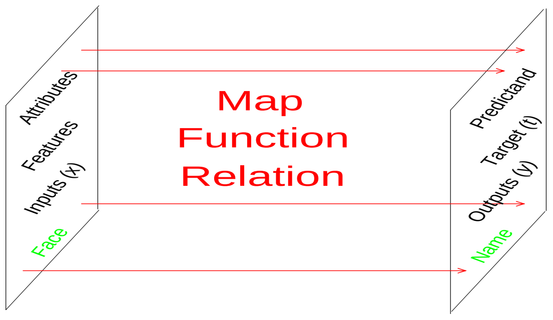
CLASSIFICATION (DISCRIMINATION)

Caren Marzban
http://www.nhn.ou.edu/˜marzban

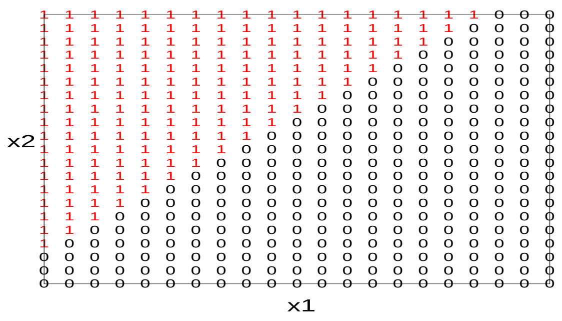
Generalities

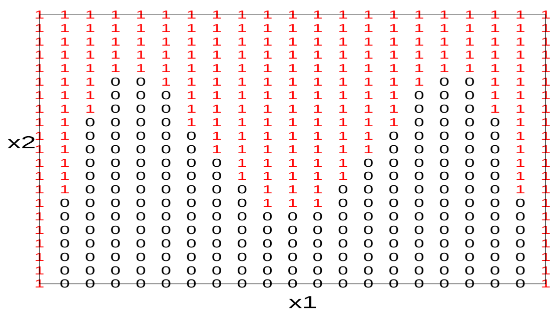
This lecture relies on the previous lecture to some extent. So, read that one first. Also, as in the previous lecture, what you are currently reading is an expanded version of what is presented at the lecture.

Recall, that the problem at hand is to estimate the weights that parameterize the relationship between a set of input variables and a set of output variables (or targets). Actually, we have to decide on the parameterization, too, since the map also depends on H, the number of hidden nodes.



In the previous lecture we considered the regression problem, where the targets are continuous variables. Now, we will see how to handle the case where the targets are discrete, with each value of the target representing some class of objects. For example, a tornado classification problem could consist of finding the map that relates some set of input variables to, say, two classes - the class of non-tornadoes, and the class of tornadoes. This way we can develop a model that can "predict" whether a set of inputs values corresponds to a tornado or not. To be more precise, the map or the function we are looking for in a classification problem is the (decision) boundary that best demarcates the regions of classes. The figure below shows examples of a linear boundary (left) and a nonlinear boundary (right) for a "noise-free" problem.





Discriminant Analysis

A traditional classification model, called Discriminant Analysis, is an enlightening ancestor of NNs. Let's look at the simple case of one input variable, x, and two classes labeled as $i = 0, 1$ (e.g., non-tornado, and tornado). The model begins by assuming that the likelihood of x, when it happens to belong to class=i, is gaussian. In other words, the conditional probability of x, given class=i, is written as

$$L_i(x) \sim e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \sim prob(x|C = i).$$

where μ_i and σ_i are the mean and standard deviation of x when it belongs to the i^{th} class. Both are estimated from a training set. Note that $L_0 + L_1 \neq 1$.

For forecasting purposes, this likelihood is the wrong probability to look at. We are not interested in the probability of finding, say wind speed = 50 Km/hr, given that there is a tornado on the ground. We want to know the probability that there is a tornado on the ground, given that wind speed = 50 Km/hr. This other probability is called the posterior probability, and the laws of probability imply that it is given by

$$P_i(x) = \frac{p_i L_i(x)}{p_0 L_0(x) + p_1 L_1(x)},$$

where p_0, p_1 are called prior (or climatological) probabilities for the two classes. They are estimated as N_0/N, and N_1/N, respectively, where N_i is just the sample size of the i^{th} class, and $N = N_0 + N_1$ is the total sample size. Note that $p_0 + p_1 = 1$ and $P_0 + P_1 = 1$.

