.





Unsupervised Learning