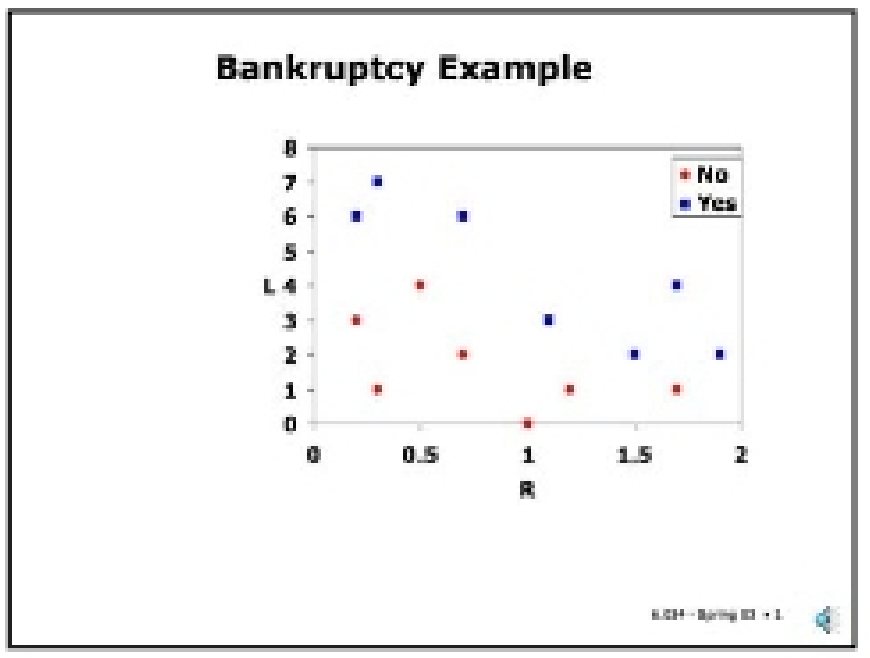
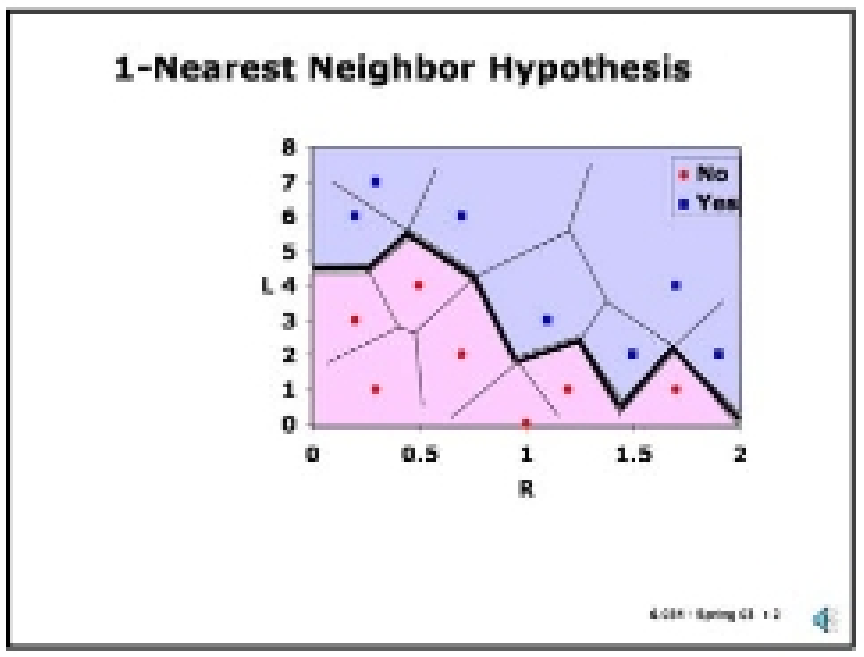
6.034 Notes: Section 7.1