Then, when we observe some value of x, if $P_1(x) > P_0(x)$, then we forecast (or classify) it as a 1 (i.e., a tornado). Actually, instead of $P_1(x) > P_0(x)$, it's more convenient to look at $\log(P_1(x)/P_0(x)) > 0$, because we can then write the left-hand side as

$$\log(P_1(x)/P_0(x)) = \frac{1}{2} D^2(x),$$

where the so-called discriminant function, $D^2(x)$, is given by

$$D^2(x) = (\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2})x^2 - 2(\frac{\mu_0}{\sigma_0^2} - \frac{\mu_1}{\sigma_1^2})x + (\frac{\mu_0^2}{\sigma_0^2} - \frac{\mu_1^2}{\sigma_1^2}) + 2\log(\frac{\sigma_0}{\sigma_1}) - 2\log(\frac{1-p_1}{p_1}).$$

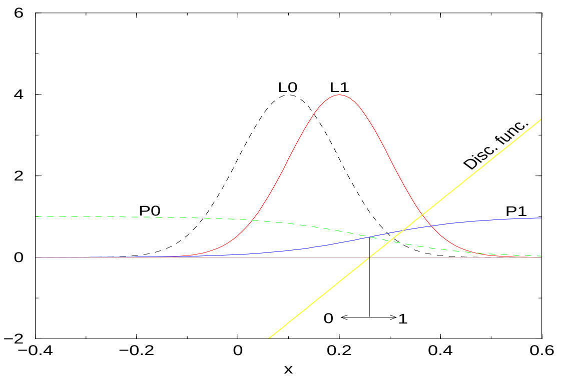
This equation follows in a straightforward way by just combining the above equations (homework).

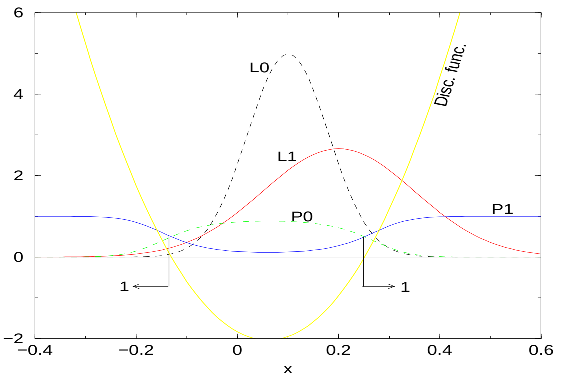
Again, an observation x from either training set or validation set is forecast as (or assigned to) class 1 (e.g., tornado), if $D^2(x) > 0$, otherwise it is classified (forecast) as a 0 (e.g., non-tornado).

In spite of its simplicity, this is a very powerful classification model. But as seen from the last equation, the best it can do is to handle quadratic (i.e., x^2) decision boundaries. It will fail if the boundary underlying our data is more nonlinear than that. Of course, the source of this limitation is the starting assumption - normality.

I urge you to consider using this model, because you may be surprised by how well it does. But even if you never see or hear about this model again, one lesson that you should not forget is that the distribution of the data is related to the nonlinearity of the boundary. In the above example, if the distributions are gaussian, then the boundary is quadratic. As a special case, if the distributions are gaussian and equivariant (i.e., $\sigma_0 = \sigma_1$), then the boundary is linear. All of this should be reminding you of the close relationship between gaussian noise and mean square error in linear regression (previous lecture).

The following figures show all the above ingredients in both the linear and nonlinear case. Just study it a bit; I'll go over it during the lecture. It will even be in color then.





NN

Discriminant Analysis provides a stepping stone for moving on to classification NNs. First, notice that $P_1(x)$ can be re-written in the form (homework)

$$P_1(x) = \frac{1}{1 + e^{-\text{discrim. func.}}} .$$

But this is the form of the logistic function discussed previously. In fact, for the case where the discriminant function is linear, it is exactly the logistic function. And that is nothing but an NN with no hidden nodes.

So, you can see that an NN *with* hidden nodes -

$$y(x, \omega, H) = g\left(\sum_{i=1}^{H} \omega_i \; f(\sum_{j=1}^{N_{in}} \omega_{ij} x_j - \theta_j) - \omega \right) \quad ,$$

is a generalization of Discriminant Analysis. Then, you should not be surprised to find out that an NN with enough hidden nodes can fit any nonlinear boundary. Additionally, you should not be surprised to find out that the output of an NN represents posterior probability.

To strengthen the probabilistic interpretation of NN outputs, in a beautiful paper, Richard & Lippmann (*Neural Computation*, **3**, 461-83, 1991) showed that "Neural network classifiers estimate Bayesian a-posteriori probabilities." In fact, that's the title of their paper. The exact statement is this: If the activation functions f and g are logistic, then the minimization of cross-entropy -

$$E = -\frac{1}{N} \sum [t \log y(x, \omega) + (1 - t) \log(1 - y(x, \omega))] \quad ,$$

where the target takes values $t = 0, 1$, guarantees

$$y(x, \omega) = P_1(x) \qquad (= prob(C = 1|x)) \quad .$$

This is a very important result because it tells you what you need to do in order to have an NN that makes probabilistic forecasts. Again, this result should remind you of the analogous regression result - that the minimization of MSE implies that the NN outputs will be conditional averages: $y(x, \omega) =< t|x >$. So, whatever NN simulation you employ to perform classification, instruct it to use cross-entropy as the error function, and to use logistic function for all the activation functions. By the way, for more than two classes, the generalization of the logistic activation function that preserves the probabilistic interpretation of the outputs is called the softmax activation function.

