Classification

Classification

Industry, academia, and the military all need classification techniques.

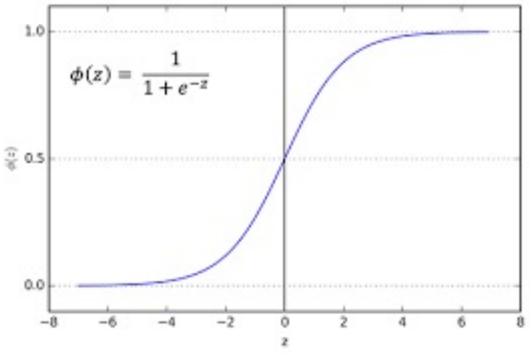
1. A potato chip manufacturer needs to decide if the chips rolling off the line are *perfect*, *overcooked*, or *undercooked*.
2. A tank gunner needs to know *if* a hot spot detected by an infrared sensor is a *friend* or *foe*.

Of all the possible uses for neural networks, <u>classification is probably the most common</u>.

Binary Decisions

The simplest classification problem is the *binary decision*. Note that when the choice is between two classes, deciding where a sample belongs is *not* a binary decision if there is the possibility that it belongs to neither class. A true binary decision admits only two possibilities. Such problems are best served by having exactly one output neuron. The network would be trained to produce a high output activation for one decision and a low activation for the other. A common mistake is to use two output neurons, one for each decision, and train so that one is on while the other is off. This gains nothing in performance, costs memory and training time, and adds ambiguity to the decision process. Many network models use the logistic activation function:

$$f(x) = \frac{1}{1+e^{-x}}$$



Making the Decision

When the trained network is put to the test with an unknown sample, we decide between the two possible outcomes based on the activation level achieved by the single output neuron. If the activation is at least as large as a predefined threshold, we choose the decision for which high activation was trained. In many cases, the threshold is set midway between the two trained extremes. This makes intuitive sense and in practice is usually effective in the absence of any additional information. However, we may want to prejudice the decision by setting the threshold closer to one of the extremes.

We must **avoid** the temptation to introduce a third, **«reject»** decision into what is *supposed* to be a strictly binary decision by setting two arbitrary thresholds. For example, suppose the network has a logistic activation function, and we have trained for output values of 0.1 and 0.9. We may be tempted to make one decision if the output is at least 0.7, make the other decision if the output is less than 0.3. and consider it undecided if the output lies between 0.3 and 0.7.

Making the Decision

The only reason to use this **two-threshold policy** is in the **validation phase**, as **it indicates inadequate training**. If that case is definitely a member of one category, it should be included in the training set for that category, and training repeated. If it is not a member of either category, the choice of a strict binary classification model must be brought into question. If we are *positive* that there are only two possible categories, we can often use the **confidence measures** we will talk about it later in the semester (6th week) to tell us approximately ***how confident we can be in our decision***.

Multiple Classes

A multiple-class model not only allows for several class decisions, but also allows for the possibility of a «reject- decision». Even if only two classes are possible, a multiple-class model must be used if we want to include the possibility of failing to classify into either of the two categories. As was pointed out in the previous section, **failing to identify clearly a winner in a single-neuron binary model may be more an indication of inadequate training than it is of failure to fall into either category**.

The generally best way to implement a multiple-class model **is to use a separate output neuron for each class.** Training is done by requiring the neuron corresponding to the class being presented to be highly activated, while all other neurons are required to be nearly off. Thus, for each training example, exactly one output neuron will be trained to be on. The only exception is if we are able also to train for a reject category. Samples from the reject category would require all output neurons to be nearly off.

Reject Category

It can sometimes be helpful to include samples from a reject category in the training collection. Since we will reject an unknown sample if all output-neuron activations are below a preordained threshold, we can help the network to reach this rejection decision when appropriate by giving it examples of when to keep all output neurons inactivated. However, we must beware of a hidden danger. The network cannot be expected to reliably reject cases that do not have exemplars in the training collection. It is all too easy to include only a partial representation of what is to be rejected, then become overconfident.

In particular, the decision rule is:

1) **If all output neuron activations are less than a specified threshold, reject the unknown sample**.

2) **Otherwise, classify the unknown into the category whose corresponding output neuron has the highest activation. In the unlikely event of a tie, break it randomly.**

Other Encoding Schemes

The traditional encoding scheme in which each class has its own dedicated output neuron has many advantages. **Perceived similarity** of a sample to each class can be assessed by the degree of activation of that class's neuron. If one or more output neurons are activated nearly as high as the winner, we know that there is the possibility of confusion. Sometimes we can even estimate confidence for that case.

The task of a neural network is to grade fruit into one of four categories:

A: Premium Quality, destined for select consumers

B: Good quality, for grocery stores

C: Poor quality, for juice

D: Reject

Other Encoding Schemes

The classes clearly have an order relationship, so using just one output neuron is appropriate. In fact, using four separate output neurons could lead to interpretational difficulties.