Slide 7.1.1
We have been using this simulated bankruptcy data set to illustrate the different learning algorithms that operate on continuous data. Recall that R is supposed to be the ratio of earnings to expenses while L is supposed to be the number of late payments on credit cards over the past year. We will continue using it in this section where we look at a new hypothesis class, **linear separators.**

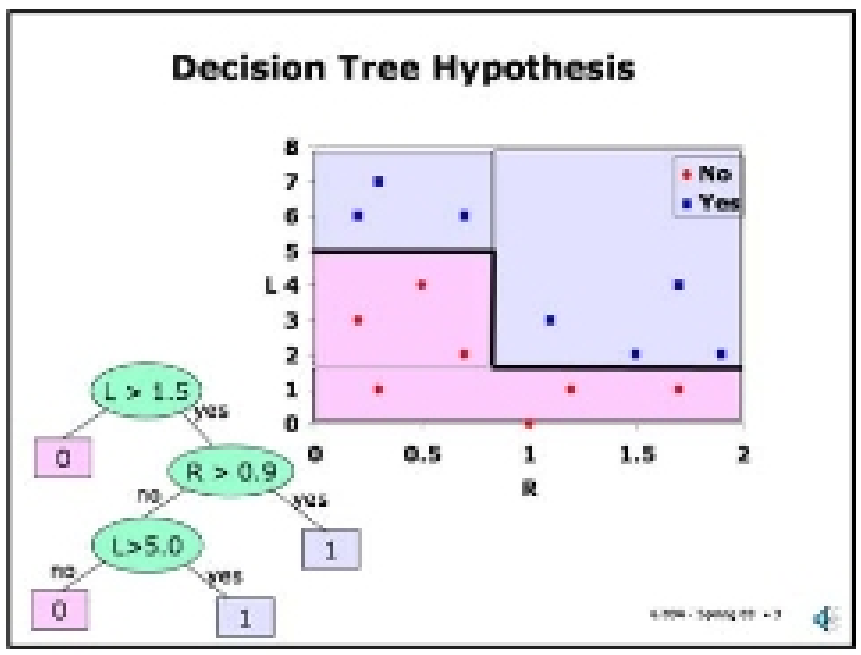
One key observation is that each hypothesis class leads to a distinctive way of defining the **decision boundary** between the two classes. The decision boundary is where the class prediction changes from one class to another. Let's look at this in more detail.

