Nonlinear regression methods

Neural networks

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.

  Single model starting from the naive biology

  Learning problem as an unconstrained optimization problem

  $\rightsquigarrow$ Method of the steepest descent
  $\rightsquigarrow$ Newton's method
  $\rightsquigarrow$ Gauss-Neuton method

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.

  Single model starting from the naive biology

  Learning problem as an unconstrained optimization problem

  $\rightsquigarrow$ Method of the steepest descent
  $\rightsquigarrow$ Introduce the **back-propagation algorithm**

  The sequential updating of weights is the preferred method for its on-line implementation. For this mode of operation, the algorithm cycles through the training sample $\{(\mathbf{x}(n), \mathbf{d}(n))\}_{n=1}^{N}$ as follows:

1. **Inizialization**: Assuming that no prior information is available

   Pick the synaptic weights and thresholds from a uniform distribution whose mean is zero and whose variance is chosen to make the standard deviation of the induced local fields of the neurons lie at the transition between the linear and standards parts of the sigmoid activation function.

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.

  Single model starting from the naive biology

  Learning problem as an unconstrained optimization problem

  $\rightsquigarrow$ Method of the steepest descent
  $\rightsquigarrow$ Introduce the **back-propagation algorithm**

  The sequential updating of weights is the preferred method for its on-line implementation. For this mode of operation, the algorithm cycles through the training sample $\{(\mathbf{x}(n), \mathbf{d}(n))\}_{n=1}^{N}$ as follows:

2. **Presentations of Training Examples**: Present the network an epoch of training examples.

   For each example in the sample, ordered in some fashion, perform the sequence of forward and backward computations.

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.
  Single model starting from the naive biology
  Learning problem as an unconstrained optimization problem
  ⇝ Method of the steepest descent
  ⇝ Introduce the **back-propagation algorithm**
  The sequential updating of weights is the preferred method for its on-line implementation. For this mode of operation, the algorithm cycles through the training sample $\{(\mathbf{x}(n), \mathbf{d}(n))\}_{n=1}^{N}$ as follows:

3. **Forward Computation**: Compute the induced local fields and function signals of the network by proceeding forward through the network, layer by layer.

$$\underbrace{v_j^{(l)}(n)}_{\substack{\text{Induced local field for}\\ \text{neuron } j \text{ in layer } l}} = \sum_i \underbrace{w_{ji}^{(l)}(n)}_{\substack{\text{Synaptic weights of}\\ \text{neuron } j \text{ in the layer } l\\ \text{fed from neuron } i \text{ in layer } l-1}} \underbrace{y_i^{(l-1)}(n)}_{\substack{\text{Output signal of}\\ \text{neuron } i \text{ in layer } l-1}}$$

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.
  Single model starting from the naive biology
  Learning problem as an unconstrained optimization problem
  ⇝ Method of the steepest descent
  ⇝ Introduce the **back-propagation algorithm**
  The sequential updating of weights is the preferred method for its on-line implementation. For this mode of operation, the algorithm cycles through the training sample $\{(\mathbf{x}(n), \mathbf{d}(n))\}_{n=1}^{N}$ as follows:

3. **Forward Computation**: Assuming the use of a sigmoid function:

$$y_j^{(l)} = \varphi_j(v_j(n)) \quad \text{output signal of neuron } j \text{in layer } l$$

$$y_j^{(0)} = x_j(n) \quad \text{first hidden layer } l = 1$$

$$y_j^{(L)} = o_j(n) \quad \text{output layer } l = L$$

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.
  Single model starting from the naive biology
  Learning problem as an unconstrained optimization problem
  ⇝ Method of the steepest descent
  ⇝ Introduce the **back-propagation algorithm**
  The sequential updating of weights is the preferred method for its on-line implementation. For this mode of operation, the algorithm cycles through the training sample $\{(\mathbf{x}(n), \mathbf{d}(n))\}_{n=1}^{N}$ as follows:

3. **Forward Computation**: Compute the error signal

$$e_j(n) = d_j(n) - o_j(n)$$

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.
  Single model starting from the naive biology
  Learning problem as an unconstrained optimization problem
  ⇝ Method of the steepest descent
  ⇝ Introduce the **back-propagation algorithm**
  The sequential updating of weights is the preferred method for its on-line implementation. For this mode of operation, the algorithm cycles through the training sample $\{(\mathbf{x}(n), \mathbf{d}(n))\}_{n=1}^{N}$ as follows:

4. **Backward Computation**: Compute the local gradients δs:

$$\delta_j^{(l)} = \begin{cases} e_j^{(L)}(n)\varphi'(v_j^{(L)}(n)) & \text{for neuron } j \text{ in output layer } L \\ \varphi'(v_j^{l}(n))\sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n) & \text{for neuron } j \text{in hidden layer } l \end{cases}$$

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.
  Single model starting from the naive biology
  Learning problem as an unconstrained optimization problem
  ⇝ Method of the steepest descent
  ⇝ Introduce the **back-propagation algorithm**
  The sequential updating of weights is the preferred method for its on-line implementation. For this mode of operation, the algorithm cycles through the training sample $\{(\mathbf{x}(n), \mathbf{d}(n))\}_{n=1}^{N}$ as follows:

4. **Backward Computation**: Adjust the synaptic weights of the network in layer l according to the delta rule:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \eta \delta_j^{(l)}(n) y_i l - 1(n)$$

η is the learning-rate parameter.

A quick recap

During the last lectures, we did...

- **Neural network models for regression**.
  Single model starting from the naive biology
  Learning problem as an unconstrained optimization problem
  ⇝ Method of the steepest descent
  ⇝ Introduce the **back-propagation algorithm**
  The sequential updating of weights is the preferred method for its on-line implementation. For this mode of operation, the algorithm cycles through the training sample $\{(\mathbf{x}(n), \mathbf{d}(n))\}_{n=1}^{N}$ as follows:

5. **Iteration**: Iterate the forward and backward computations under points 3 and 4 by presenting new epochs of training examples to the network until the chosen stopping criterion is met.

Today's goal

Today, we going to continue with...

- **Neural network models for regression**.
  Issues on neural network training
  Neural network for the solubility problem

Reading list

- Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*, Springer (2014)
- Simon Haykin. *Neural Networks: A Comprehensive Foundation*, Pearson (2008, 3rd edition)

Issues in training neural networks

The neural network model is generally **overparametrized**, and the **optimization problem is nonconvex** and unstable unless certain guidelines are followed.

Optimization

Treating neural network as a nonlinear regression models, the weights are usually optimized to minimize the sum of the squared residuals.

This can be a challenging numerical optimization problem (there are no constraints on the weights of this complex nonlinear model).

The back-propagation algorithm is an efficient methodology that works with derivatives to find the optimal parameters.

- However, it is common that a solution to **this equation is not a global solution**, meaning that we cannot guarantee that the resulting set of parameters are uniformly better than any other set.

Issues in training neural networks

The neural network model is generally **overparametrized**, and the **optimization problem is nonconvex** and unstable unless certain guidelines are followed.

Overfitting

Neural networks have a tendency to over-fit the relationship between the predictors and the response due to the large number of regression coefficients.

Several different approaches have been proposed.

- **Early stopping**: The iterative algorithms for solving for the regression equations can be prematurely halted.

  It would stop the optimization procedure when some estimate of the error rate starts to increase (instead of some numerical tolerance to indicate that the parameter estimates or error rate are stable).

Issues in training neural networks

The neural network model is generally **overparametrized**, and the **optimization problem is nonconvex** and unstable unless certain guidelines are followed.

Overfitting

Neural networks have a tendency to over-fit the relationship between the predictors and the response due to the large number of regression coefficients.

