Example in Detail: Iris Data

The data that we'll look at is a classic in the literature. This is the iris data that was examined by one of the great statisticians, Sir Ronald Fisher (1936). We will perform a linear classification on the data to see if the data is linearly separable.

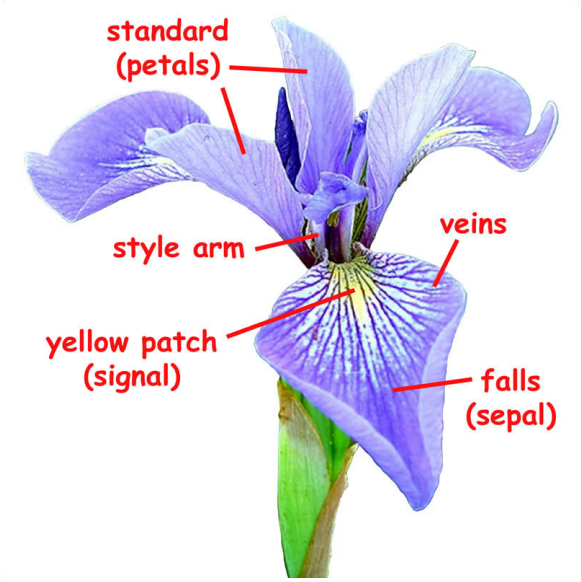

Figure 1: Parts of an iris flower, by Jim Conrad, "Backyard Nature". We are measuring parts of the petals (standard) and sepal (falls).

Description of the data

The data consists of measurements taken from three species of iris (Iris setosa, Iris virginica, and Iris versicolor). The four measurements from each flower are the length and width of the petals, and the length and width of the sepals (the parts of the iris that bend down). We would like to develop a way of distinguishing one species from another based on this data.

The data has already been placed in a Matlab data format, and saved as `IrisDataX.mat`. If you load the data, you will see two variables. Matrix `X` is 150×4, so the data is stored row-wise, and we have 50 samples from each species. That is, the first 50 samples are for one species, the next 50 for the next, and so on.

The Method

We will perform a basic linear classification, but to do that we need some targets. We will take species 1, 2, and 3 as being the three standard basis vectors $[1,0,0]^T$, $[0,1,0]^T$ and $[0,0,1]^T$, respectively.

To get the matrix X to be the same dimensions as our class discussion, you might re-set X so that it is 4×150:

```
X=X';
```

Our next task is to create the desired targets T. You might use the "ones" and "zeros" commands to help you create the 3×150 matrix.

Finally, we will use batch training to get W and $\mathbf{b}$, as we did in class.

```
hatX=[X;ones(1,150)];
hatW=T/hatX;
W=hatW(:,1:4);
b=hatW(:,5);
```

We form the model outputs one of two ways:

```
Y=hatW*hatX;        or       Y=W*X+repmat(b,1,150);
```

A typical output vector will be $[0.98, -0.12, 0.34]^T$- How should we interpret this? A good way to get the classification is to take the index of the maximum value (in this case, 1). Generally speaking, this would be:

```
[vals,idx]=max(Y,[],1);
```

The vector `idx` will be a vector with numbers 1, 2 or 3.

The Results and Analysis

In order to understand the results, we could just record the percent of time that our model is correct, but a better way of reporting results is to use what is called a "confusion matrix". Along the top, we list the actual classes, and along the side, we list the predicted classes. Therefore, if the model predicts a flower to be class 2, but it was actually class 3, then that would be recorded in the $(2, 3)$ position of the matrix.

In general, the (i, j) position refers to something actually in Class j, and predicted to be in Class i. The code below shows us how to construct this.

In General

To keep the coding of this problem simple, we are doing something that we should never do. That is, the process of training (finding the parameters) and the process of testing (building the confusion matrix) should use completely separate data sets. That is, at the beginning of the problem, we really ought to split the data into two sets- One for training, one for testing. Typically, we use about 20% of the data for training, and 80% for testing, but that can vary wildly depending on the problem and how much data you have. That means, that in the beginning, we should randomly split the input and targets so that we have:

```
Xtrain, Ttrain      Xtest, Ttest
```

We'll do this later (Matlab has this built-in to its training routines).

Matlab Code

```
load IrisDataX;
X=X';
[nr,nc]=size(X);

T=[ones(1,50), zeros(1,100);
  zeros(1,50), ones(1,50), zeros(1,50);
  zeros(1,100), ones(1,50)];

hatX=[X;ones(1,nc)];
hatW=T/hatX;

Y=hatW*hatX;              %Output the classifier
[vals,idx]=max(Y,[],1);  %Tell us which class
[vals,idx2]=max(T,[],1); %For comparison purposes

Classes=zeros(3,1);  % Keeps track of how many in each class
Confusion=zeros(3,3);
for j=1:nc
   Confusion(idx(j),idx2(j))=Confusion(idx(j),idx2(j))+1;
   Classes(idx2(j))=Classes(idx2(j))+1;
end

%To get percentages, divide each column by its number of classes:
Confusion=Confusion./repmat(Classes',3,1);
```