Logistic Regression

Last time we introduced the logistic regression method for building a classification algorithm from training data.

In summary, we model the probability of being one of the two classes (orange, in the example) is a transformation of a linear combination of the feature variables X and Y.

We then classify the predicted class based on whether the probability is greater than 0.5

$$\text{Probability}(\bullet) = F(a + b \times X + c \times Y)$$

More Variables

We can easily extend our model to include more feature variables by adding them along with a new constant.

While we lose the ability to make simply, pretty visualizations of the classification task, most of the same general concepts hold.

For example, with three dimensions, the model creates a classification plane in three dimensions that partitions the three dimensional space into two halves.

$$\text{Probability}(\bullet) = F(a + b \times X + c \times Y + d \times Z)$$

Too Many Variables

In theory we can include as many features as we want. Adding up to a dozen or two is usually fine.

However, once we start adding a larger number of features, it becomes too easy to find a set of model parameters that can seperate the two classes in the training data but is unable to work well on the validation data.

The phenomenon of matching the training data in a way that does not generalize to new data is called **overfitting**. It's hard to define a precise way, but you'll start to understand the concept through looking at examples.

Working with Many Variables

Fortunately, there is a solution to making logistic regression feasible when we have a large number of features.

In a standard logistic regression we find the set of parameters that maximizes how well the data fits the parameters. Specifically, we want to maximize the probability of observing the data we found assuming the modelling probabilities are correct.

Logistic Regression =>

Maximise [**Fit**]

Penalized Logistic Regression

In penalized logistic regression, we instead maximize the fit of the data minus a measurement of the complexity of the model.

We use a parameter $\lambda > 0$ to control the relative tradeoff between fitting the training data well and not having a model that is overly complex.

Penalized Logistic Regression =>

$$\text{Maximise} \left[\text{Fit} - \lambda \times \text{Complexity} \right]$$

Elastic Net

There are several different varieties of penalized regression, which depend on the way that we measure the complexity of a model.

The most important for us this semester is a penalized regression method called the **elastic net**. It has a few sub-varieties, but in its simpliest form, the measure of complexity it given by the sum of the absolute value of the non-intercept model parameters.

$$\text{Complexity} \Rightarrow \quad |b| + |c| + |d|$$

Elastic Net

It should be clear that using this complexity measurement will have parameters that are smaller than those from the standard logistic regression.

The amazing thing, which is not immediately clear, is that the elastic net method will often set some of the model parameters exactly the zero. This is called the **parsimonious** or **model selection** property.

$$\text{Complexity} \Rightarrow \quad |b| + |c| + |d|$$

Elastic Net: Applications

In addition to being a good overall method for predictive modelling, the elastic net is also useful for identifying features that are most related to a specific respose.

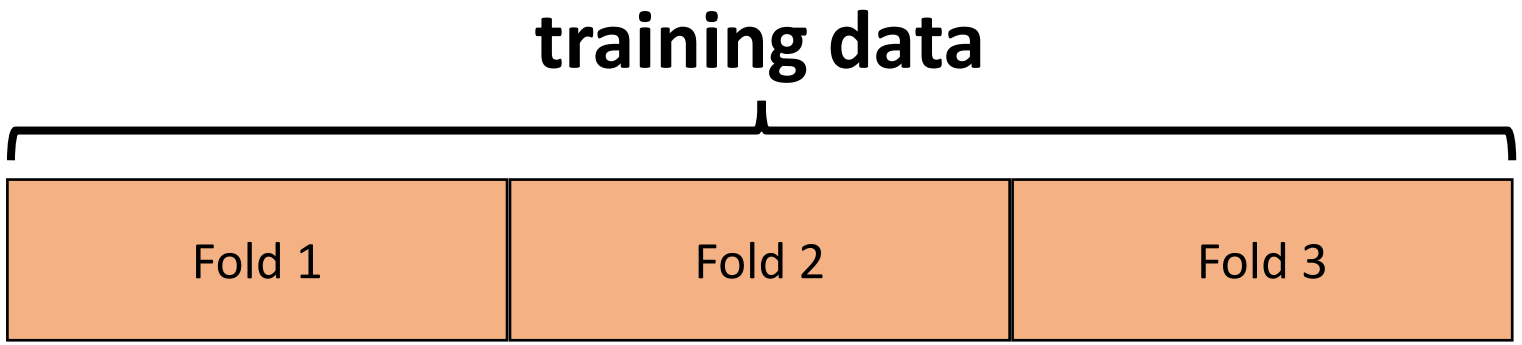
For example, this technique is often used in bioinformatics. For example, one might collect data about the genes of 50 patients who repond well to a theraputic treatement and 50 patients who responded poorly. Then, we could use the elastic net method to predict treatement outcomes based on the tens of thousands of features from the genes. The genes that have non-zero parameters are those that might be responsible for whether the treatment works.

In this class, we will mostly be doing text prediction where the features are the number of times a particular word is used. There are a large number of words, but the elastic net will identify the small set of words that are important for identifying a particular category.

