Before moving to the practical issues regarding neural networks we give an overview of the machine learning process.

**Machine Learning Process**

The **machine learning process** can be visualized in the following way:

**Sampling**

Sampling corresponds to process of obtaining a finite sized set of examples $D$ from the underlying distribution $D$.

**Control of the sampling process:**

1. Data set $D$ is given in its entirety at the beginning of the machine learning process (we do not have control over the sampling process). This is the most common scenario in applications.
2. We have complete or partial control over the sampling process by controlling the size of data set and the manner in which it is sampled. The **active learning** area within machine learning deals with the problem of cost-effective sampling that maximizes the learning accuracy.

**Types of sampling:**

1. Unbiased sampling. It is corresponding to random sampling from $D$.
2. Biased sampling. It corresponds to non-random sampling from $D$ (e.g. when $D$ is obtained by interviewing randomly chosen Temple students, and $D$ is the population of all students in Philadelphia)

**NOTE:** Standard assumption in machine learning is that $D$ is an unbiased sample from the underlying distribution $D$. In real-life, it is very difficult to obtain an unbiased sample, and $D$ is most often biased.

**NOTE:** Unbiased data set $D$ guarantees that there will not be surprises when the learned model is applied on new examples obtained from the underlying distribution $D$. 
Splitting the data into training and test sets
It is critical to allow appropriate estimation of the quality of learned model. Ideally, randomly selected subset of D, called the test data D_{Test}, is set aside at the beginning of the machine learning process. It is used **exclusively** after the learning is finished for the purpose of estimating the accuracy of the trained predictor. The remaining data, called the training data D_{Train}, are used to construct as accurate predictor as possible.

Data preprocessing
This step involves process of transforming the raw data in a way that maximizes the quality of learning. Data preprocessing is a **critical step** for the success of learning.

Roughly, data preprocessing consists of:
1. **Data cleaning**: Correcting for inconsistencies and errors in data (e.g. the recorded body temperature is 9.83F), dealing with missing data (i.e. values of some attributes for some examples are not available)
2. **Attribute construction**: Often, the data are not available in the standard tabular format. For example, we might be given a set of JPEG images each representing a handwritten digit and a set of the associated class labels. Each JPEG image should be represented by a set of attributes believed to be useful for accurate classification. This step is extremely important for successful learning. Successful attributes are often obtained through a painful trial-and-error process.
3. **Attribute transformation**: Involves transformation of raw attributes into a form more suitable for a specific learning algorithm.
   a. Transforming categorical into numerical attributes: The step necessary if neural networks are used.
   b. Binning: Discretizing a continuous attribute into a finite set of discrete values. Necessary for certain machine learning algorithms.
   c. Rescaling: Linear or nonlinear transformation of the original continuous attribute. For example, the patient’s temperature may be recorded in °C when we require it in °F for building the predictor).
   d. Dimensionality reduction: Reducing the number of attributes by projecting the original attribute space to a lower dimensional attribute space while preserving the information needed for successful learning.

Learning
The process of using the preprocessed training data to train a predictor. Involves selection of the accuracy measure, selection of the machine learning algorithm, selection of the algorithm parameters, and application of the algorithm to construct a predictor.

Estimating the prediction accuracy
NOTE: Before the predictor is applied on test data to estimate its accuracy, the test data should be subject to **exactly the same** set of transformations used to preprocess the training data.
Practical Issues related to Neural Networks

**Overfitting**
Corresponds to the ability of **powerful learning algorithms** to over-learn the training data. The consequence of over-learning is often reduced accuracy on unseen data.

In the above figure, data generating process is $y = f(x) + \varepsilon$, where $f(x)$ is a linear function of $x$ and $\varepsilon$ is random noise. Assuming a very powerful learning algorithm able to learn continuous function of an arbitrary complexity, the dotted blue line could represent the learned function $g(x)$ based on the training data (filled circles). However, it is obvious that $g(x)$ is far from the optimal one defined by $f(x)$. Therefore, the accuracy of $g(x)$ is significantly lower than that of $f(x)$. It seems that using a less powerful algorithm would have led to better predictor!!

1) **Overfitting of neural network caused by large number of epochs**
If we plot a graph of $\text{MSE}_{\text{Train}}$ and $\text{MSE}_{\text{Test}}$ against the number of epochs (number of weight update iterations) of a neural network, it will look like the following:
**Explanation:** As the number of epochs is increasing, weights in neural networks are becoming more specialized to the properties of the training data set. This will result in a neural network highly sensitive to small changes in attributes (i.e. function represented by the neural network will be highly nonlinear), and therefore to overfitting.

**How to prevent overfitting:** From the figure it is evident that training should be stopped when MSE_{Test} starts to increase. However, we are not allowed to use test data during the learning process.

**Validation (or, early stopping) data:** \( D_{Training} \) is randomly divided into two subsets. One set is used to update the weights and another is used as a validation set to determine when to stop training.