Several different approaches have been proposed.

- **Weight decay**: we introduce a penalization method to regularize the model similar to ridge regression.

  We add a penalty for large regression coefficients so that any large value must have a significant effect on the model errors to be tolerated.

  The value of the penalizing parameter is a tuning parameter for the model and it should be specified together with the number of the hidden units.

  Since the weights are being summed, they should be on the same scale; hence the predictors should be centered and scaled prior to modeling.

Issues in training neural networks

The neural network model is generally **overparametrized**, and the **optimization problem is nonconvex** and unstable unless certain guidelines are followed.

Scaling of the inputs

The scaling of the inputs determines the effective scaling of the weights in the bottom layer, it can have a large effect on the quality of the final solution.

- It is best to standardize all inputs to have mean zero and standard deviation one.
- This ensures all inputs are treated equally in the regularization process, and allows one to choose a meaningful range for the random starting weights.
- With standardized inputs, it is typical to take random uniform weights over the range $[-0.7, +0.7]$.

Issues in training neural networks

The neural network model is generally **overparametrized**, and the **optimization problem is nonconvex** and unstable unless certain guidelines are followed.

Number of hidden units and layers

Generally speaking 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 if appropriate regularization is used.
- It is most common to put down a reasonably large number of units and train them with regularization.
- Some researchers use cross-validation to estimate the optimal number, but this seems unnecessary if cross-validation is used to estimate the regularization parameter.

Issues in training neural networks

The neural network model is generally **overparametrized**, and the **optimization problem is nonconvex** and unstable unless certain guidelines are followed.

Number of hidden units and layers

Generally speaking it is better to have too many hidden units than too few.

- Choice of the number of hidden layers is guided by background knowledge and experimentation.
- Each layer extracts features of the input for regression.

  Use of multiple hidden layers allows construction of hierarchical features at different levels of resolution.

Issues in training neural networks

The neural network model is generally **overparametrized**, and the **optimization problem is nonconvex** and unstable unless certain guidelines are followed.

Multiple minima

Given the challenge of estimating a large number of parameters, the fitted model finds parameter estimates that are locally optimal

- ⇝ The algorithm converges, but the resulting parameter estimates are unlikely to be the globally optimal estimates.

  Very often, different locally optimal solutions can produce models that are very different but have nearly equivalent performance.
- ⇝ As an alternative, several models can be created using different starting values and averaging the results of these model to produce a more stable prediction.

  Such model averaging often has a significantly positive effect on neural networks.

Issues in training neural networks

The neural network model is generally **overparametrized**, and the **optimization problem is nonconvex** and unstable unless certain guidelines are followed.

Multiple minima

The averaged neural network models are often adversely affected by high correlation among the predictor variables

For mitigating this issue we can

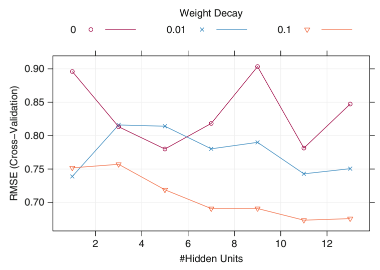
- pre-filter the predictors to **remove the predictors that are associated with high correlations**.
- Use a feature extraction technique, such as principal component analysis, can be used prior to modeling to eliminate correlations.

Both these approaches is that fewer model terms need to be optimized, thus improving computation time.

Neural network for the solubility data

For the solubility data, Kuhn and Johnson used averaged neural networks.

- Three different weight decay values were evaluated along with a single hidden layer with sizes ranging between 1 and 13 hidden units.
- The final predictions are the averages of five different neural networks created using different initial parameter values.



Increasing the amount of weight decay clearly improved model performance, while more hidden units also reduce the model error.

The optimal model used 11 hidden units with a total of 2,531 coefficients.

The performance of the model is fairly stable for a high degree of regularization, so smaller models could also be effective for these data.