Method

As I said, we still have to worry about all the things we worried about in the regression lecture: Weight-decay, local minima, bootstrap (or something like it), etc. Then, fill in the big table in the previous lecture with different Seed and Bootstrap Trials. The table will allow you not only to identify the optimal number of hidden nodes, but also to place error bounds on the performance of the NN.

One difference with the regression NN is that the choice of the error function is a little less ad hoc, in the sense that only cross-entropy (with logistic activation functions) will yield probabilistic outputs.

Another difference is that you *may* have to worry about the representation of the different classes in the training set. For example, if one of the classes is drastically under-represented in the training set (compared with the other classes), then the NN *may* just treat the cases in that class as noise. It is this concern that often prompts some people to "balance" the classes in the training set. This is actually not recommended, because the outputs will no longer represent posterior probabilities. For the outputs to have that probabilistic interpretation, it is crucial for the different classes in the training set to be represented exactly proportional to their respective prior (or climatological) probabilities. So, it's better not t0 dabble with the class sizes. Having said that, there will be times that the above concern will in fact be realized. In that case, it is possible to practice that balancing act, if you assure that the outputs are scaled in such a way to resurrect their probabilistic interpretation. In particular, you would have to multiply each output node by the prior probability of the class designated by that output node, and then renormalize to assure that all the probabilities add up to 1.

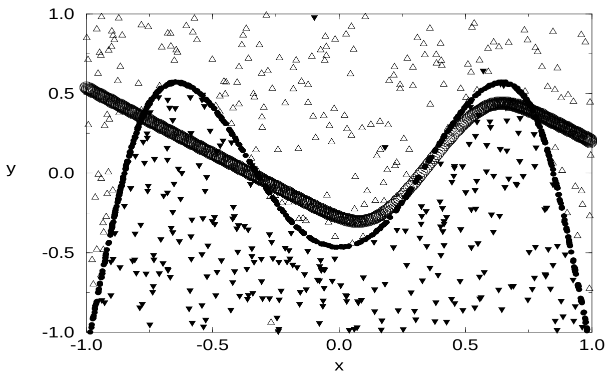
One final difference is in assessing the performance of a classification NN, especially if the outputs are arranged to be posterior probabilities. Because then you are dealing with probabilistic forecasts, and if you are a meteorologist you probably know that the verification of probabilistic forecasts is a topic that requires a whole book. I'll discuss some of the important concepts below, and in my next lecture. But at least check out the following two classic papers by Murphy, A. H., and R. L. Winkler: *Int. J. Forecasting*, **7**, 435-455 (1992), and *Mon. Wea. Rev.*, **115**, 1330-1338 (1987). Wilks' book is good to have as well (Wilks, D. S., 1995: *Statistical Methods in the Atmospheric Sciences.* Academic Press, NY.)

Overfitting

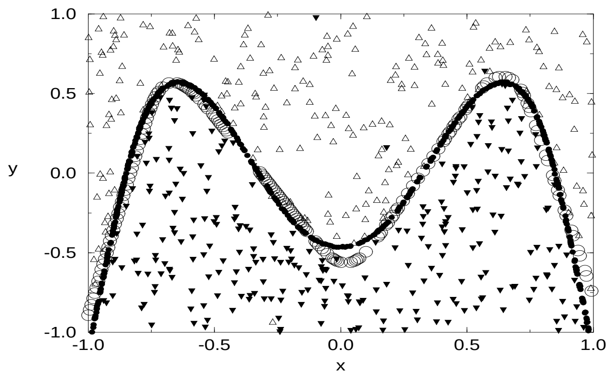
In practice, we still have to worry about all the nuisances we talked about for the regression NN. Overfitting (or the optimal number of hidden nodes) is the first concern. The following figures illustrate overfitting at work. I have concocted a noisy version of the classification example I showed in the beginning of the lecture. The boundary is the same inverted Mexican hat as before (labeled with light circles), but it's noisy in the sense that a few data points are on the wrong side of the boundary. Instead of 1 and 0, I've used filled and unfilled triangles. So, some filled triangles are above the boundary and some unfilled triangles are below the boundary.

