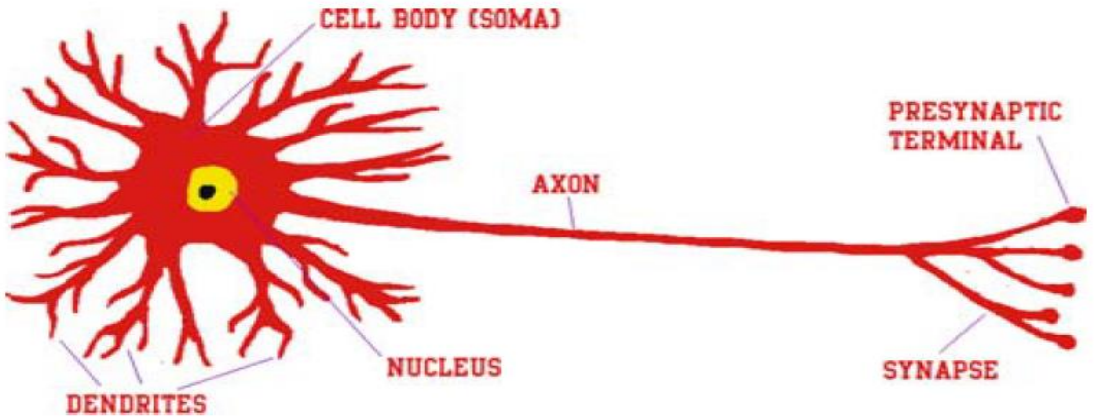
Neural Networks

Stephen Jun Villejo

School of Statistics

Neural Networks

- The central idea of neural networks is to extract linear combinations of the predictors as derived features, and then model the response as a nonlinear function of these features.
- The term "neural networks" derives from the fact that they were first developed as models for the human brain. Each unit represents a neuron, and the connections represent synapses.

