Neural Networks

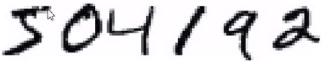
Steve Cygu

January 24, 2019

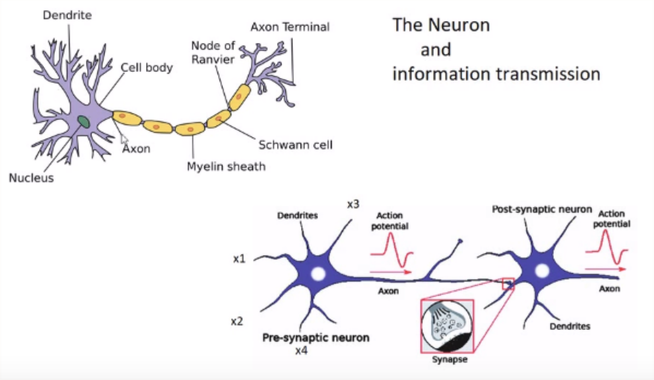
OUTLINE

Why Machine Learning?

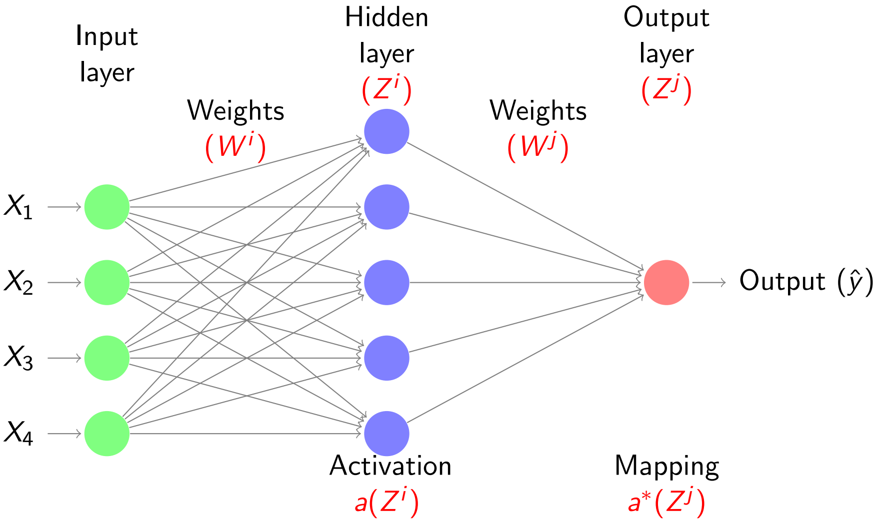
Can we write algorithm to correctly identify each of the objects?



Neural Networks and Human Brain

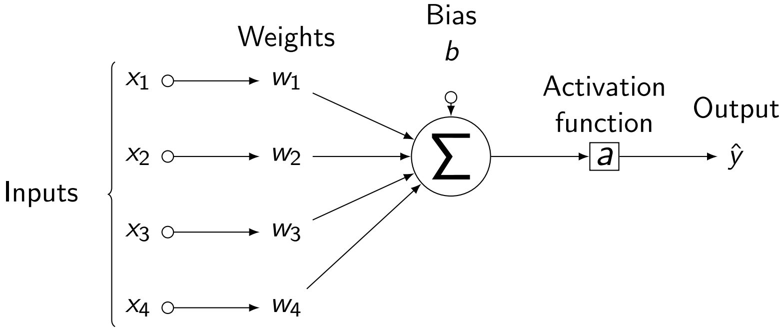


Multi-layer perceptron (MLP)



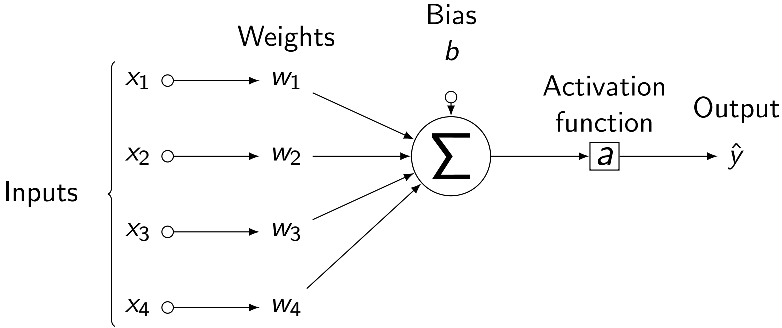
Single-layer Perceptron

- No hidden layer, a single neuron.