I've trained a series of NNs with different number of hidden nodes: $H = 2$, 4, 6, 8, 10, and 15. The corresponding decision boundary is shown with the dark line. You can see that the $H = 2$ NN underfits the boundary. $H = 4$ does better, but $H = 6$ is already overfitting in the sense that it thinks the single filled triangle at the top of the upper region is a real data point, rather than noise, which it clearly is. Larger NNs, with $H = 8$, 10, and 15, just continue to overfit even more. So, $H = 4$ is optimal.

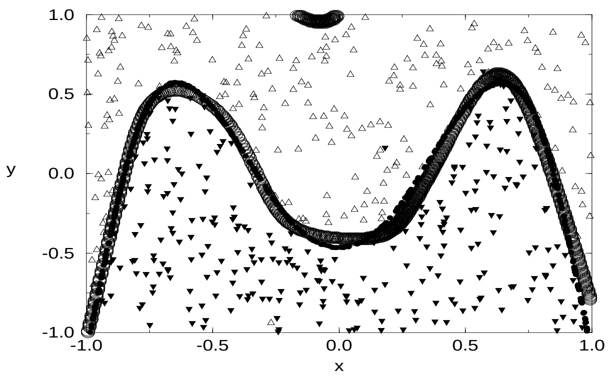
$H = 2$



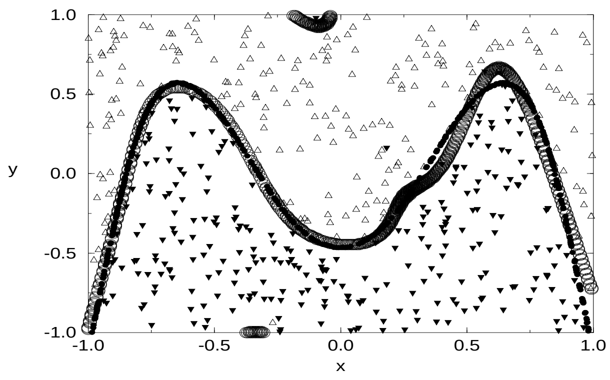
$H = 4$



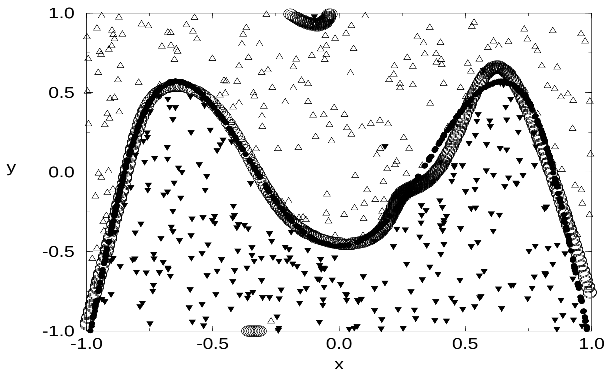
$H = 6$



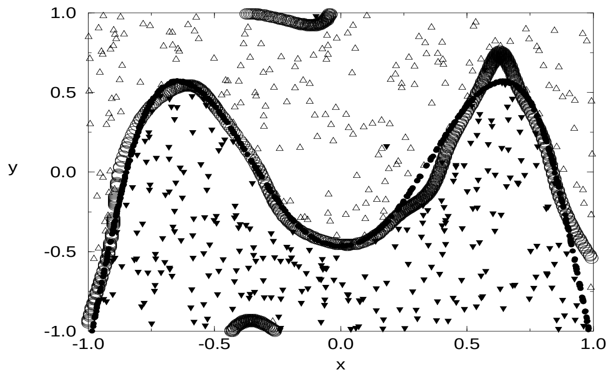
$H = 8$



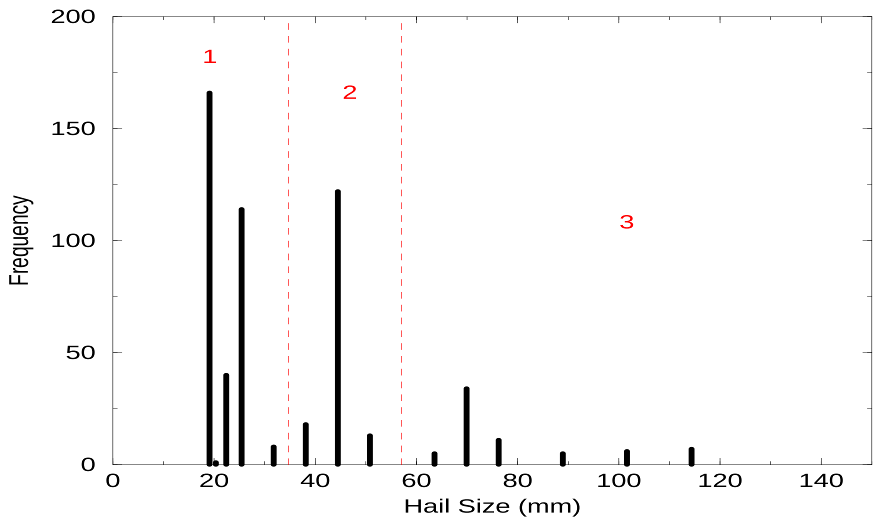
$H = 10$



$H = 15$



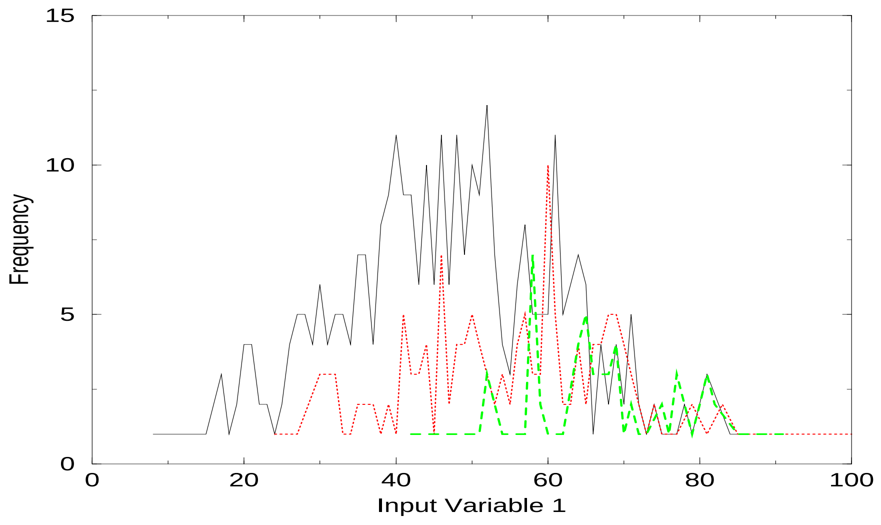
Practical Application - Hail Size



Treating hail size as a continuous quantity, we developed an NN that "predicts" hail size. Given the 3 peaks in the distribution of hail size (above), it is natural to develop an NN that "predicts" the size *class* to which a set of input values belong. In fact, it would be useful to find the *probability* of getting hail size of a certain class?

The reason for the three peaks in the distribution is mostly human related. It turns out there are two types of meteorologists - those who report hail size in millimeters (or hundredth of an inch), and those who classify hail as being coin size, golf ball size, or baseball size.

The preprocessing for a classification NN is mostly the same as for a regression NN. Excluding collinear inputs, removing outliers, transforming to z-scores, etc. One difference is that consideration has to be given to the class. For instance, instead of examining the distribution of input variables (like we did in the regression NN), we must look at the class-conditional distribution of the input variables. That means that you must plot the distribution/histogram of the input variables separately for each class. The figure below shows this for one of the input variables, for the 3 classes of hail size. As expected, the distributions are bell-shaped, and shift to the right as the size of the corresponding class increases.