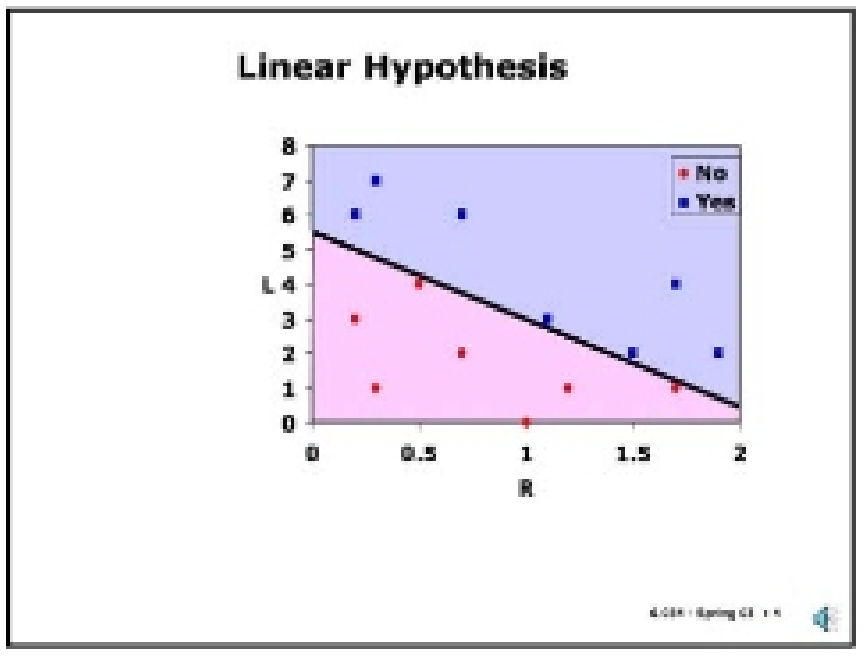




Slide 7.1.2
We mentioned that a hypothesis for the 1-nearest neighbor algorithm can be understood in terms of a Voronoi partition of the feature space. The cells illustrated in this figure represent the feature space points that are closest to one of the training points. Any query in that cell will have that training point as its nearest neighbor and the prediction will be the class of that training point. The decision boundary will be the boundary between cells defined by points of different classes, as illustrated by the bold line shown here.

Slide 7.1.3
Similarly, a decision tree also defines a decision boundary in the feature space. Note that although both 1-NN and decision trees agree on all the training points, they disagree on the precise decision boundary and so will classify some query points differently. This is the essential difference between different learning algorithms.