Single-layer Perceptron

- No hidden layer, a single neuron.



$$\hat{y} = b + \sum_{i=1}^{n} x_i w_i$$

Components

Layers

- **Input nodes**
  No computation
- **Hidden nodes (Neurons)**
  Intermediate processing, computation and transfers to another hidden layer or output.
- **Output nodes**
  Uses a function (not necessarily activation function) to map the input from other layers to desired output format.
  - Sigmoid
  - Softmax

Synapse/Connections

- Transfers the output of neuron i to the input of neuron j.
- Each connection is assigned weight, W_{ij}

Activation function

Introduces nonlinearity into the neuron output.

- Sigmoid (Logistic Activation Function)

$$a(z) = \frac{1}{1 + exp(-z)}$$

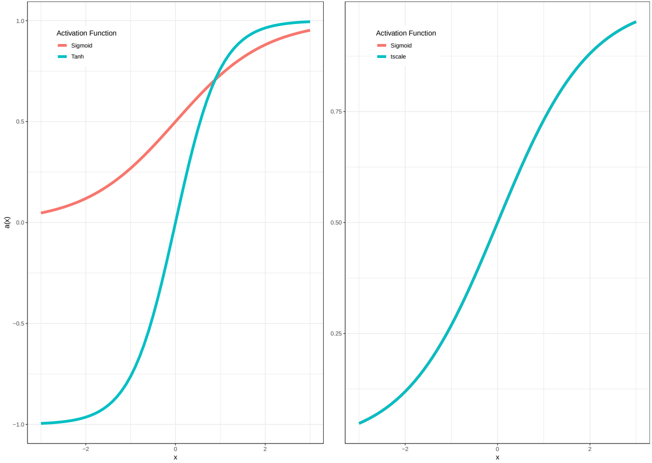
- Tanh (hyperbolic tangent Activation Function)

$$a(z) = tanh(z) = \frac{2}{1 + exp(-2z)} - 1 = 2sigmoid(2z) - 1$$

- ReLU (Rectified Linear Unit Activation Function)

$$a(z) = max(0, z)$$

Sigmoid and Tanh Activation Functions



Fitting Neural Networks

Weights are the parameters. The generic approach is by **gradient descent**.

- Forward-propagation (feed-forward)
- Backward-propagation

Feed-forward

- Randomly assign starting weights and consider any Z_k^i , $k = 1 \cdots n$. Then,

Feed-forward

- Randomly assign starting weights and consider any Z_k^i , $k = 1 \cdots n$. Then,

  $$Z_1^i = X_1 W_{1,1}^i + X_2 W_{2,1}^i + X_3 W_{3,1}^i + X_4 W_{4,1}^i$$

Feed-forward

- Randomly assign starting weights and consider any Z_k^i , $k = 1 \cdots n$. Then,

$$Z_1^i = X_1 W_{1,1}^i + X_2 W_{2,1}^i + X_3 W_{3,1}^i + X_4 W_{4,1}^i$$
$$Z_2^i = X_1 W_{1,2}^i + X_2 W_{2,2}^i + X_3 W_{3,2}^i + X_4 W_{4,2}^i$$

Feed-forward

- Randomly assign starting weights and consider any Z_k^i , $k = 1 \cdots n$. Then,

$$
\begin{aligned}
Z_1^i &= X_1 W_{1,1}^i + X_2 W_{2,1}^i + X_3 W_{3,1}^i + X_4 W_{4,1}^i \\
Z_2^i &= X_1 W_{1,2}^i + X_2 W_{2,2}^i + X_3 W_{3,2}^i + X_4 W_{4,2}^i \\
&\vdots
\end{aligned}
$$

Feed-forward

- Randomly assign starting weights and consider any Z^i_k , $k = 1 \cdots n$. Then,

$$
\begin{aligned}
Z^i_1 &= X_1 W^i_{1,1} + X_2 W^i_{2,1} + X_3 W^i_{3,1} + X_4 W^i_{4,1} \\
Z^i_2 &= X_1 W^i_{1,2} + X_2 W^i_{2,2} + X_3 W^i_{3,2} + X_4 W^i_{4,2} \\
&\vdots
\end{aligned}
$$

Z^i ‘component’ is the sum of weighted inputs to each neuron.

$$Z^i = XW^i \tag{1}$$

Feed-forward

Apply activation function to 1

$$a^i = a(Z^i) \tag{2}$$

Propagate 2 to the output layer

$$Z^j = a^i W^j \tag{3}$$

$$\implies \hat{y} = a^j = a^*(Z^j) \tag{4}$$

Back propagation

- Aim is to estimate weights that ensures the model fits the training data well.
- Calculate the error at the output nodes and propagate them back to the network.

