Example I: Predicting the Weather
We decide (or experimentally determine) to use a hidden layer with 42 sigmoidal neurons.
In summary, our network has
- 207 input neurons
- 42 hidden neurons
- 4 output neurons
Because of the small output vectors, 42 hidden units may suffice for this application.

Example I: Predicting the Weather
In summary, our network has to cover all seasons because of dramatic changes in weather across seasons.

Example I: Predicting the Weather
Now we have to train our network.
If we use samples in one-hour intervals for one year, we have 8,760 exemplars.
Our network has $207 \times 42 + 42 \times 4 = 8862$ weights, which means that data from ten years, i.e., $87,600$ exemplars would be desirable (rule of thumb).

Example I: Predicting the Weather
The next thing we need to do is collecting the training exemplars.
First we have to specify what our network is supposed to do:
In production mode, the network is fed with the current weather conditions, and its output will be interpreted as the weather forecast for tomorrow. Therefore, in training mode, we have to present the network with exemplars that associate known past weather conditions at a time $t$ with the conditions at $t - 24$ hrs.
So we have to collect a set of historical exemplars with known correct output for every input.

Example I: Predicting the Weather
The selection of exemplars that we need depends, among other factors, on the amount of changes in weather at our location.
For example, in Honolulu, Hawaii, our exemplars may not have to cover all seasons, because there is little variation in the weather.
In Boston, however, we would need to include data from every calendar month because of dramatic changes in weather across seasons.
As we know, some winters in Boston are much harder than others, so it might be a good idea to collect data for several years.

Example I: Predicting the Weather
And how about the granularity of our exemplar data, i.e., the frequency of measurement?
Using one sample per day would be a natural choice, but it would neglect rapid changes in weather.
If we use hourly instantaneous samples, however, we increase the likelihood of conflicts.
Therefore, we decide to do the following:
We will collect input data every hour, but the corresponding output pattern will be the average of the instantaneous patterns over a 12-hour period.
This way we reduce the possibility of errors while increasing the amount of training data.

Example I: Predicting the Weather
Since with a large number of samples the hold-one-out training method is very time consuming, we decide to use partial-set training instead.
The best way to do this would be to acquire a test set (control set), that is, another set of input-output pairs measured on random days and at random times.
After training the network with the 87,600 exemplars, we could then use the test set to evaluate the performance of our network.
Example I: Predicting the Weather

Neural network troubleshooting:
- Plot the global error as a function of the training epoch. The error should decrease after every epoch. If it oscillates, do the following tests.
- Try reducing the size of the training set. If then the network converges, a conflict may exist in the exemplars.
- If the network still does not converge, continue pruning the training set until it does converge. Then add exemplars back gradually, thereby detecting the ones that cause conflicts.

Example II: Face Recognition

Now let us assume that we want to build a network for a computer vision application.
More specifically, our network is supposed to recognize faces and face poses.
This is an example that has actually been implemented.
All information, program code, data, etc, can be found at:
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html

Example I: Predicting the Weather

- If this still does not work, look for saturated neurons (extreme weights) in the hidden layer. If you find those, add more hidden-layer neurons, possibly an extra 20%.
- If there are no saturated units and the problems still exist, try lowering the learning parameter \( \eta \) and training longer.
- If the network converges but does not accurately learn the desired function, evaluate the coverage of the training set.
- If the coverage is adequate and the network still does not learn the function precisely, you could refine the pattern representation. For example, you could include a season indicator to the input, helping the network to discriminate between similar inputs that produce very different outputs.

Then you can start predicting the weather!

Example II: Face Recognition

The goal is to classify camera images of faces of various people in various poses.
Images of 20 different people were collected, with up to 32 images per person.
The following variables were introduced:
- expression (happy, sad, angry, neutral)
- direction of looking (left, right, straight ahead, up)
- sunglasses (yes or no)
In total, 624 grayscale images were collected, each with a resolution of 30 by 32 pixels and intensity values between 0 and 255.

Example II: Face Recognition

The network presented here only has the task of determining the face pose (left, right, up, straight) shown in an input image.
It uses
- 960 input units (one for each pixel in the image),
- 3 hidden units
- 4 output neurons (one for each pose)
Each output unit receives an additional input, which is always 1.
By varying the weight for this input, the backpropagation algorithm can adjust an offset for the net input signal.

Example II: Face Recognition

The following diagram visualizes all network weights after 1 epoch and after 100 epochs.
Their values are indicated by brightness (ranging from black = -1 to white = 1).
Each 30 by 32 matrix represents the weights of one of the three hidden-layer units.
Each row of four squares represents the weights of one output neuron (three weights for the signals from the hidden units, and one for the constant signal 1).
After training, the network is able to classify 90% of new (non-trained) face images correctly.
Example II: Face Recognition

32 x 32 resolution input images

Network weights before training through each training example

Network weights after 100 iterations through each training example

Example III: Character Recognition

http://sund.de/netze/applets/BPN/bpn2/ochre.html