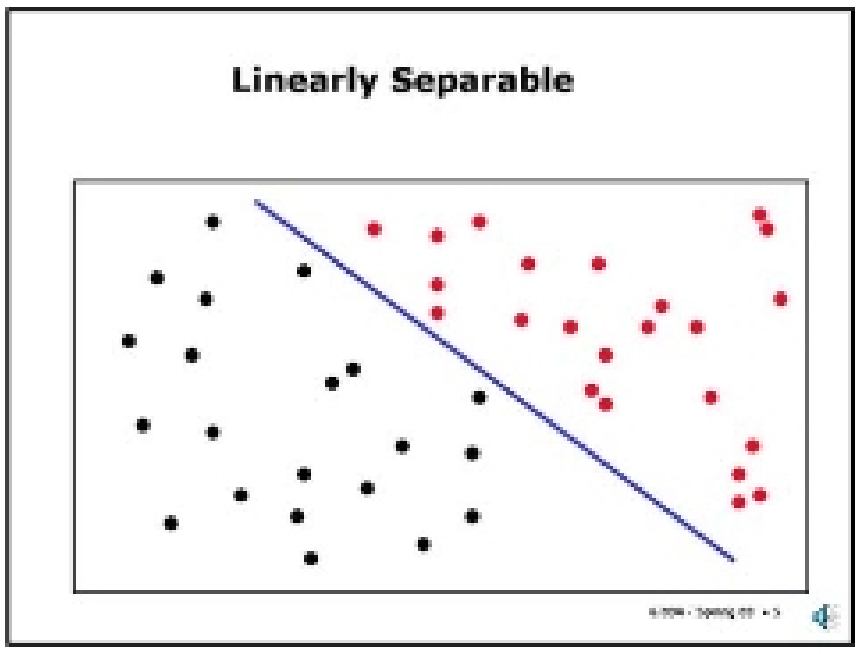


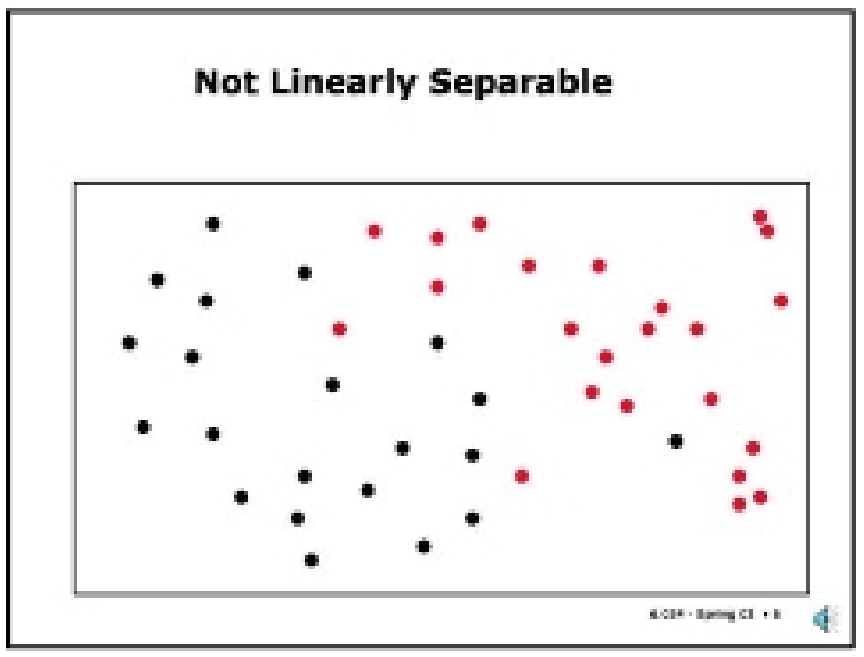


Slide 7.1.4
In this section we will be exploring **linear separators** which are characterized by a single linear decision boundary in the space. The bankruptcy data can be successfully separated in that manner. But, notice that in contrast to 1-NN and decision trees, there is no guarantee that a single linear separator will successfully classify any set of training data. The linear separator is a very simple hypothesis class, not nearly as powerful as either 1-NN or decision trees. However, as simple as this class is, in general, there will be many possible linear separators to choose from.

Also, note that, once again, this decision boundary disagrees with that drawn by the previous algorithms. So, there will be some data sets where a linear separator is ideally suited to the data. For example, it turns out that if the data points are generated by two Gaussian distributions with different means but the same standard deviation, then the linear separator is optimal.

Slide 7.1.5
A data set that can be successfully split by a linear separator is called, not surprisingly, **linearly separable**.