When the categories have an order relationship, we must decide what activation levels to train for. The simplest approach is to space them equally. In the above example of grading fruit, it would be reasonable to train the network so that samples from **grade A activate the neuron to 90 percent activation**, **B to 63 percent**, **C to 37 percent,** and **D to 10 percent**. In some instances, there may be reasons for un-equal spacing and, if so, there is no reason for demanding equal spacing. On the other hand, ad hoc decisions are always dangerous. In the absence of obvious justification, avoid subjective choices of activations.

Another situation in which separate output neurons for each class may not be best **is if the classes are defined by two or more binary (or other low radix) variables**. One example of this would be blood typing. Three of the medically significant factors that may or may not be present in human blood are the A, B and Rh components. Blood that contains neither the A nor the B components is said to be type O. Otherwise, its type is indicated by the letters A and\or B. If the Rh factor is present, the blood is called positive; otherwise, it is negative. Thus, blood that contains all three factors is AB positive, and blood that contains none of them is 0 negative. There are **8 possible blood types**.

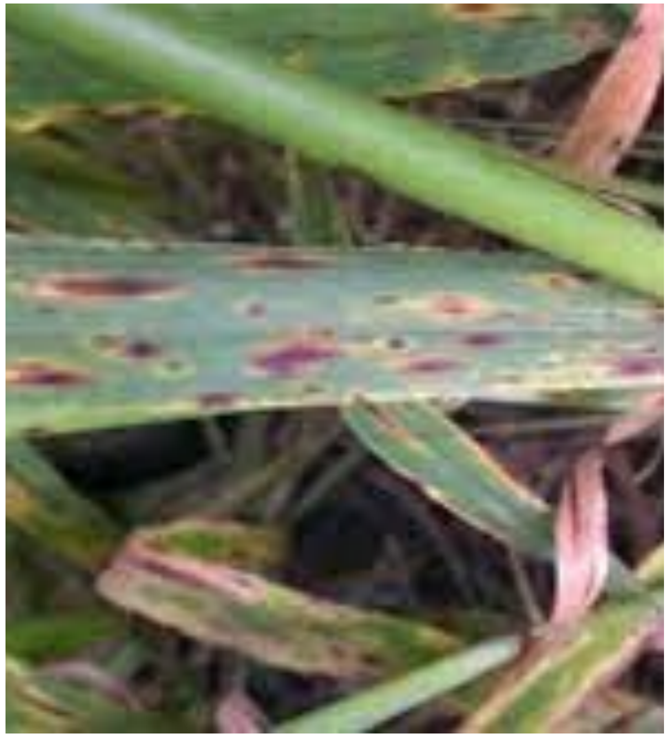
Other Encoding Schemes

Suppose that our network's input is photographs of all three samples. It would be wasteful to use eight output neurons, one for each type. **A much more sensible approach would be to use just three neurons, one for each factor. In fact, it may be faster to train three separate one-output-neuron networks than one three-output neuron network**. In problems of this type, it is better to use the more compact binary encoding for output classes.

Perhaps an employee of some government is pulling samples of high-altitude photographs of crops from a satellite transmission. These crop photos are fed to a neural network, which must decide the type of crop (<u>corn, wheat, soybeans</u>, et cetera) and whether the crop is diseased. It would be tempting to allocate one output neuron for each type of crop and then just one neuron for the health-disease question. But this is not a separable decision. **The manifestation of disease could be different for each crop**. **A far better approach would be to have two (or more, if a variety of diseases is possible) output neurons for each crop type.**

Other Encoding Schemes

One neuron would be activated for healthy corn, another for diseased corn, and so forth. Many neurons would be required. But training time and performance would most likely be **better than if one neuron were dedicated to the entire range of crop diseases**.



Supervised versus Unsupervised Training

It is possible to let neural networks discover salient characteristics of training data on their own. Models that are capable of *unsupervised training* can be especially valuable in exploratory work. Sometimes the researcher hypothesizes that distinct classes exist in a collection of samples but is not sure what those classes are. Another possible situation is that the data falls into defined classes, but it is suspected that some of the classes contain several subclasses.

Identification of these subclasses can not only aid in understanding the problem but can often lead to improved performance by suggesting restructuring of the training set.

Summary

1. Binary decisions need exactly one output neuron. Do not use two seperate output neurons.
2. Use confidence measures to show how much confident we are in our decision.
3. With one output neuron, do not introduce a third decision, *reject*, by using two arbitrary thresholds.
4. In multiple classes, reject the input sample if all the outputs are less than a specific threshold. If there are multiple ones that are higher than that threshold, then select the *highest* one. If they are the same, then choose the one class *randomly*.
5. If there is an order relationship, then using one output neuron with arbitrary thresholds will be the best solution.
6. If the classes are defined by two or more binary variables, then use seperate output neurons for each class.
7. Unsupervised learning can be used to find the distinct classes.