Similarly, in identifying collinear inputs, instead of calculating the r of two inputs, you should look at the r of two inputs for each class separately. It's possible that two inputs are highly collinear when they belong to one class, but not at all collinear in another class. Clearly, you would not want to exclude a member of such a pair, at least not based on their collinearity.

Going through the previously outlined method, we find the following optimal values: The number of inputs nodes = 9, and the number of hidden nodes = 4. The number of output nodes is a bit tricky; it depends on how we choose to encode the class information in the target nodes. For instance, it's possible to have one output node, with values 0, 0.5, and 1 representing the 3 classes. And if the three classes have an "order" associated with them, then that's probably fine. A better way - and one that preserves the probabilistic interpretation of the outputs - is to use what's called 1-of-c encoding. In that scheme, there is one output node for each class, and the larger one designates the class. During training, for class 1 cases, the three output nodes are assigned the values (1,0,0) as their target values; class 2 cases are assigned (0,1,0) as their corresponding output nodes, etc.

Performance

This section is a little long, so make yourself comfortable. And, no, it could not have been placed as an appendix because its content is crucial for properly assessing the performance of an NN.

Performance measures come in a variety of types, depending on the type of forecast (and observation). For example, if the forecasts are categorical, then one may have a scalar measure of performance, or a multidimensional measure (like a diagram). Similarly, probabilistic forecasts can be assessed in terms of scalars and/or diagrams. The following table lists an example of each type.

	Scalar	Multi-dimensional
Categorical	Equitable Threat Score	ROC Diagram
Probabilistic	Ranked Probability Score	Attributes Diagram

ROC stands for Receiver's Operating Characteristic, and will be defined below. Needless to say, multidimensional measures carry more information than scalar ones; similarly, probabilistic ones contain more information than categorical ones. Then again, they all have their place in the world.