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{5}$$

Back propagation

- Aim is to estimate weights that ensures the model fits the training data well.
- Calculate the error at the output nodes and propagate them back to the network.

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{5}$$

$$J(W) = \frac{1}{2} \sum \left(y - a^*(a(XW^i)W^j)\right)^2 \tag{6}$$

Back propagation

- Aim is to estimate weights that ensures the model fits the training data well.
- Calculate the error at the output nodes and propagate them back to the network.

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{5}$$

$$J(W) = \frac{1}{2}\sum \left(y - a^*(a(XW^i)W^j)\right)^2 \tag{6}$$

- Compute the gradient; $\frac{\partial J}{\partial W^i}$ and $\frac{\partial J}{\partial W^j}$

Back propagation

- Aim is to estimate weights that ensures the model fits the training data well.
- Calculate the error at the output nodes and propagate them back to the network.

$$J = \sum \frac{1}{2}(y - \hat{y})^2 \tag{5}$$

$$J(W) = \frac{1}{2}\sum \left(y - a^*(a(XW^i)W^j)\right)^2 \tag{6}$$

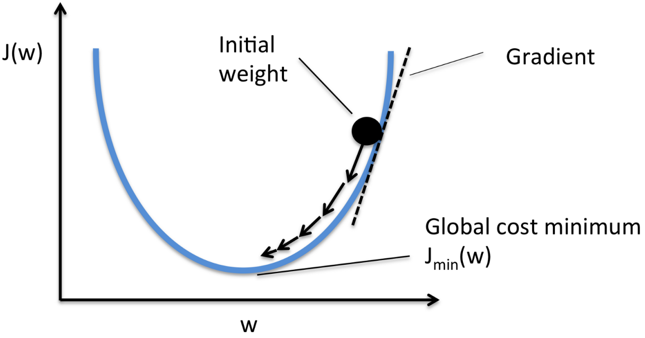
- Compute the gradient; $\frac{\partial J}{\partial W^i}$ and $\frac{\partial J}{\partial W^j}$
- Adjust the weights using optimization method such as **Gradient Descent**.

Gradient descent

$$W_{t+1} = W_t - \gamma \Delta J(W_t)$$

Gradient descent

$$W_{t+1} = W_t - \gamma \Delta J(W_t)$$



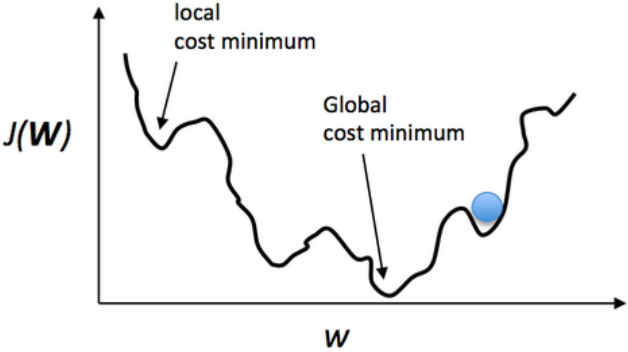
Some Issues in Training Neural Networks

1. Starting values
   - Starting weights are random numbers near zero.
   - However, near zero weights collapses NN into approximately linear model.
   - Exactly zero weights leads to zero derivatives and perfect symmetry.
   - Large weights lead to poor results.

Some Issues in Training Neural Networks

1. Starting values
   - Starting weights are random numbers near zero.
   - However, near zero weights collapses NN into approximately linear model.
   - Exactly zero weights leads to zero derivatives and perfect symmetry.
   - Large weights lead to poor results.
2. Overfitting and Stopping Criterion
   - Reduce training error to some predetermined threshold $\longrightarrow$ overfitting.
   - Regularization by *weight decay* (analogous to ridge regression), $\lambda \geq 0$. Larger values of λ shrinks weights toward zero.
   - Cross-validation is used to estimate λ.

3. Convergence at the Local Minima



Example

- Breast Cancer Wisconsin Data Set

Example

- Breast Cancer Wisconsin Data Set
- The data set contains 569 cases with 31 variables. The diagnosis classification is either (M = Malignant) or (B = Benign).

Example

- Breast Cancer Wisconsin Data Set
- The data set contains 569 cases with 31 variables. The diagnosis classification is either (M = Malignant) or (B = Benign).