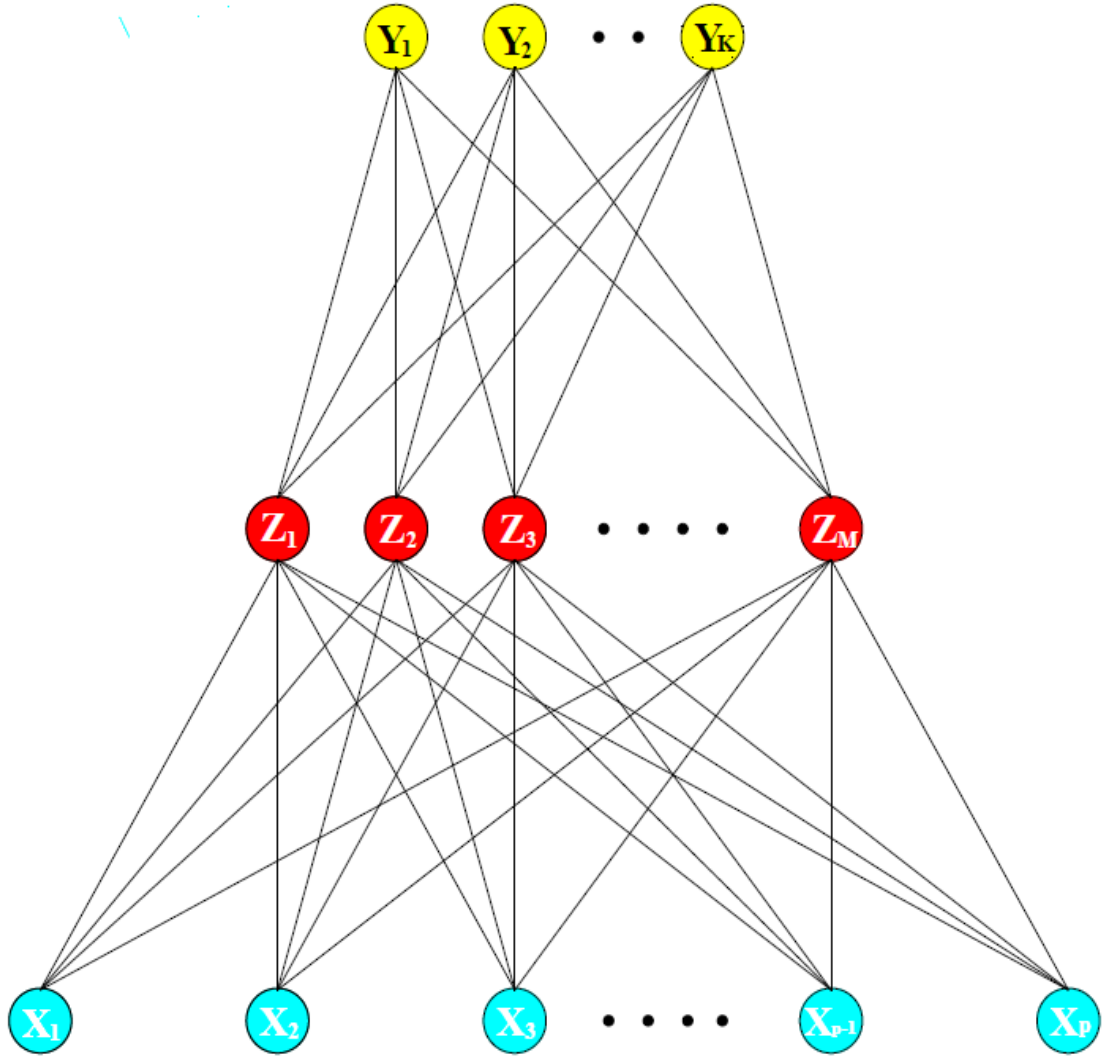


Neural Networks

- A **multilayer feedforward neural network (perceptron)** is a multivariate technique that nonlinearly maps a set of inputs to a set of outputs.
- A multilayer perceptron has the following architecture: set of inputs, one or more layers of hidden nodes, and output nodes