![Validation Data Diagram](image)

**Stopping criterion:** We could stop after the first iteration when MSE_{Validation} increases. However, it is possible that if training is continued MSE_{Validation} continues to decrease. Therefore, stopping criterion should be more forgiving. For example, a good choice can be: if MSE_{Validation} in 5 successive epochs was not smaller than the previously smaller MSE_{Validation} training is stopped. The number of epochs Epochs_{Best} resulting in the smallest MSE_{Validation} should be recorded.

**Retraining:** Since the available data D is often limited (and, remember, that one portion of it was already reserved as \( D_{Test} \)), using validation set could result in too small training data set. As a consequence, the quality of neural network will be lower than if all the examples from D were used in training. Therefore, once the optimal number of epochs Epochs_{Best} is determined, we should add the \( D_{Validation} \) to training data and repeat the training with exactly Epochs_{Best} epochs.

**2) Overfitting caused by neural network complexity:**
Neural network complexity is defined by the number of hidden layers and the number of nodes in each of the hidden layers. The following is often the case:

![Overfitting Diagram](image)
Therefore, as network complexity is increasing, its ability to learn training data increases. However, if complexity is too high, there is a danger of overfitting.

**Controlling the network complexity:** To determine the appropriate complexity for a given data set, it is advisable to train neural networks with different complexities. Usually, one should start from the simplest network (i.e. using one hidden neuron) and gradually increase number of hidden nodes until accuracy could not be improved (or it even starts deteriorating).

Given two neural networks with different complexity and same accuracy, the simpler one is preferred (this is also called the Occam’s rule). Barring other explanation, training time is proportional to the complexity of neural network.

The process of selecting the complexity of prediction model (e.g. neural network) is called **Model Selection**.

**3) The Learning Curve**

Learning curve corresponds to the **relationship between** size of training data set and the accuracy of a predictor trained on it.

The region depicted as a ‘Good choice of data set’ is the best region because (1) if a smaller training data size is used then the MSE will be significantly higher and (2) if a larger training data size is used then computational effort would increase without noticeable increase in prediction accuracy.

**Theoretical Results:**

For Linear Regression, the learning curve satisfies the following equation

\[
MSE = MSE^* + \varepsilon, \quad E(\varepsilon) = O\left(\frac{1}{N}\right), Vari(\varepsilon) = O\left(\frac{1}{N}\right),
\]

where \(MSE^*\) is best obtainable accuracy if infinitely large training data is available, and \(\varepsilon\) is a random number whose mean and variance decrease with training data size as \(1/N\). This means that for small \(N\), MSE will be quite larger than \(MSE^*\) and it value will vary
largely depending on the specific training set with N examples. As N grows, \(MSE\) will be always very close to \(MSE^*\).

For Neural Networks, learning curve would satisfy

\[
MSE = MSE^* + \Delta MSE + \epsilon, \quad E(\epsilon) = O\left(\frac{1}{N}\right), Var(\epsilon) = O\left(\frac{1}{N}\right)
\]

where \(\Delta MSE\) is a positive random number due to the existence of local minima. So, if neural network training converges to a local minimum, \(\Delta MSE\) is a difference in accuracy as compared to the case when training would converge to the global minimum.

**NOTE:** Existence of \(\Delta MSE\) does not mean that linear regression is better than neural networks. \(MSE^*\) of neural networks can be much smaller than that of linear regression, and it could easily offset the effects of \(\Delta MSE\).

**4) Influence of test data size to accuracy estimation**

Given a trained neural network represented as \(f(x; \theta)\) the goal is to estimate its accuracy defined as \(MSE = E_{x}(y - f(x; \theta))^2\) (i.e. expected squared prediction error).

In practice, the available test data size is limited, and MSE can be estimated as

\[
ASE = \frac{1}{N} \sum_{i=1}^{N} \Sigma (y_i - F(\tilde{x}_i, \tilde{\theta}))^2
\]

where \(N\) is the size of the test data set and ASE is average squared error.

Let \(se_i = (y_i - F(\tilde{x}_i, \tilde{\theta}))^2\). Then \(ASE = \frac{1}{N} \sum_i se_i\)

Therefore, \(E(ASE) = E(se_i) = E_{x}(y - f(x; \theta))^2 = MSE\). So, in expectation, ASE equals MSE. However, there will always be some randomness in ASE as compared to true MSE. This randomness can be measured by its variance:

\[
Var(ASE) = \sum_{i=1}^{N} Var\left(\frac{se_i}{N}\right) = \sum_{i=1}^{N} \frac{Var(se_i)}{N^2} = \frac{Var(se_i)}{N}
\]
Therefore, variance of ASE decreases as 1/N with the size of test data. In other words, the larger the test data set, the more accurate is the estimate of neural network accuracy.

5) Checking accuracy of Neural Networks
The important question is: given a data set D of limited size, how do we use it to maximize learning accuracy while allowing proper testing of its accuracy? These two goals are conflicting – we would like to use as much data as possible both for training and for testing.