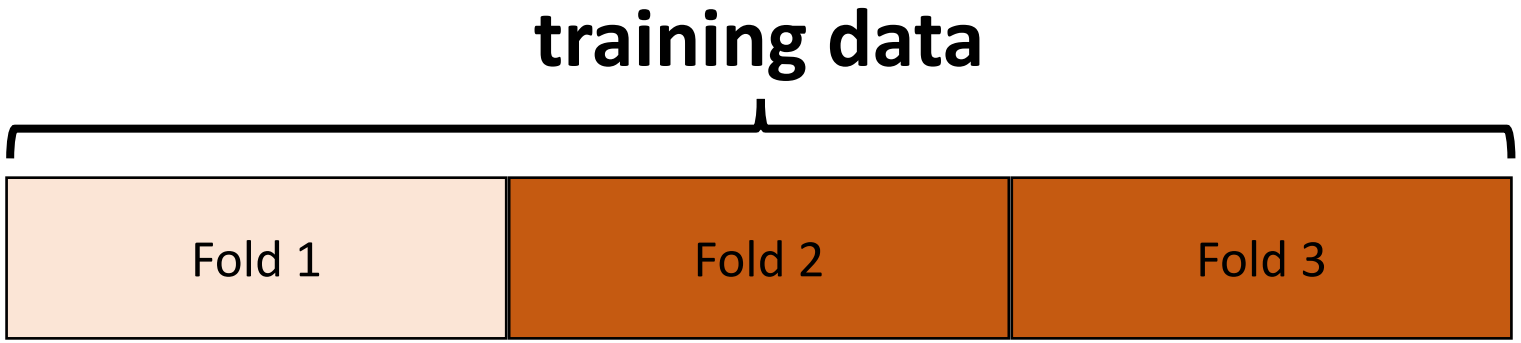
Finding the Tuning Parameter

When using the elastic net, how can we find a good value of the parameter λ ? If we want to just make good predictions, we can use a technique called **cross validation**. This is an important technique in machine learning, so let's take a moment to understand how it works.

In cross validation, we start by randomly partitioning the training data into groups that are called **folds**. We will illustrate using an example with three folds:



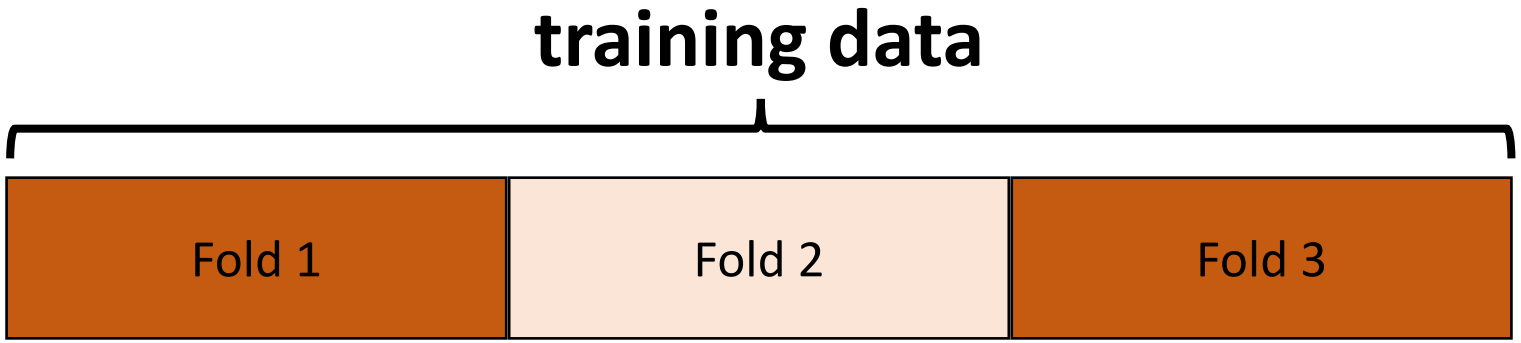
Cross Validation



Now, we fit a collection of elastic net models with different values for λ using all the training data **except** for the data in the first fold.

Then, we use each of these models to predict how well they perform on the held-out data in the first fold.