Neural Networks

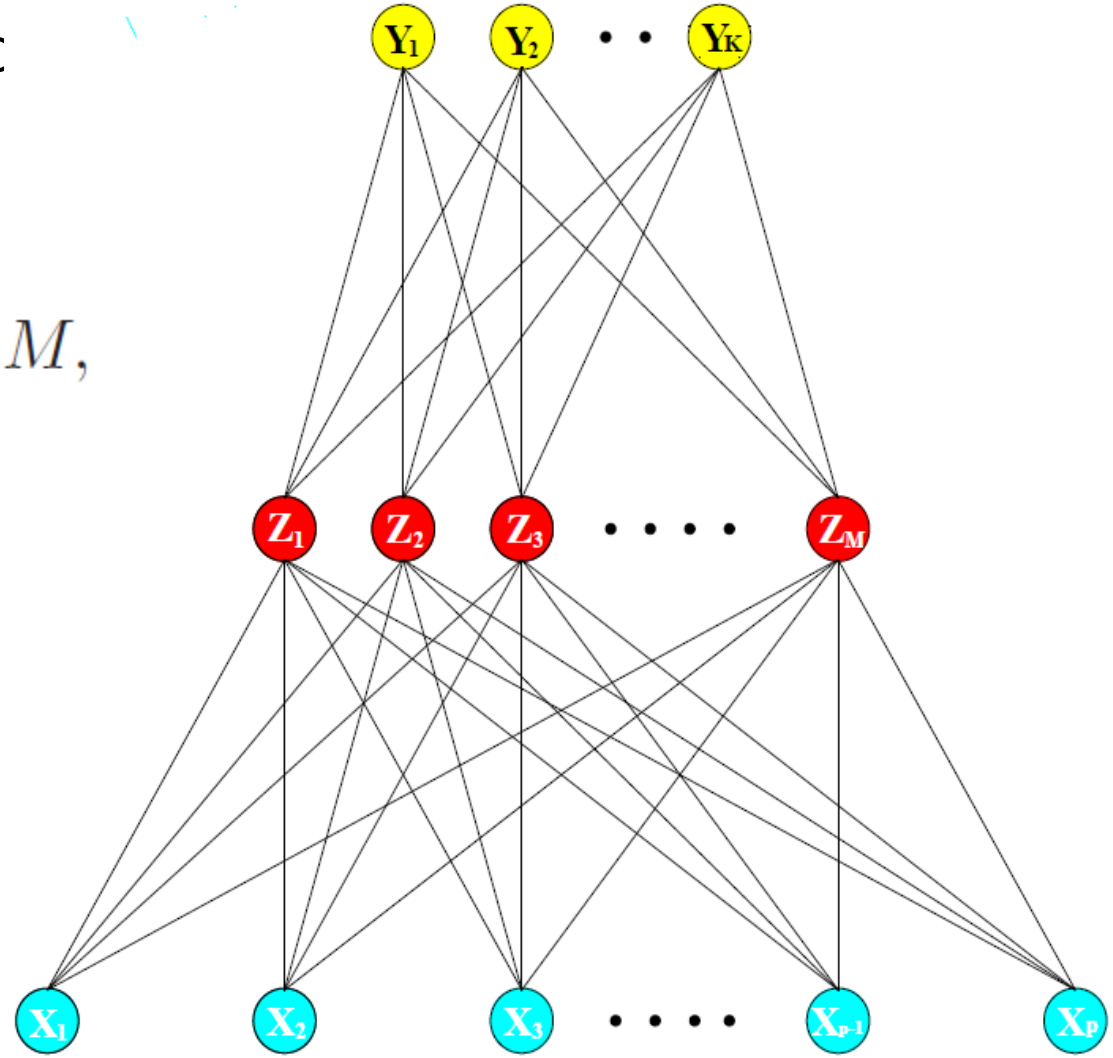
- Figure on the right is a schematic of a **single hidden layer neural network**, one of the most widely used neural net.
- $X_1, \ldots, X_p$ are the predictors.
- $Z_1, Z_2, \ldots, Z_M$ are derived features
- The target $Y_1, \ldots, Y_K$ are modeled as a function of linear combinations of the Z's.



Neural Networks

- Figure on the right is a schematic of a single hidden layer neural network.