In practice, three scenarios can be considered with respect to the size of the available data set D:

**Case 1:** Very large (almost unlimited) data set is available.
Using all the data in training would computationally too expensive (causing memory overload or unacceptably long training times), and could be deep within the convergence region of the learning curve. Similarly, testing data could also be unacceptably large. Therefore, we could easily take e.g., 1% of this data as D_{Training} and another 1% as D_{Testing} while obtaining both accurate neural network and properly checking its accuracy.

**Case 2:** Moderately large data set is available.
Moderately large data set means that by splitting the data set randomly into equally sized training and test sets could lead to accurate neural network training and to the proper estimate of its accuracy.

**Case 3:** Small data is available.
The available data set D is not of sufficient size to provide accurate learning (i.e. its size is not within a convergence region of the learning curve). Therefore, it is very important to be very careful in using the available data for accurate learning and accuracy estimate.

Cross-validation is appropriate procedure in such a scenario. Its idea is to use all the data both in training and testing! How is it possible?

Let us consider 3-cross validation procedure:

Divide the data randomly into 3 equally sized subsets D1, D2, and D3.
Step 1: D1 is test data; D2 & D3 are training data; train NN1 on D2+D3 and test it on D1.
Step 2: D2 is test data; D3 & D1 are training data; train NN2 on D1+D3 and test it on D2.
Step 3: D3 is test data; D1 & D2 are training data; train NN3 on D1+D2 and test it on D3.
Step 4: Reported accuracy is an average testing accuracy from the 3 cross-validation rounds.
Step 5: Train a final neural network on whole data set D. The accuracy obtained in step 4 will be a pessimistic estimate of the trained neural network.

Case 4: Very small data set available
If data set D is very small, we can apply the extreme case of cross validation – N-cross validations (also known as leave-out-one validation).
NOTE: Leave-one-out validation provides the best possible use of available data for accurate estimation of prediction accuracy. However, it is also most expensive since it requires training of N neural networks.

Other practical issues related to neural networks

1) Attribute Scaling
Neural network training could be quite sensitive to the range of the attributes. This is specially the case when different attributes ranges that differ by orders of magnitude (e.g. one attribute measures size of small objects \(x_1=0.0003\), while other measures their concentration \(x_2=245,555\)).
Solution: To train accurate neural networks it is often needed to rescale original attributes in the similar range. It could be done by transforming attribute \(X\) to \(X'\) by using the Z-score normalization with
\[
X' = \frac{X - \mu_X}{\sigma_X}
\]
where \(\mu_X\) and \(\sigma_X\) are the mean and the standard deviation of attribute \(X\).

2) Removing Attribute Correlations
Neural network training is very sensitive to the presence of correlated attributes (two attributes are correlated if their correlation coefficient is close to 1 or \(-1\)).
Solution: Original attributes are transformed into new attribute space where correlation coefficient between each pair of attributes is zero. This procedure is called Principal Component Analysis (PCA).
From the above figure, correlation coefficient between X1 and X2 is near 1 (may be > 0.8). Therefore, the use of PCA will result in two new orthogonal attributes X1’ and X2’ whose correlation coefficient is zero.

3) Choice of initial weights
Proper choice of initial weights could be very important for speedy and successful training of neural network.

**Brief analysis:** If initial weights are large, the input to each neuron is likely to be very large, and its output will be close to 1 or 0 (if sigmoid neuron is used). The derivative of sigmoid function will therefore be very small (because the slope of sigmoid will be close to zero) and the weight update will be small. Therefore, large number of epochs will be needed to escape the initial bad choice of weights.

**Solution:** Initial weights are chosen to be relatively small random numbers (e.g. within a range between -0.1 and 0.1)

**NOTE:** For such small initial weights, the input to each neuron is relatively small, so its output is a near-linear function of its input. As a consequence, the output of an initial neural network is approximately linear combination of its inputs. This is very desirable starting point – only if necessary will some weights become large which will result in neural network being a nonlinear function of its inputs.

4) Ensemble of Neural Networks and Bagging
Starting from different initial weights, each trained neural network will be different even if identical training data set is used.

Common sense indicates that averaging the large number of neural networks trained using different initial weights will result in better prediction than when using only a single neural network. This assumption is most often confirmed in applications, and constructing an **ensemble of neural networks** is therefore highly recommended.

**Bagging:** The accuracy of an ensemble could be further increased by introducing the randomness in the choice of the training data set. Bagging is a procedure that almost always results in better accuracy than a single neural network:

- **Step 1:** Start from a given training data set \( D_{Training} \) of size \( N \).
- **Step 2:** Select \( N \) examples randomly with repetition from \( D_{Training} \) to construct \( D_{Training}^* \).
- **Step 3:** Train a neural network using \( D_{Training}^* \).
- **Step 4:** Repeat Steps 2 and 3 \( M \) times
- **Step 5:** Final predictor is an average prediction of an ensemble of the \( M \) neural networks.

**NOTE:** There are theoretical results showing that bagging is successful procedure for maximizing the accuracy of powerful machine learning algorithms.