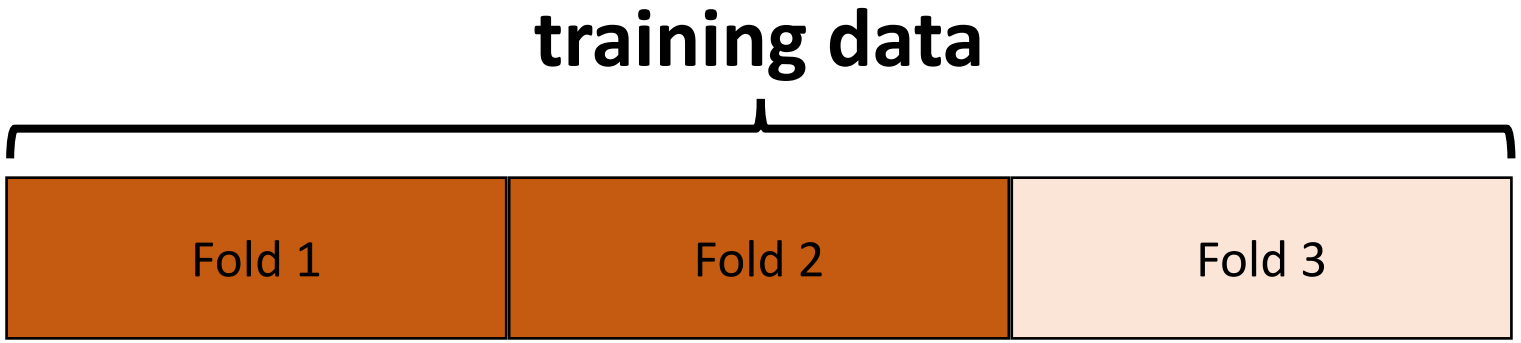
Cross Validation



Then, we repeat the process but now determine all the model parameters using all the training data except for the observations found in fold 2.

These model are then used to predict the values in fold 2.

Cross Validation



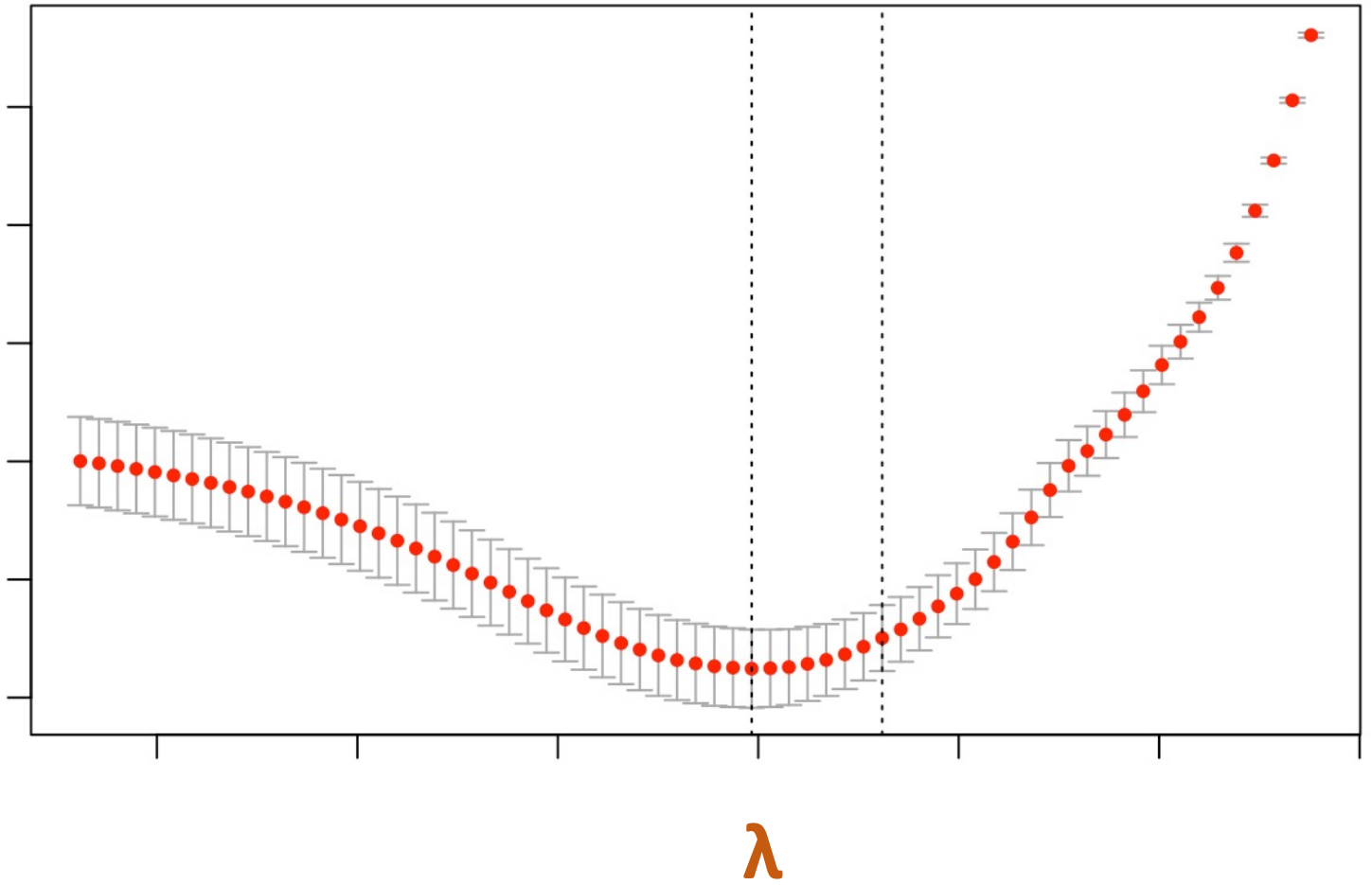
Finally, we do this all again for the data in fold 3.

When we are done, we have a measurement of how well each particular value of lambda is able to predict the held-out data in each of the folds.