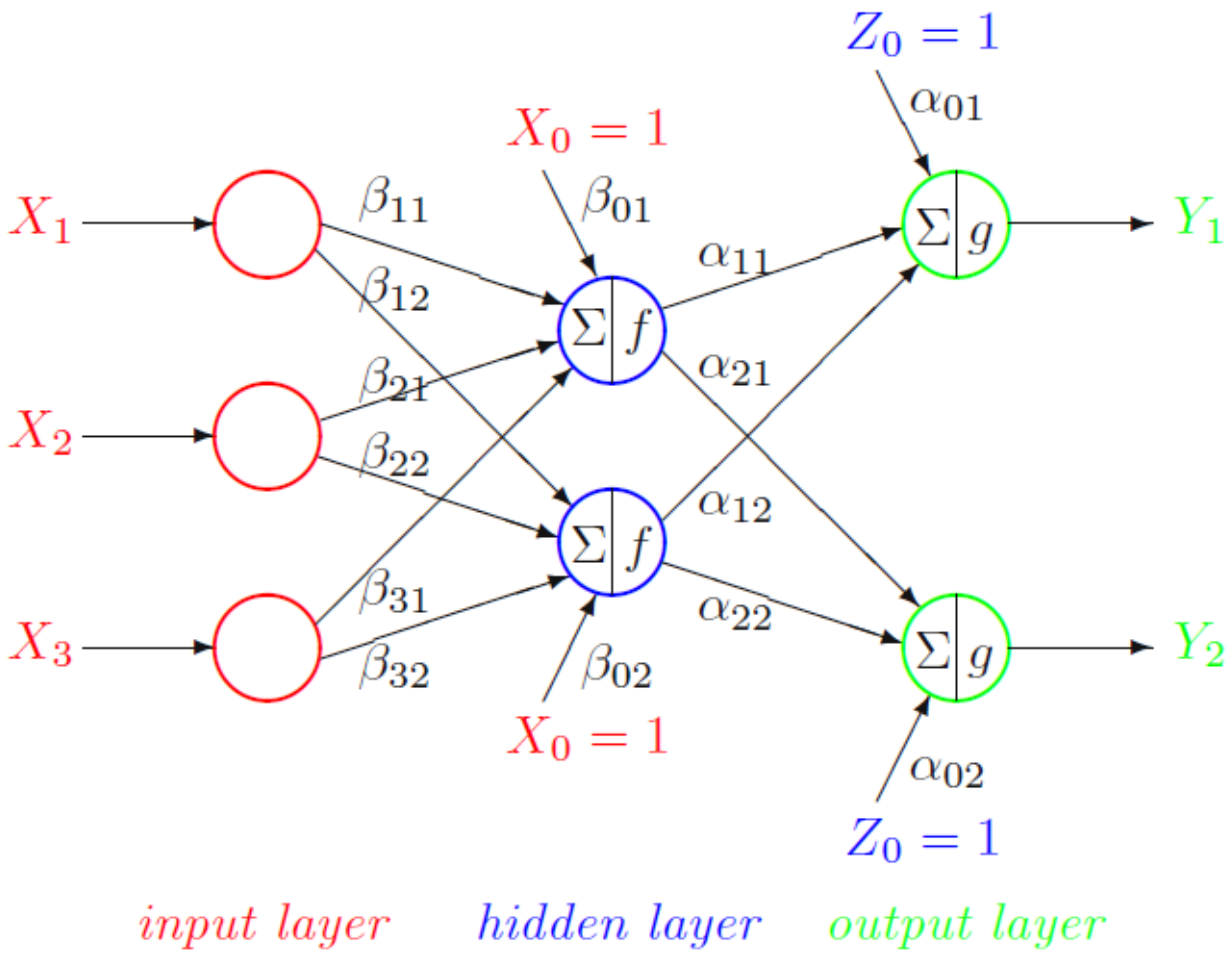
$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1, \ldots, M,$$
$$T_k = \beta_{0k} + \beta_k^T Z,\ k = 1, \ldots, K,$$
$$f_k(X) = g_k(T),\ k = 1, \ldots, K,$$



Neural Networks

Illustration

Figure on the right is a single hidden layer neural network with 3 input nods, 2 nodes in the hidden layer, and 2 output nodes. The α's and β's are the weights and f and g are activation functions.