Show 32 entries　　Search:

	vars	labels
1	id_number	ID number
2	diagnosis	Diagnosis (M = malignant, B = benign)
3	radius_mean	radius (mean of distances from center to points on the perimeter) - Mean
4	texture_mean	texture (standard deviation of gray-scale values) - Mean
5	perimeter_mean	perimeter - Mean
6	area_mean	area - Mean
7	smoothness_mean	smoothness (local variation in radius lengths) - Mean
8	compactness_mean	compactness (perimeter^2 / area - 1.0) - Mean
9	concavity_mean	concavity (severity of concave portions of the contour) - Mean
10	concave_points_mean	concave points (number of concave portions of the contour) - Mean
11	symmetry_mean	symmetry - Mean
12	fractal_dimension_mean	fractal dimension ("coastline approximation" - 1) - Mean
13	radius_se	radius (mean of distances from center to points on the perimeter) - Standard error
14	texture_se	texture (standard deviation of gray-scale values) - Standard error
15	perimeter_se	perimeter - Standard error
16	area_se	area - Standard error
17	smoothness_se	smoothness (local variation in radius lengths) - Standard error
18	compactness_se	compactness (perimeter^2 / area - 1.0) - Standard error
19	concavity_se	concavity (severity of concave portions of the contour) - Standard error
20	concave_points_se	concave points (number of concave portions of the contour) - Standard error
21	symmetry_se	symmetry - Standard error
22	fractal_dimension_se	fractal dimension ("coastline approximation" - 1) - Standard error
23	radius_worst	radius (mean of distances from center to points on the perimeter) - Worst
24	texture_worst	texture (standard deviation of gray-scale values) - Worst
25	perimeter_worst	perimeter - Worst
26	area_worst	area - Worst
27	smoothness_worst	smoothness (local variation in radius lengths) - Worst
28	compactness_worst	compactness (perimeter^2 / area - 1.0) - Worst
29	concavity_worst	concavity (severity of concave portions of the contour) - Worst
30	concave_points_worst	concave points (number of concave portions of the contour) - Worst

Methods

- Data partitioning: 80% training set and 20% test set
- Automatic grid search, with a $tunelength = 10$ was used to find optimal parameter values

Methods

- Data partitioning: 80% training set and 20% test set
- Automatic grid search, with a $tunelength = 10$ was used to find optimal parameter values
  - Weight decay

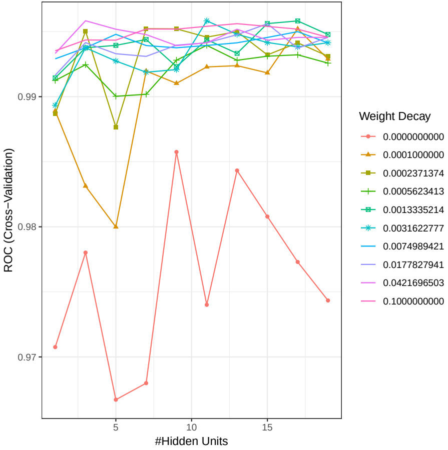
Methods

- Data partitioning: 80% training set and 20% test set
- Automatic grid search, with a $tunelength = 10$ was used to find optimal parameter values
  - Weight decay
  - Number of hidden neurons

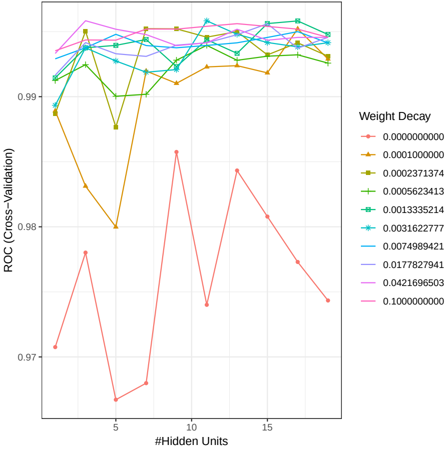
Methods

- Data partitioning: 80% training set and 20% test set
- Automatic grid search, with a $tunelength = 10$ was used to find optimal parameter values
  - Weight decay
  - Number of hidden neurons
- 10-fold cross validation
- ROC was used as the performance metric