Cross Validation

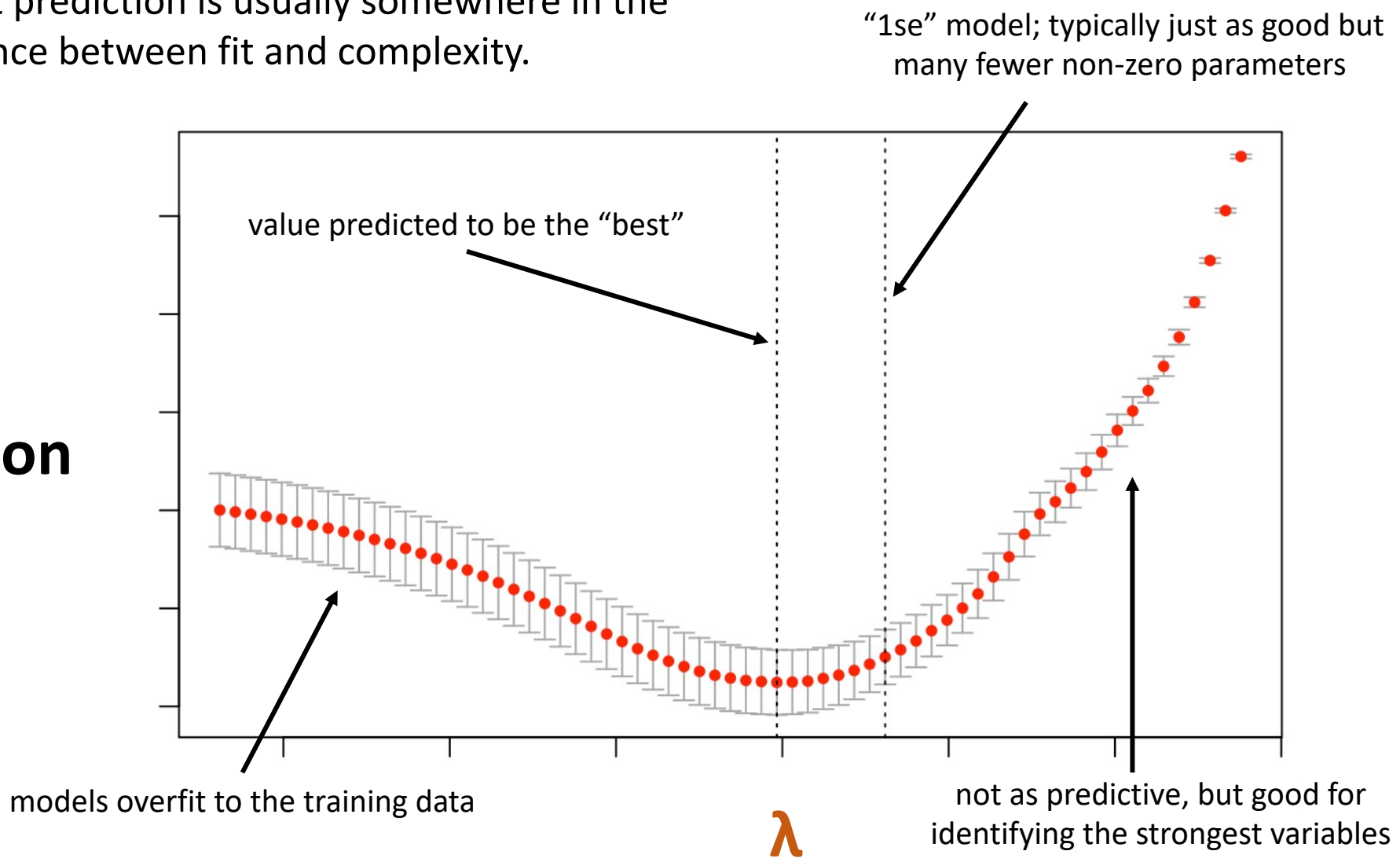
Typically, when we are done we have a curve that looks something like this. The bars give confidence intervals from each of the folds.

Classification Error



Cross Validation

Notice that the best prediction is usually somewhere in the middle, with a balance between fit and complexity.