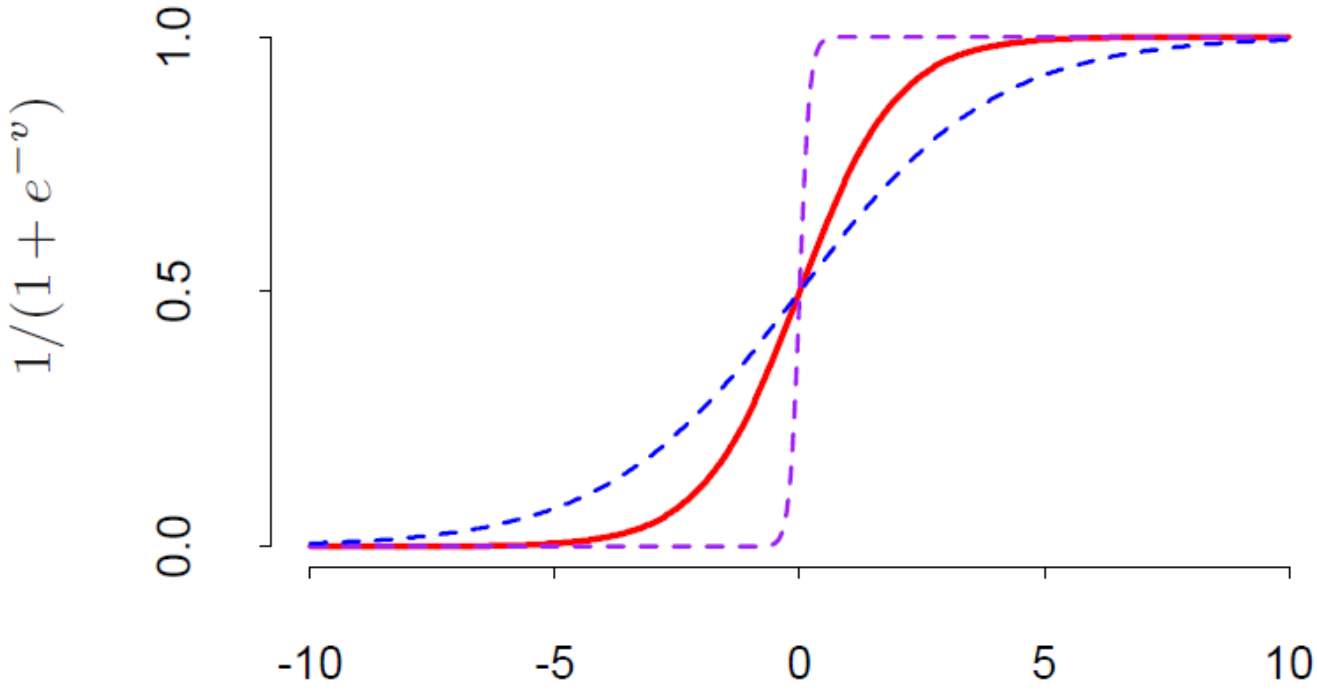
Neural Networks

- The function $\sigma(\,)$ is called the activation function. Examples of activation functions are the following:

Activation Function	$f(u)$	Range of Values
Identity, linear	u	$\Re$
Hard-limiter	$\text{sign}(u)$	$\{-1, +1\}$
Heaviside, step, threshold	$I_{[u \geq 0]}$	$\{0, 1\}$
Gaussian radial basis function	$(2\pi)^{-1/2} e^{-u^2/2}$	$\Re$
Cumulative Gaussian (sigmoid)	$\sqrt{2/\pi} \int_0^u e^{-z^2/2} dz$	$(0, 1)$
Logistic (sigmoid)	$(1 + e^{-u})^{-1}$	$(0, 1)$
Hyperbolic tangent (sigmoid)	$(e^u - e^{-u})/(e^u + e^{-u})$	$(-1, +1)$

Neural Networks

- The most interesting of these activating functions $\sigma(\,)$ are the sigmoidal ("S-shaped") functions, such as the logistic and hyperbolic tangent.
- Below is a plot of the sigmoidal function.



Neural Networks

- If $\sigma(\,)$ is the identity function, then the entire model collapses to a linear model in the inputs.
- Hence, a neural network can be thought of as a nonlinear generalization of the linear model, both for regression and classification.

Neural Networks

Fitting Neural Networks

The unknown parameters in the neural network model are called **weights**. The complete set of weights are

$$\{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\}\ \ M(p+1) \text{ weights},$$
$$\{\beta_{0k}, \beta_k;\ k = 1, 2, \ldots, K\}\ \ K(M+1) \text{ weights}.$$

- Measure of fit for regression is $R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2.$
- Measure of fit for classification is $R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i)$

Neural Networks

Fitting Neural Networks

- The generic approach to minimizing $R(\theta)$ is through a method called **backpropagation**.
- The backpropagation algorithm efficiently computes the first derivatives of an error function with respect to the network weights. These derivatives are then used to estimate the weights by minimizing the error function through an iterative gradient-descent method.

Neural Networks

Issues in Training Neural Networks

- The model is generally overparameterized, and the optimization problem is unstable. Too many weights will lead to overfitting.
- The backpropagation algorithm is a slow learner, the estimates may be unstable, there may exist many local minima, and convergence is not assured in practice.