At one extreme we have categorical forecasts and observations, like 3 hail classes being classified (forecast) into 3 classes. The starting point for assessing the performance for such a classifier is the contingency table:

$$\text{C-table} = \begin{pmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{pmatrix},$$

where n_{12} is the number of class 1 cases (incorrectly) classified as belonging to class 2. Etc. In my (nonuniversal) convention, the rows correspond to observations, and the columns represent forecasts.

There is an infinity of ways to construct scalar measures from the 3×3 contingency table, but almost no multidimensional measures. By contrast there are numerous multidimensional measures that can be constructed from a 2×2 contingency table. For that reason, the problem of classifying three classes - labeled 1, 2, and 3 - is broken down to three 2-class problems: classification of class 1 or otherwise, class 2 or otherwise, and class 3 or otherwise. Although this treatment of a 3-class problem can lead to some loss of information, the loss is partially compensated by the availability of multidimensional measures.

Then, performance can be represented with three 2×2 contingency tables:

$$\text{C-table} = \begin{pmatrix} n_{11} & n_{12} \\ n_{21} & n_{22} \end{pmatrix},$$

A couple of common measures derived from this table are are the Probability of Detection and False Alarm *Rate* (not Ratio - that's something else). If the class we are trying to detect is the one labeled 2, then

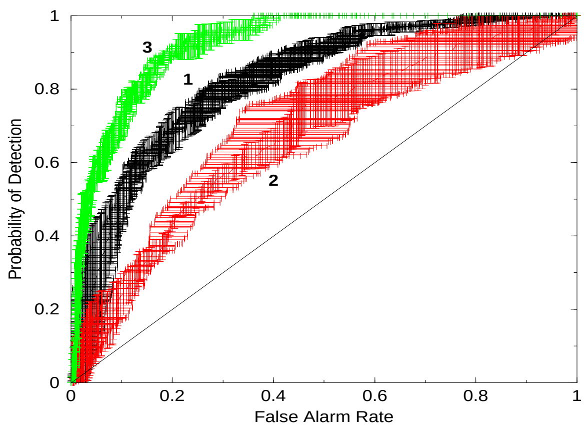
$$\mathrm{POD} = \frac{n_{22}}{n_{21} + n_{22}}, \quad \mathrm{FAR} = \frac{n_{12}}{n_{11} + n_{12}}.$$

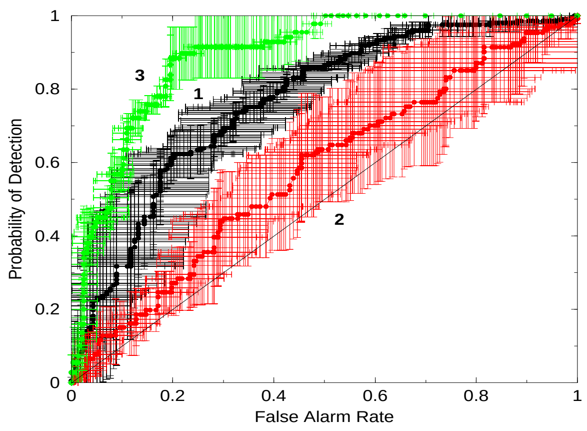
Note that the total number of class 1 cases is given by $N_1 = n_{11} + n_{12}$, and that of class 2 cases is $N_2 = n_{21} + n_{22}$. The total sample size is $N = N_1 + N_2$.

Even though, our NN produces probabilistic forecasts, it is possible to categorize the forecasts by thresholding them; you simply pick a number between 0 and 1 (because probability lives in that range) and then say that any forecast less than that number belongs to class 1, and any forecast larger than the threshold belongs to class 2. In this way, one can reduce a continuous quantity like probability into a categorical one. Then, the question is which threshold to pick. The ROC diagram handles that problem.

In fact, an ROC diagram is a parametric plot of POD versus FAR, as the threshold is varied from 0 to 1. So, you pick a threshold, but you don't have to stick to it; you vary it from 0 to 1 in some increments, calculating POD and FAR at each increment, and then plotting them on an x-y plot. It is easy to show that a classifier with no ability to discriminate between two classes yields a diagonal line of slope one and intercept zero; otherwise it bows above the diagonal line. (The area under the curve is often taken as a scalar measure of the classifier's performance, and so, a perfect classifier would have an area of 1 under its ROC curve, while a random classifier would have an area of 0.5. In addition to its multidimensionality, another virtue of a ROC diagram is its ability to express performance without a specific reference to a unique threshold. A specific choice of the threshold calls for knowledge of the costs of misclassification which are user dependent. In this way, a ROC diagram offers a user-independent assessment of performance.