Elastic Net: Summary

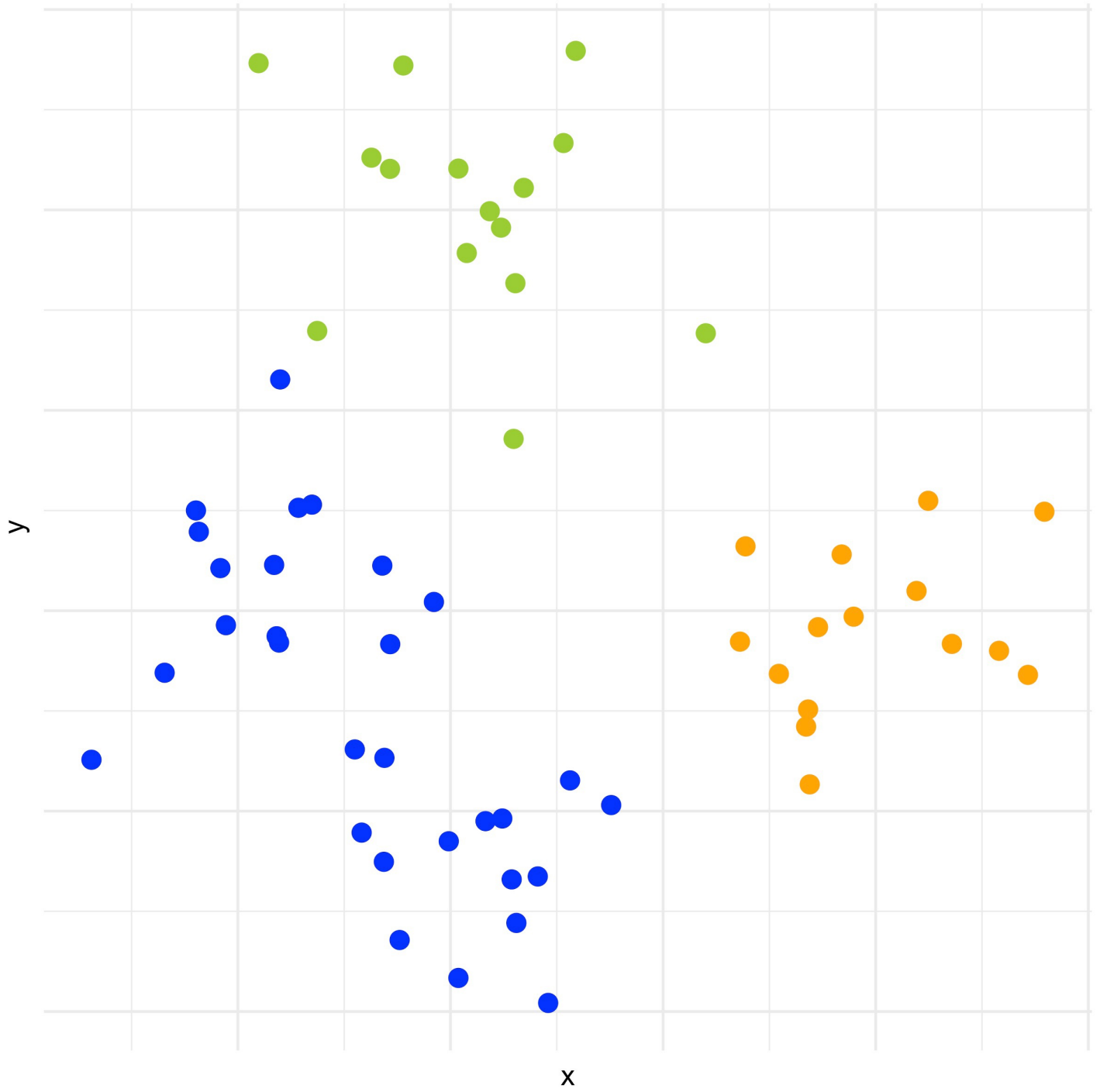
The elastic net will be our most important method for building predictive models this semester. It allows us to do classification with a large number of variables, is able to produce some of the most predictive models in many application domains, and produces interpretable models.

One thing to keep in mind, that is easy to forget, is that the elastic net is a method for producing a linear prediction of probabilities. The only thing that is different is the way that we **pick** the parameters in the logistic model. Once choosen, the model works exactly the same way.