Neural Networks

Guidelines in Training Neural Networks

- Starting values of weights are usually chosen to be near zero. Hence, the model starts out nearly linear, and becomes nonlinear as the weights increases.
- Because of the slow progression of the algorithm, overfitting has been accidentally avoided by stopping the algorithm prior to convergence (*early stopping*).
- Applying *weight decay*, which is shrinking the weights toward zero also helps avoid overfitting.
- It is best to standardize all inputs to have mean zero and standard deviation one, so that all inputs are treated equally.

Neural Networks

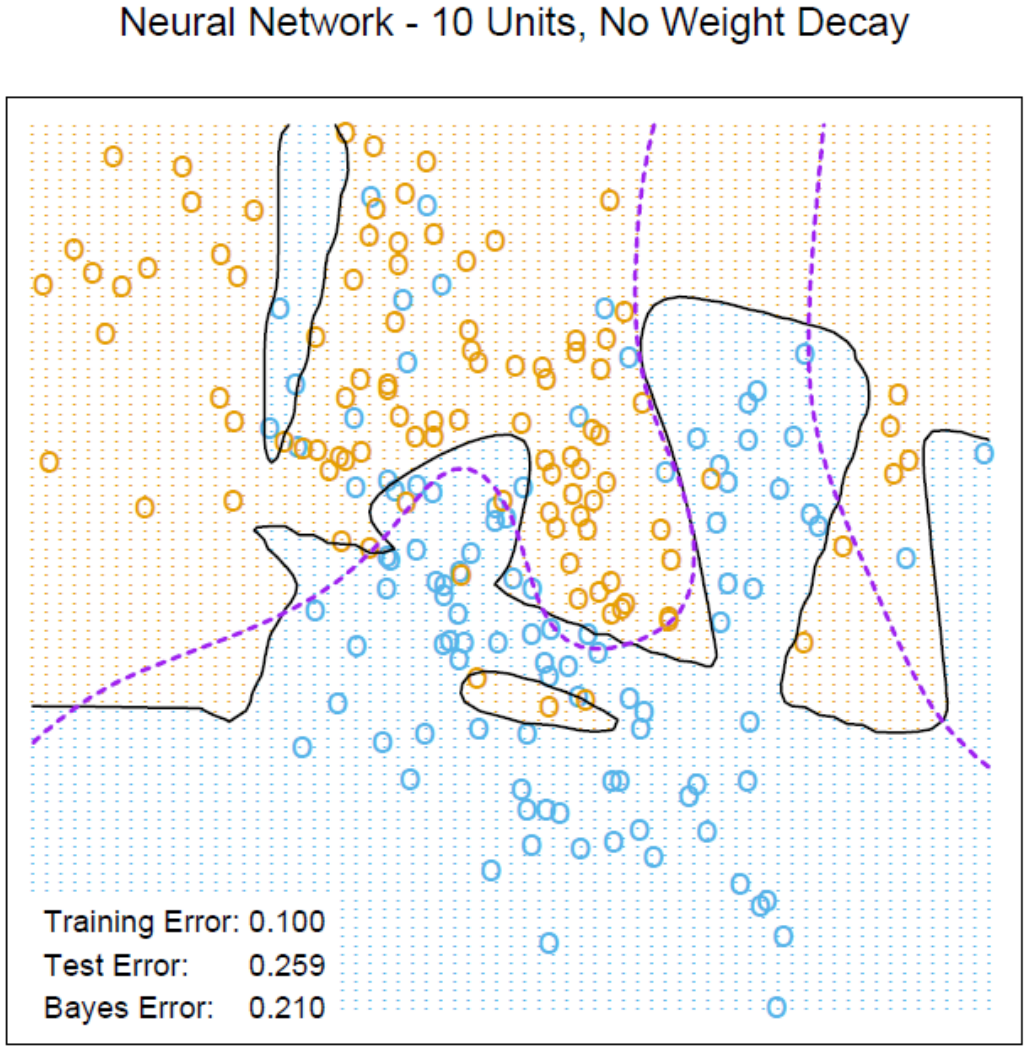
Guidelines in Training Neural Networks

- Generally, it is better to have too many hidden units than too few. With too few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data; with too many hidden units, the extra weights can be shrunk toward zero.
- Typically, the number of hidden units is somewhere in the range of 5 to 100. It is common to start with a large number of units and then apply regularization.
- Start with a number of random starting configurations.