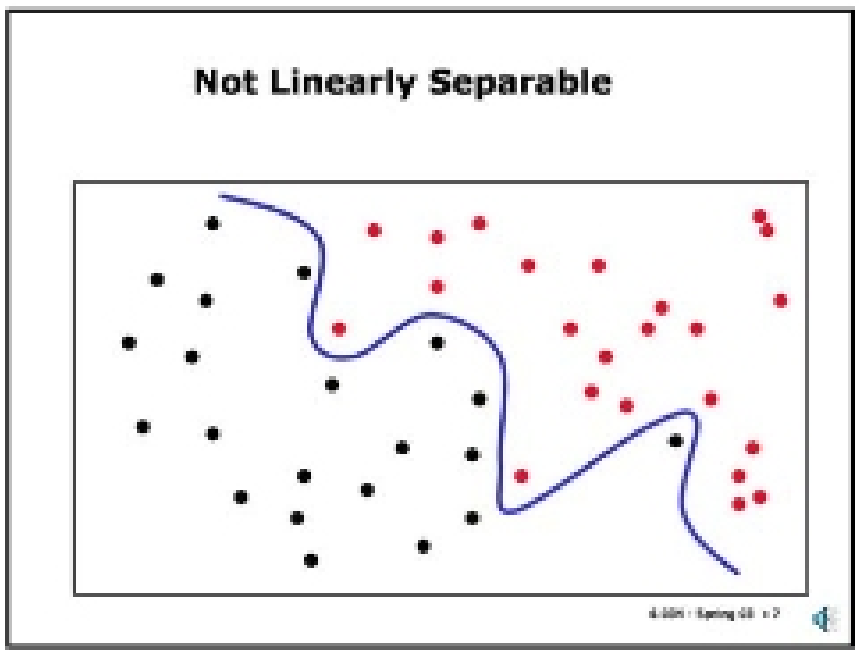


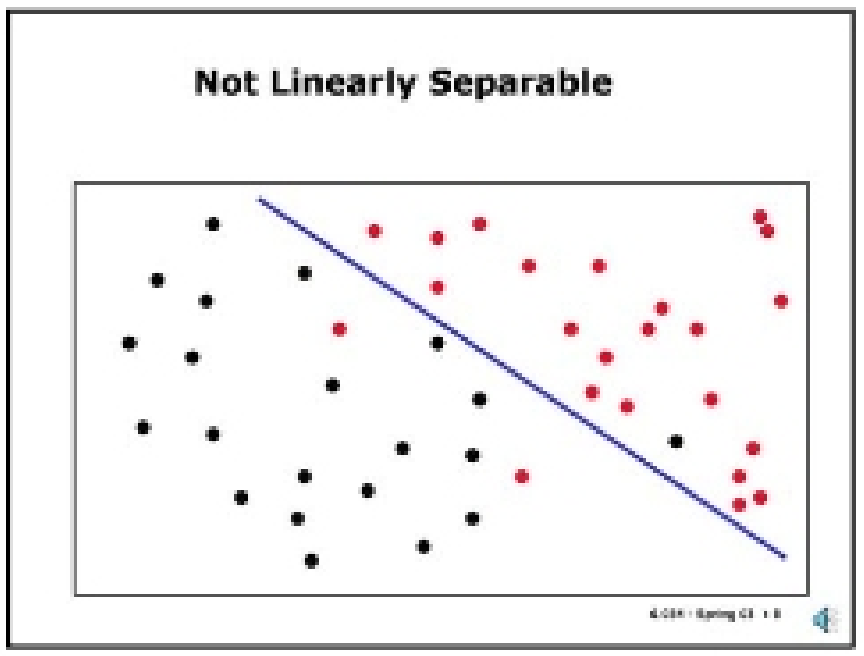


Slide 7.1.6
As we've mentioned, not all data sets are linearly separable. Here's one for example. Another classic non-linearly-separable data set is our old nemesis XOR.

It turns out, although it's not obvious, that the higher the dimensionality of the feature space, the more likely that a linear separator exists. This will turn out to be important later on, so let's just file that fact away.

Slide 7.1.7
When faced with a non-linearly-separable data set, we have two options. One is to use a more complex hypothesis class, such as shown here.