$$\text{Probability}(\bullet) = F(a + b \times X + c \times Y + d \times Z)$$

Multiclass Prediction

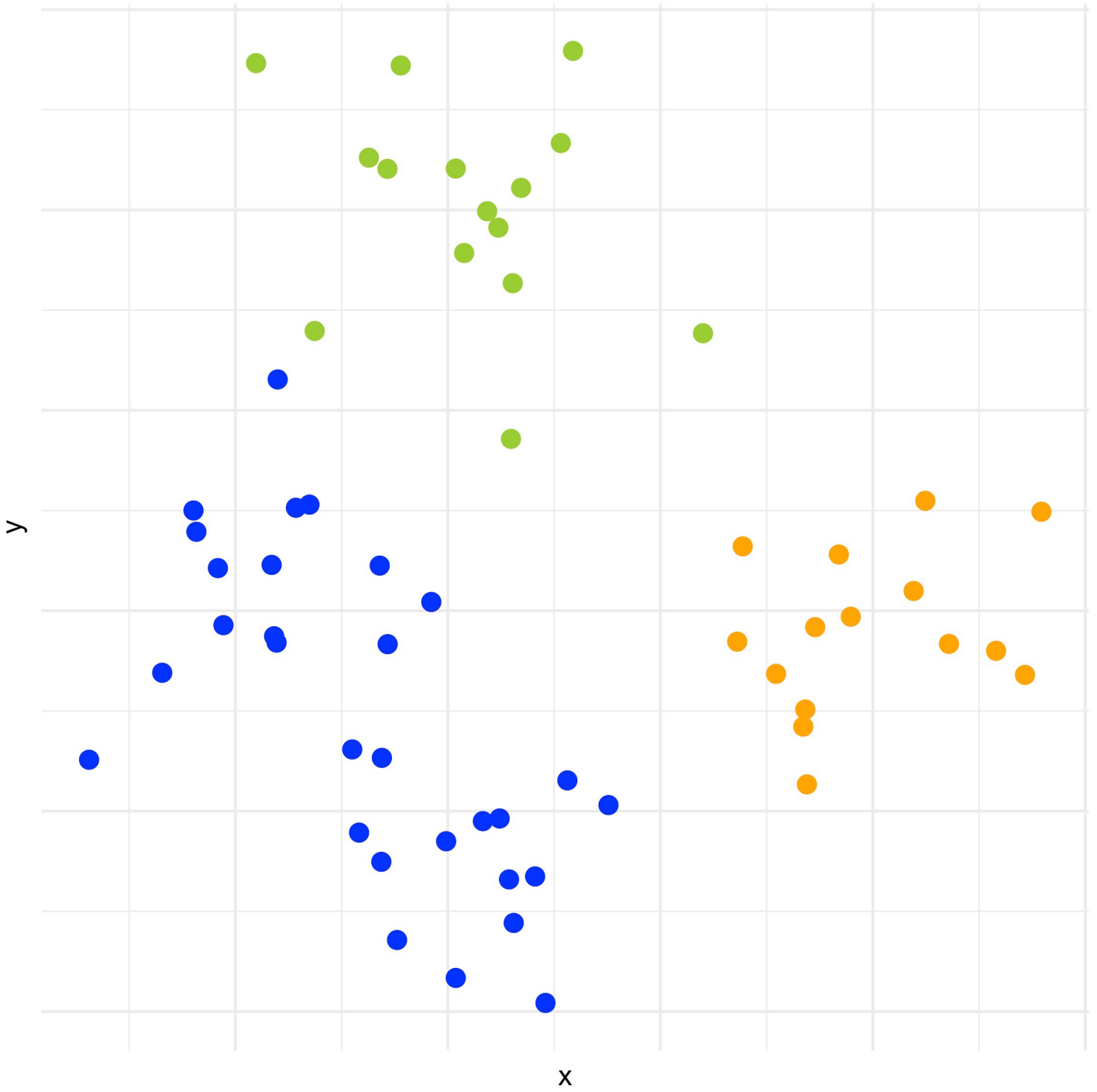
There is one final tweak to the elastic net model that we will need. So far we have only considered the case where there are two classes. Often, though, we will want to consider a prediction task where there are three or more classes.



Multiclass Prediction

The solution is to build one logistic regression model for each class. Prediction is then done using the maximum label probability for each point.

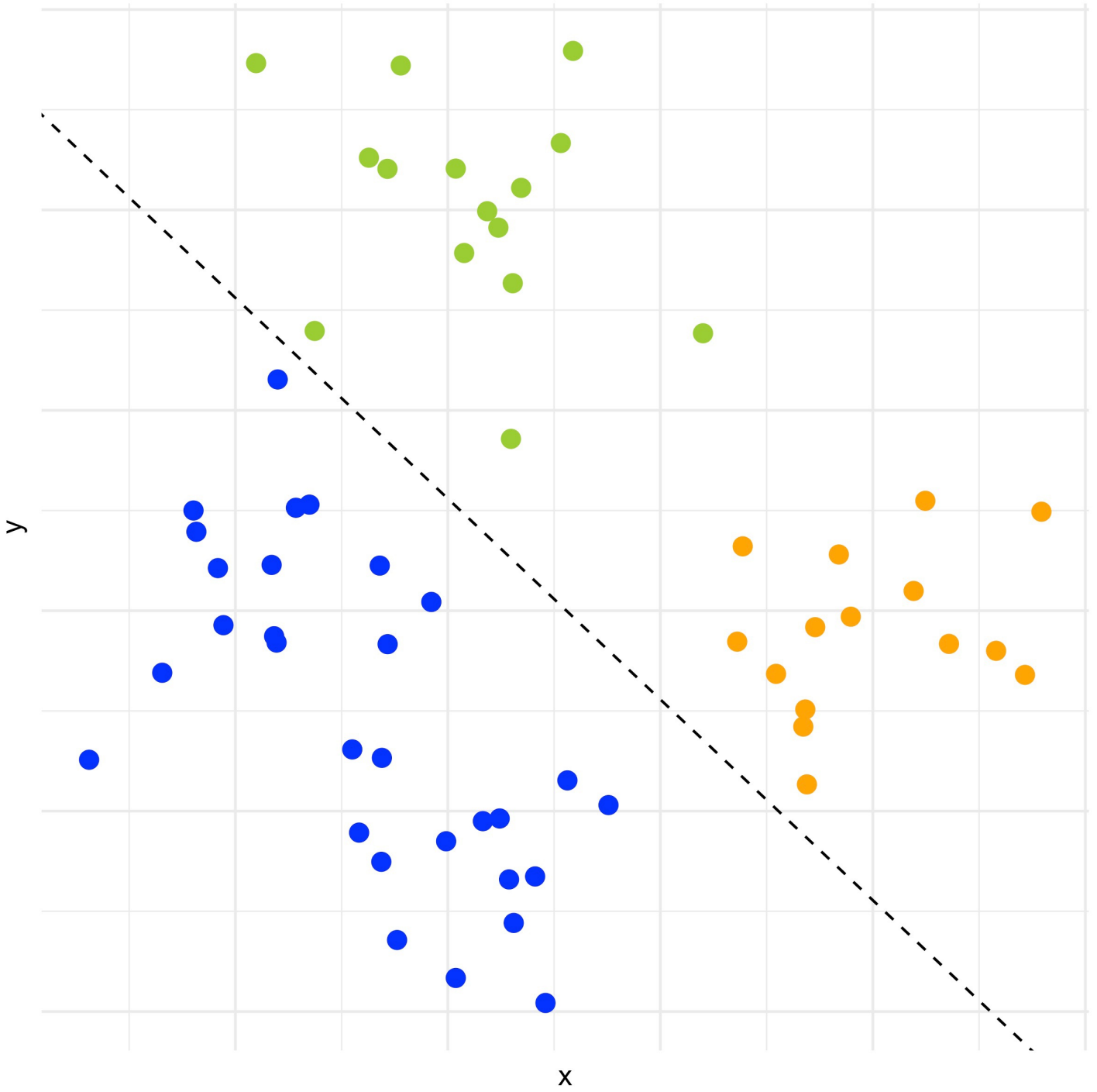
Probability(🔵) = F(**a** + **b** × X + **c** × Y)
Probability(🟢) = F(**d** + **e** × X + **f** × Y)
Probability(🟡) = F(**g** + **h** × X + **i** × Y)