Neural Networks

Illustration

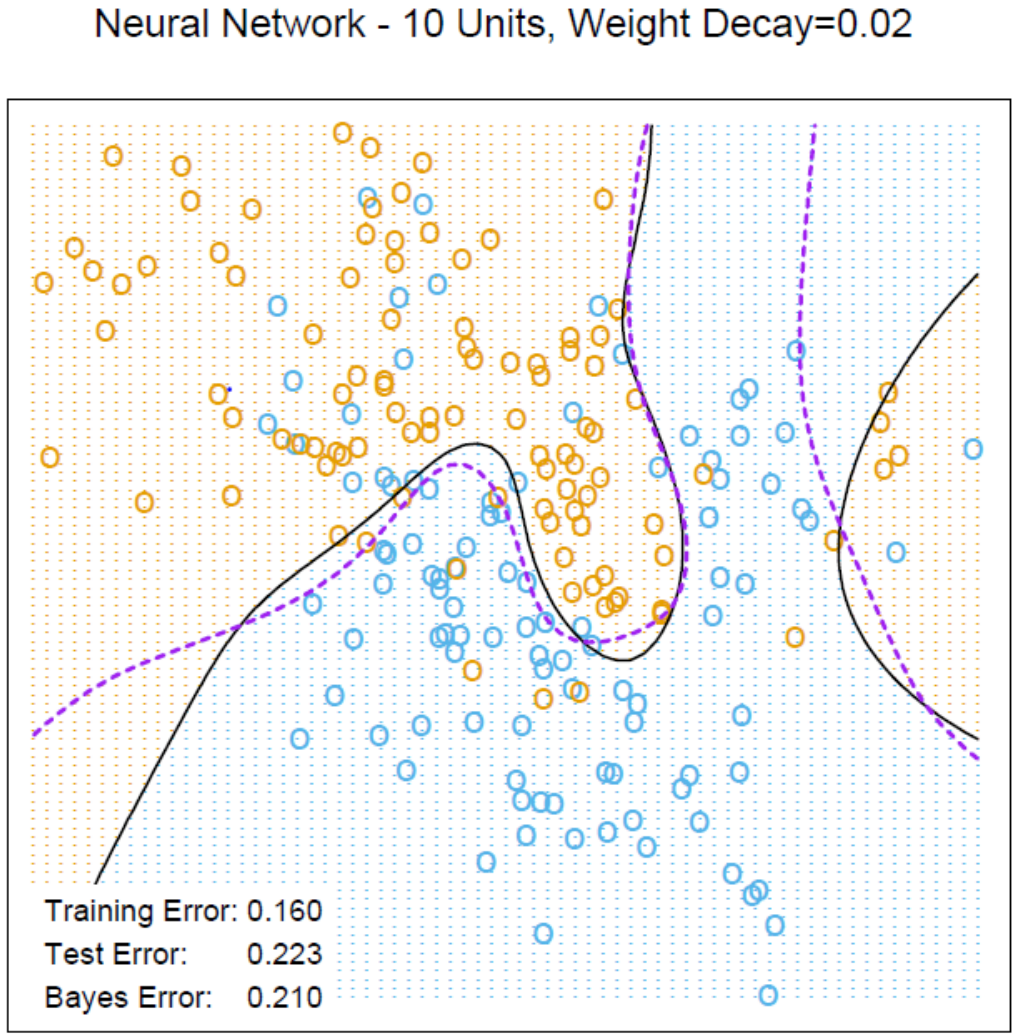
- Figure on the right is a result of training a neural network with 10 hidden units, without weight decay.
- The neural network overfits the data.



Neural Networks

Illustration

- Figure on the right is a result of training a neural network with 10 hidden units with weight decay.
- Weight decay has clearly improved prediction.



Neural Networks

Illustration

Below we see that weight decay dampened the weights in both layers: the resulting weights are spread fairly evenly over the then hidden units.