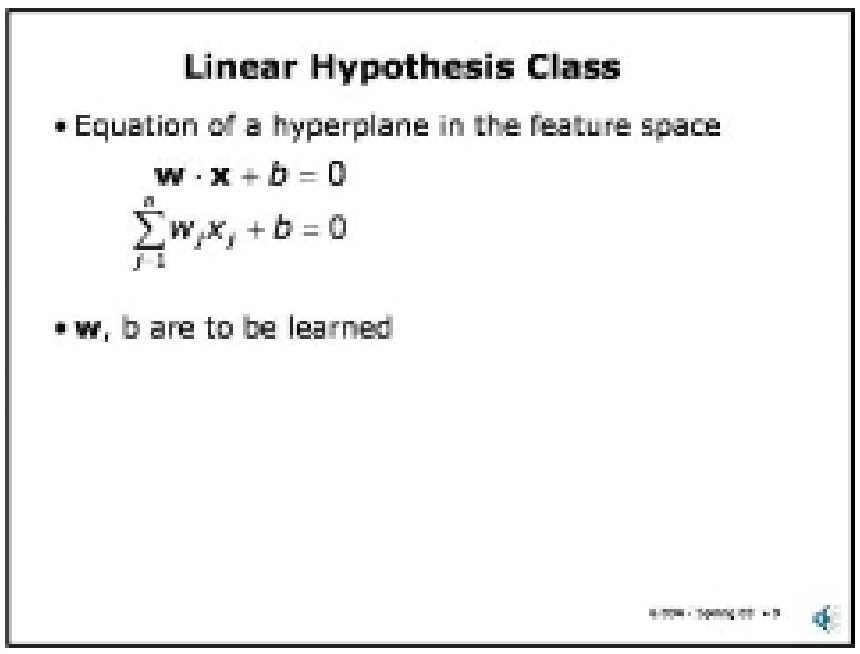


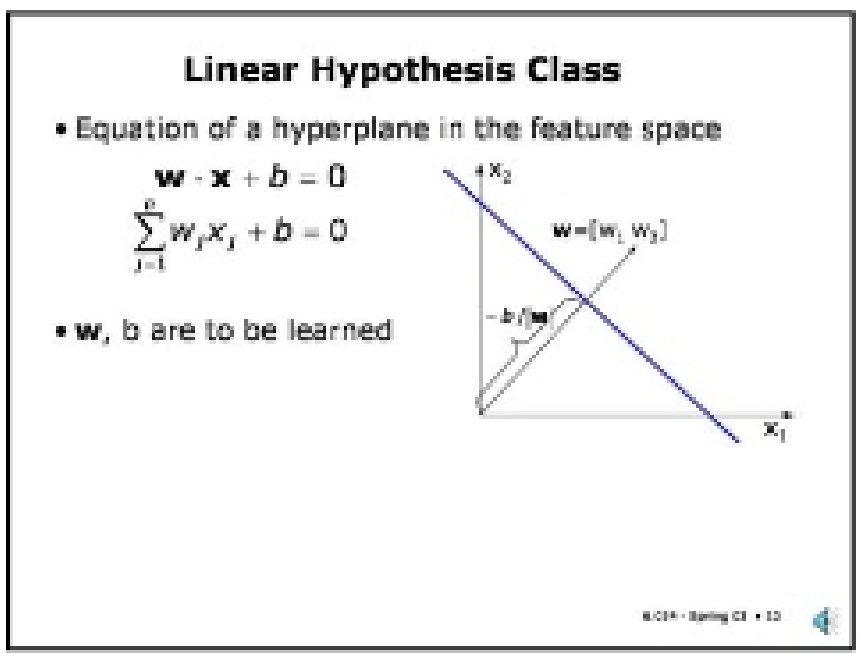
Slide 7.1.8
Or, keep the simple linear separator and accept some errors. This is the classic bias/variance tradeoff. Use a more complex hypothesis with greater variance or a simpler hypothesis with greater bias. Which is more appropriate depends on the underlying properties of the data, including the amount of noise. We can use our old friend cross-validation to make the choice if we don't have much understanding of the data.

Slide 7.1.9
So, let's look at the details of linear classifiers. First, we need to understand how to represent a particular hypothesis, that is, the equation of a linear separator. We will be illustrating everything in two dimensions but all the equations hold for an arbitrary number of dimensions.

The equation of a linear separator in an n-dimensional feature space is (surprise!) a linear equation which is determined by n+1 values, the components of an n-dimensional coefficient vector **w** and a scalar value *b*. These n+1 values are what will be learned from the data. The **x** will be some point in the feature space.

We will be using dot product notation for compactness and to highlight the geometric interpretation of this equation (more on this in a minute). Recall that the dot product is simply the sum of the componentwise products of the vector components, as shown here.





Slide 7.1.10
In two dimensions, we can see the geometric interpretation of **w** and *b*. The vector **w** is perpendicular to the linear separator; such a vector is known as the **normal** vector. Often we say "the vector normal to the surface". The scalar *b*, which we will call the **offset**, is proportional to the perpendicular distance from the origin to the linear separator. The constant of proportionality is the negative of the magnitude of the normal vector. We'll examine this in more detail soon.

By the way, the choice of the letter "w" is traditional and meant to suggest "weights", we'll see why when we look at neural nets. The choice of "b" is meant to suggest "bias" - which is the third different connotation of this word in machine learning (the bias of a hypothesis class, bias vs variance, bias of a separator). They are all fundamentally related; they all refer to a difference from a neutral value. To keep the confusion down to a dull roar, we won't call *b* a bias term but are telling you about this so you won't be surprised if you see it elsewhere.

Slide 7.1.11
Sometimes we will use the following trick to simplify the equations. We'll treat the offset as the 0^{th} component of the weight vector **w** and we'll augment the data vector **x** with a 0^{th} component that will always be equal to 1. Then we can write a linear equation as a dot product. When we do this, we will indicate it by using an overbar over the vectors.