Multiclass Prediction

Here is a line for the blue class (the blue class is below the line):

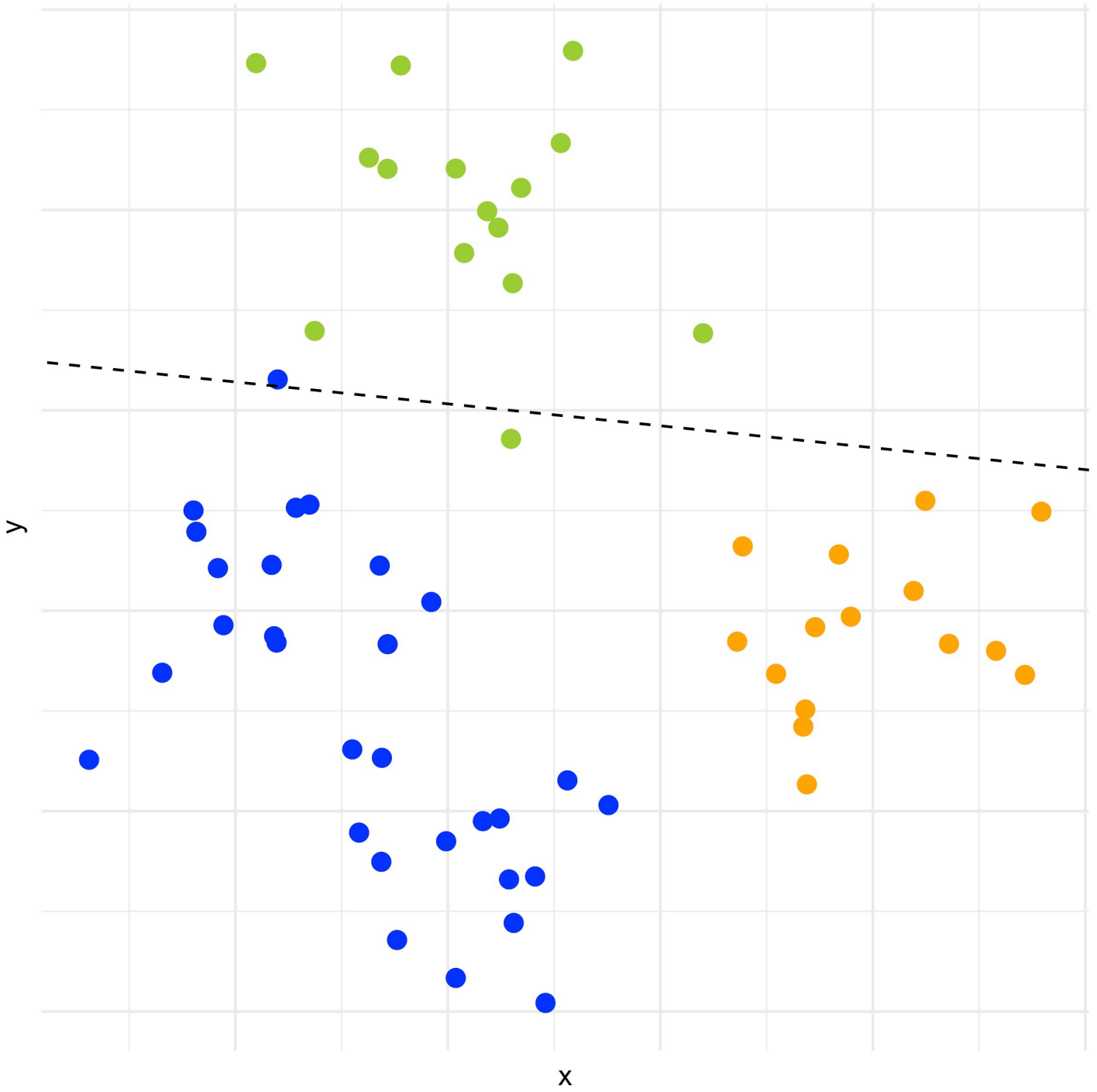
Probability(🔵) = F(**a** + **b** × X + **c** × Y)



