



Introduction to Statistical Machine Learning

Cheng Soon Ong & Christian Walder

Machine Learning Research Group
Data61 | CSIRO

and

College of Engineering and Computer Science
The Australian National University

Canberra

February – June 2017

(Many figures from C. M. Bishop, "Pattern Recognition and Machine Learning")

Outlines

- Overview
- Introduction
- Linear Algebra
- Probability
- Linear Regression 1
- Linear Regression 2
- Linear Classification 1
- Linear Classification 2
- Kernel Methods
- Sparse Kernel Methods
- Neural Networks 1
- Neural Networks 2
- Continuous Latent Variables
- Autoencoders
- Graphical Models 1
- Graphical Models 2
- Graphical Models 3
- Sampling
- Mixture Models and EM 1
- Mixture Models and EM 2
- Sequential Data 1
- Sequential Data 2
- Combining Models



Part X

Neural Networks 2

Review

Error Backpropagation

*Regularisation in Neural
Networks*

*Bayesian Neural
Networks*



- The composition of two functions is given by

$$f \circ g(x) = f(g(x))$$

First apply g then f .

- Let f and g be differentiable functions with derivatives f' and g' respectively
- Chain rule

$$\frac{d}{dx}f(g(x)) = f'(g(x))g'(x)$$

- If we write $u = g(x)$ and $y = f(u)$,

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



- Goal: Efficiently update the weights in order to find a local minimum of some error function $E(\mathbf{w})$ utilizing the gradient of the error function.
- Core ideas :
 - 1 Propagate the errors backwards through the network to efficiently calculate the gradient.
 - 2 Update the weights using the calculated gradient.
- Sequential procedure : Calculate gradient and update weights for each data/target pair.
- Batch procedure : Collect gradient information for all data/target pairs for the same weight setting. Then adjust the weights.
- Main question in both cases: How to calculate the gradient of $E(\mathbf{w})$ given one data/target pair?

Review

Error Backpropagation

*Regularisation in Neural
Networks*

*Bayesian Neural
Networks*



- Assume the error is a sum over errors for each data/target pair

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}).$$

- After applying input \mathbf{x}_n to the network, we get the output \mathbf{y}_n and calculate the error $E_n(\mathbf{w})$.
- What is the gradient for one such term $E_n(\mathbf{w})$?
- Note : In the following, we will drop the n on weights \mathbf{w} and targets \mathbf{t} in order to unclutter the equations.
- Notation: Input pattern is \mathbf{x}_n .
Scalar x_i is the i^{th} component of the input pattern \mathbf{x}_n .

Error Backpropagation - Simplified Model



- Simple linear model **without** hidden layers
- One layer only, identity function as activation function!

$$y_k = \sum_l w_{kl} x_l,$$

and error after applying input \mathbf{x}_n

$$E_n(\mathbf{w}) = \frac{1}{2} \sum_{k=1} (y_k - t_k)^2.$$

- The gradient with respect to w_{ji} is now

$$\begin{aligned} \frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} &= \sum_{k=1} (y_k - t_k) \frac{\partial}{\partial w_{ji}} y_k = \sum_{k=1} (y_k - t_k) \frac{\partial}{\partial w_{ji}} \sum_l w_{kl} x_l \\ &= \sum_{k=1} (y_k - t_k) \sum_l x_l \delta_{jk} \delta_{il} \\ &= (y_j - t_j) x_i. \end{aligned}$$

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks

Error Backpropagation - Simplified Model



- Do the same using the directional derivative:
input vector $\mathbf{x} \in \mathbb{R}^{D_1}$, output vector $\mathbf{y} \in \mathbb{R}^{D_2}$

$$\mathbf{y} = \mathbf{W} \mathbf{x} \quad \mathbf{W} \in \mathbb{R}^{D_2 \times D_1},$$

and error after applying input training pair (\mathbf{x}, \mathbf{t})

$$E_n(\mathbf{W}) = \frac{1}{2}(\mathbf{y} - \mathbf{t})^T(\mathbf{y} - \mathbf{t}) = \frac{1}{2}(\mathbf{W}\mathbf{x} - \mathbf{t})^T(\mathbf{W}\mathbf{x} - \mathbf{t}).$$