For the Hail problem, the ROC plots are shown below. The left diagram is for the training set and the right one is for the validation set. (The error bars are the standard error from the bootstrap trials.) It can be seen that performance of class 3 and class 1 forecasts is quite high (given by the amount of bowing away from the diagonal). However, class 2 forecasts are not doing too well, since according to the validation ROC plot, they are almost as good as random (evidenced by the overlap of the error bars and the diagonal line). So, the NN does good in "predicting" the largest and the smallest of the hail classes (in that order), but not too well with the midsize class. If you think about it, it makes sense that the midsize class would be the hardest one to discriminate from the others.



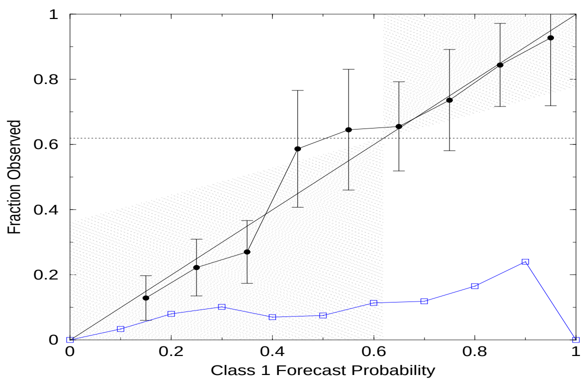


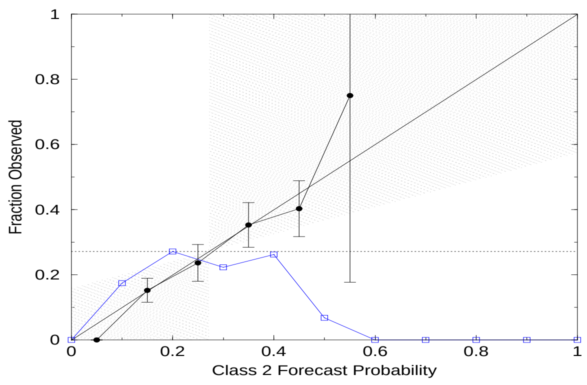
Of course, it is possible to assess the quality of the forecasts without categorizing the forecasts. Murphy and Winkler (referenced, above) have shown us how to do that. They show that the various aspects of the quality of probabilistic forecasts can be captured by three diagrams - reliability (or calibration), refinement, and discrimination diagrams. (A slightly improved version of the reliability diagram is an attributes diagram; it shows whether or not a given forecast actually contributes to the skill of the classifier). These are all computed from the joint probability of forecasts and observations, but I'll give a simpler description here. I'll return to the joint distribution in the next lecture.

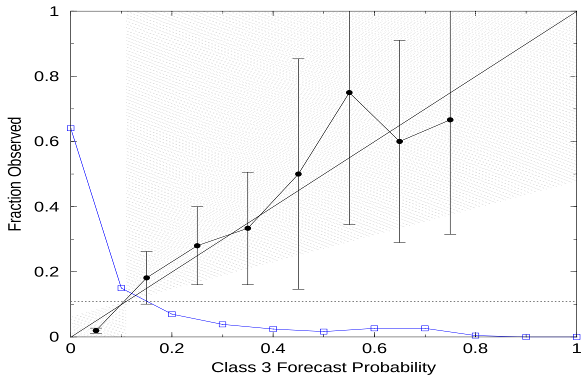
A reliability diagram displays the extent to which the frequency of an event at a given forecast probability matches the actual forecast probability. To construct it, take all the cases that have a forecast of 10%. Mark this number on the x-axis. What fraction of the whole data is the number of cases in this set? Mark that answer on the y-axis. We now have one point on the x-y plane, which is hopefully somewhere near the diagonal, because a 10% probability is supposed to mean that 10% of the time we get that outcome. Then repeat this for forecasts that are 20%, 30%, ..., 90%. All of the resulting points should lie somewhere near the diagonal; the closer to the diagonal, the more reliable the forecasts.

A refinement diagram displays the sharpness of the forecasts, i.e., the extent to which very high or very low probabilities are issued. To construct it, just plot the histogram of all the forecasts. There are many ways in which a classifier can be poor in terms of refinement. For example, a classifier that produces only two forecasts, say 10% and 90%, has poor refinement. A good classifier would produce a range of all forecasts (between 0 and 100%).