Multiclass Prediction

Here's one for the green class:

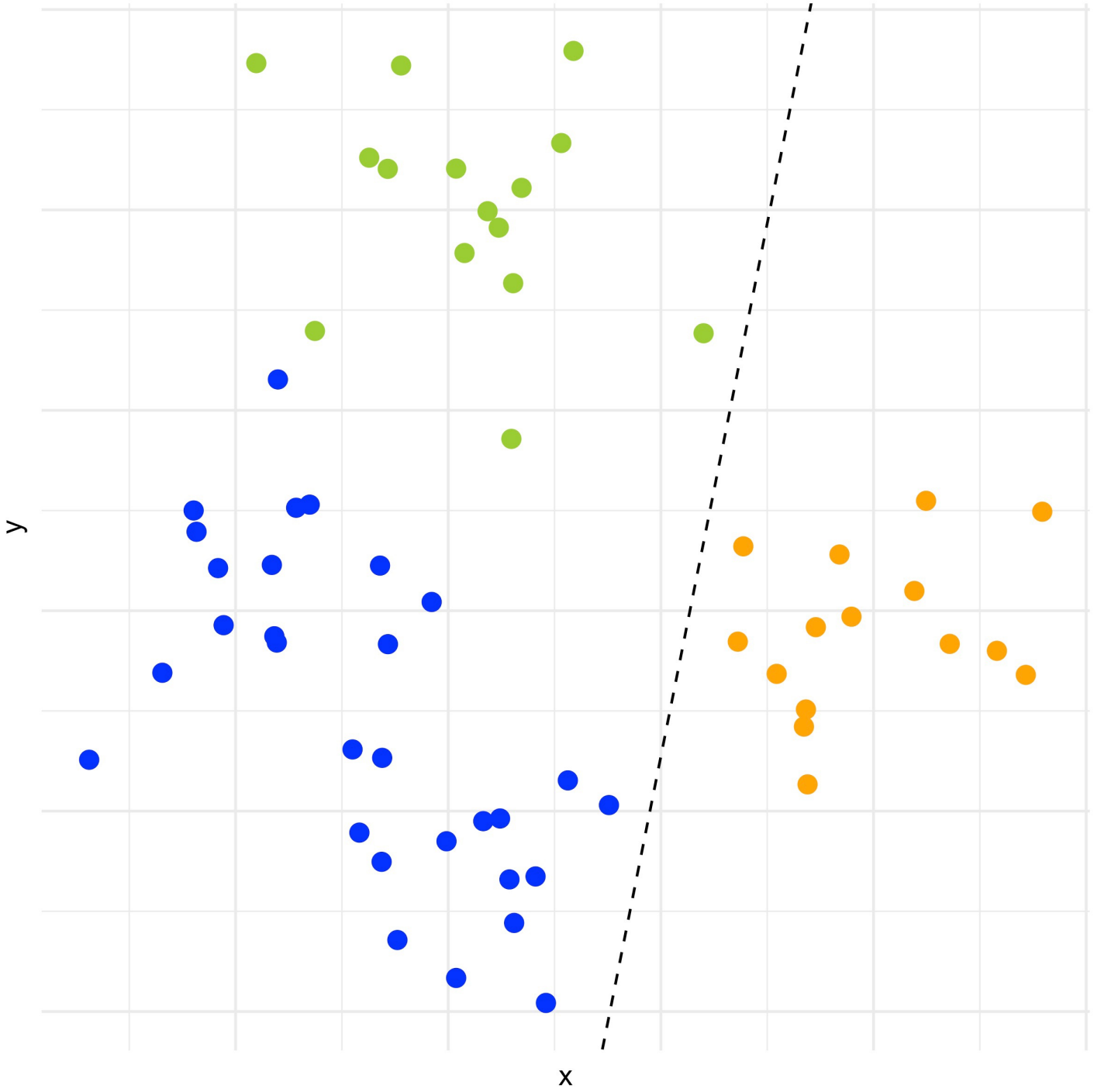
Probability(●) = F(**d** + **e** × X + **f** × Y)



Multiclass Prediction

And finally, one for the orange group.

Probability(🟠) = F(**g** + **h** × X + **i** × Y)



Multinomial Regression

This method is called multinomial regression, in contrast to logistic regression.

Going forward, we will always use multinomial regression, and therefore use the term **elastic net** without referring to whether we are doing logistic or multinomial classification.