- The directional derivative with respect to \mathbf{W} is now

$$\mathcal{D}E_n(\mathbf{W})(\xi) = \frac{1}{2}((\xi\mathbf{x})^T(\mathbf{y} - \mathbf{t}) + (\mathbf{y} - \mathbf{t})^T\xi\mathbf{x}) = \mathbf{x}^T\xi^T(\mathbf{y} - \mathbf{t})$$

- With canonical inner product $\langle A, B \rangle = \text{tr}\{A^T B\}$ the gradient of $E_n(\mathbf{W})(\xi)$ is

$$\mathcal{D}E_n(\mathbf{W})(\xi) = \text{tr} \left\{ \underbrace{\mathbf{x}^T \xi^T (\mathbf{y} - \mathbf{t})}_{\text{just a number}} \right\} = \text{tr} \left\{ \xi^T \underbrace{(\mathbf{y} - \mathbf{t}) \mathbf{x}^T}_{\text{gradient}} \right\}$$

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



- The gradient

$$\nabla_{\mathbf{W}} E_n(\mathbf{W}) = (\mathbf{y} - \mathbf{t}) \mathbf{x}^T$$

or in components

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = (y_j - t_j) x_i.$$

looks like the product of the output error $(y_j - t_j)$ with the input x_i associated with an edge for w_{ji} in the network diagram.

- Can we generalise this idea to nonlinear activation functions?

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



Review

*Error Backpropagation**Regularisation in Neural
Networks**Bayesian Neural
Networks*

Error Backpropagation

- Now consider a network with **nonlinear** activation functions $h(\cdot)$ composed with the sum over the inputs z_i in one layer and z_j in the next layer connected by edges with weights w_{ji}

$$a_j = \sum_i w_{ji} z_i$$
$$z_j = h(a_j).$$

- Use the **chain rule** to calculate the gradient

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \frac{\partial E_n(\mathbf{w})}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i,$$

where we defined the **error** $\delta_j = \frac{\partial E_n(\mathbf{w})}{\partial a_j}$

- Same intuition as before: gradient is output error times the input associated with the edge for w_{ji} .



- Need to calculate the errors δ in EACH layer.

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \delta_j z_i \qquad \delta_j = \frac{\partial E_n(\mathbf{w})}{\partial a_j}$$

- For the output units, we have

$$\delta_k = y_k - t_k.$$

- For the hidden units we calculate

$$\delta_j = \frac{\partial E_n(\mathbf{w})}{\partial a_j} = \sum_k \frac{\partial E_n(\mathbf{w})}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j},$$

using the definition of δ_k .

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



- Express a_k as a function of the incoming a_j

$$a_k = \sum_j w_{kj} z_j = \sum_j w_{kj} h(a_j),$$

- and differentiate

$$\frac{\partial a_k}{\partial a_j} = w_{kj} \frac{\partial h(a_j)}{\partial a_j} = w_{kj} \frac{\partial h(s)}{\partial s} \Bigg|_{s=a_j} = w_{kj} h'(a_j).$$

- Finally, we get for the error in the previous layer

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k.$$

Review

Error Backpropagation

Regularisation in Neural
Networks

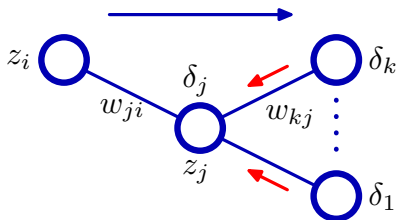
Bayesian Neural
Networks



- The backpropagation formula

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k.$$

- Functional form of $h'(\cdot)$ is known, because we choose the activation function $h(\cdot)$.



Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



Review

Error Backpropagation

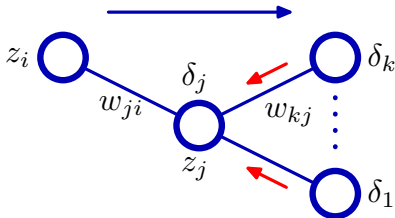
Regularisation in Neural
NetworksBayesian Neural
Networks

Error Backpropagation Algorithms

- 1 Apply the input vector \mathbf{x} to the network and forward propagate through the network to calculate all activations and outputs of each unit.
- 2 Backpropagate the errors through the network.
- 3 Calculate all components of ∇E_n by

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \delta_j z_i$$

- 4 Update the weights \mathbf{w} using $\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}}$.