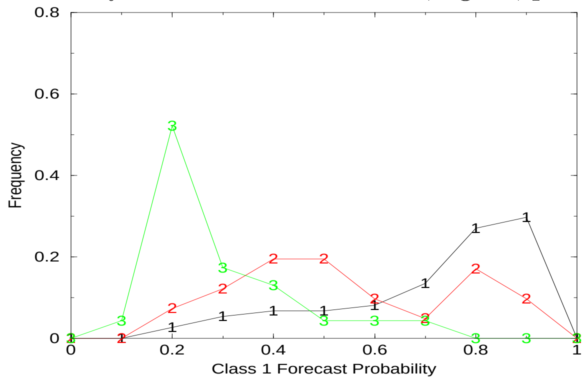
I put reliability and refinement plots on the same diagram (below). Again, the error bars are the standard error from the bootstrap trials. The diagonal line corresponds to perfect reliability. We can see that within the error bars the NN produces forecasts that are perfectly reliable for all three classes. The curve (blue) not along the diagonal is the refinement plot. The class 1 forecasts have a very reasonable refinement plot in that they are relatively flat in the full range of 0 to 100%. The refinement plot for class 3 forecasts suggests that they never exceed 60%; below 60% they are reasonably refined. Class 3 forecasts are reasonably refinement as well; the peak at 0% reflects the fact that class 3 cases are rare. The shaded regions are what make these attributes diagrams. See Wilks book (referenced above) for details.

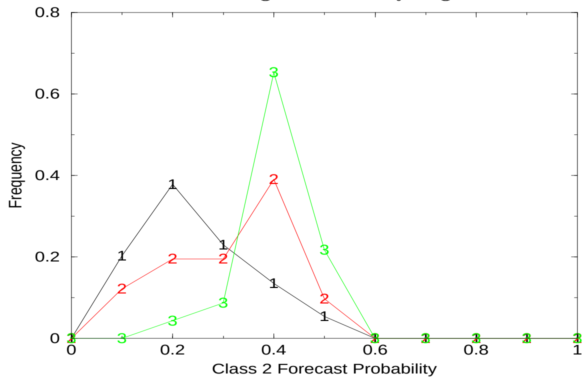


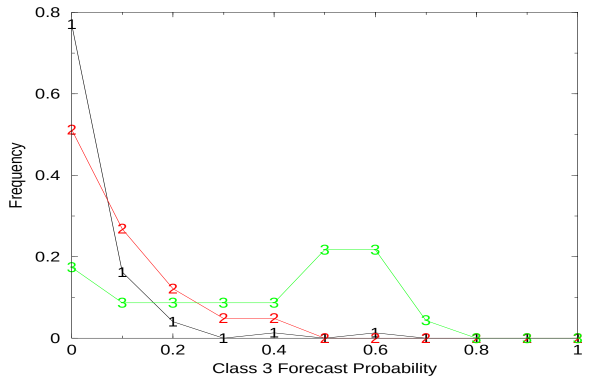




A discrimination plot exhibits the extent to which the forecasts discriminate between the classes. To construct it, plot the class-conditional distributions (or histograms) of the forecasts. The top figure (below) is almost a text-book case of what a good classifier's discrimination plot should look like. When class 1 cases are presented to the NN, it produces class 1 forecasts that peak at higher probabilities. At the same time, if class 3 cases are presented to the NN, the class 1 forecasts are mostly in the lower probabilities. Class 2 cases occupy the middle ground. Skipping to the bottom diagram, the behavior is almost the complement of that seen in the top diagram. Class 3 forecasts are more frequent when class 3 cases are shown, etc. Finally, in the middle diagram, it can be seen that classes 1 and 3 are reasonably discriminated by class 2 forecasts. Same with class 1 and 3. But classes 2 and 3 are not at all discriminated by class 2 forecasts. This, again, points to the problem the NN is having in classifying class 2 cases.







Conclusion

We are done. My goal was to teach you about regression and classification as done with NNs. Although I have covered only the tip of the iceberg, it should give you enough to play with and to get your hands dirty. In the process, we developed a number of NNs for estimating hail size from some measurable attributes. We even ended-up using the same data set for developing both a regression and a classification NN for that task. Together they provide an output like this: Forecast hail size = 24mm, Prob. of coin-size hail = 80%, Prob. of golf ball-size hail = 15%, Prob. of baseball-size hail = 5%.

My other goal was to address the issue of performance (or verification). In my experience, performance does not get nearly the attention it deserves. After all, what's the point of developing a high-powered NN if we don't have a proper way of assessing if it is indeed high-powered? As if the performance assessment of categorical forecasts is not complex enough, probabilistic forecasts make the business even more complex. The source of the complexity is really the multidimensionality of performance itself. But don't loose hope; the situation is relatively under control, and a little reading in the literature should confirm that. Just remember that because of the multidimensionality of performance, it's better not to quantify performance too much, because quantification usually requires distillation, which in turn implies loss of information.