results : Weight decay and hidden neurons

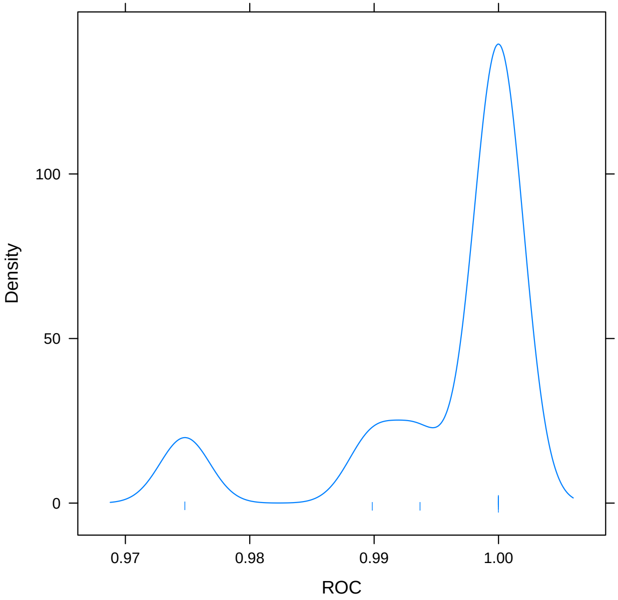


results : Weight decay and hidden neurons

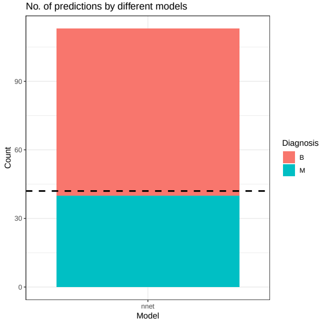


	size	decay
1	3	0.04216965

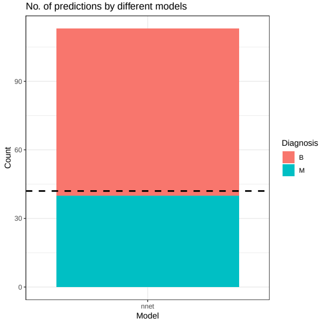
Result: Resampling distribution

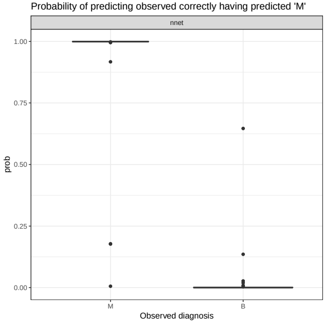


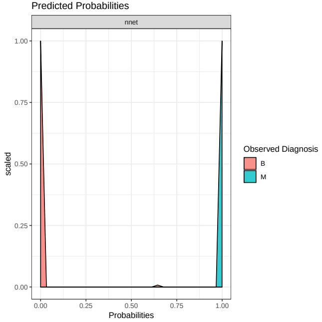
Result: Predictions



Result: Predictions

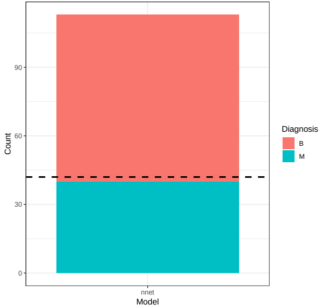




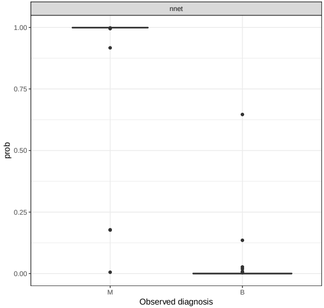


Result: Predictions

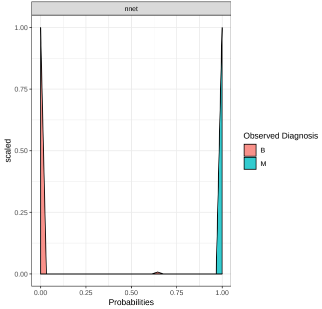
No. of predictions by different models



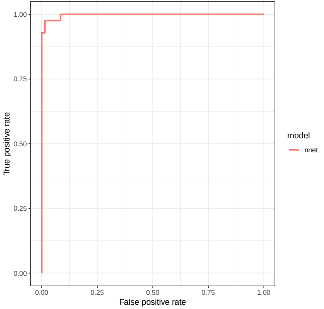
Probability of predicting observed correctly having predicted 'M'



Predicted Probabilities



ROCs comparison



Conclusions

- Discussed Gradient descent in neural network
- Appled neural network to classify breast cancer. The model provides a good classification of the data, with ROC of 0.99.

References

[1] Trevor, H., Robert, T., & JH, F. (2009). The elements of statistical learning: data mining, inference, and prediction. *Springer series in statistics*. Second Edition

[2] Tom M. Mitchell. (1997). Machine Learning *McGraw-Hill International Editions*.

[3] Internet sources (2019).