- For batch processing, we repeat backpropagation for each pattern in the training set and then sum over all patterns

$$\frac{\partial E(\mathbf{w})}{\partial w_{ji}} = \sum_{n=1}^N \frac{\partial E_n(\mathbf{w})}{\partial w_{ji}}$$

- Backpropagation can be generalised by assuming that each node has a different activation function $h(\cdot)$.

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks

Efficiency of Error Backpropagation



- As the number of weights is usually much larger than the number of units (the network is well connected), the complexity of calculating the gradient $\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}}$ via error backpropagation is of $O(W)$ where W is the number of weights.
- Compare this to **numerical differentiation** using

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji})}{\epsilon} + O(\epsilon)$$

or the numerically more stable (fewer round-off errors)
symmetric differences

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{ji}} = \frac{E_n(w_{ji} + \epsilon) - E_n(w_{ji} - \epsilon)}{2\epsilon} + O(\epsilon^2)$$

which both need $O(W^2)$ operations.

Review

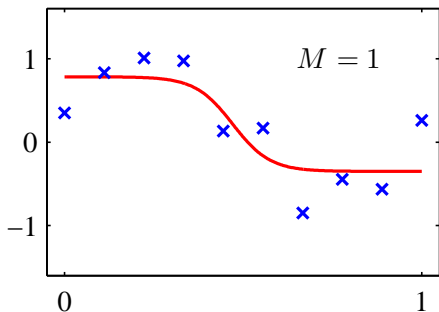
Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



- Number of input and output nodes determined by the application.
- Number of hidden nodes is a free parameter.



Training a two-layer network with 1 hidden node.

Review

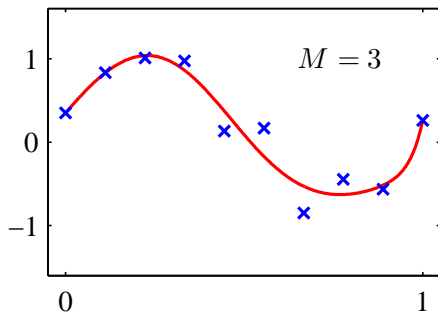
Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



- Number of input and output nodes determined by the application.
- Number of hidden nodes is a free parameter.



Training a two-layer network with 3 hidden nodes.

Review

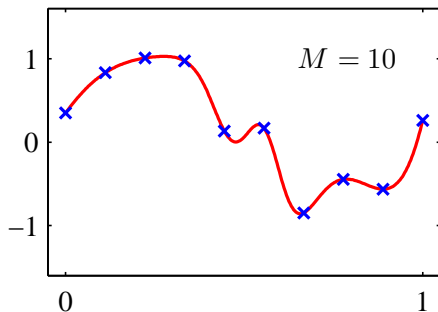
Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



- Number of input and output nodes determined by the application.
- Number of hidden nodes is a free parameter.



Training a two-layer network with 10 hidden nodes.

Review

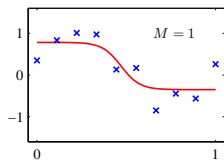
Error Backpropagation

Regularisation in Neural
Networks

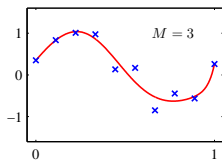
Bayesian Neural
Networks



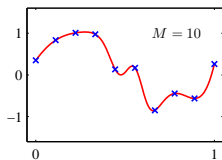
- Model complexity matters again.



$M = 1$



$M = 3$



$M = 10$

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



- But the relation between generalisation error and the number of hidden units M is more complex than for fitting a polynomial. Reason : presence of local minima in the error function for the neural network.
- As before, we can use the **regularised error**

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$$



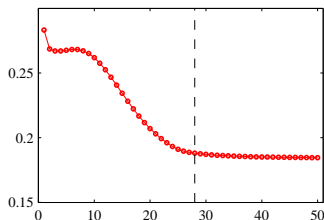
- Stop training at the minimum of the validation set error.

Review

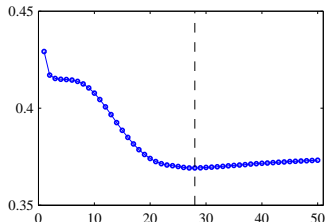
Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



Training set error.



Validation set error.



- If input data should be invariant with respect to some transformations, we can utilise this for training.
- Use training patterns including these transformations (e.g. handwritten digits translated in the input space).
- Or create extra artificial input data by applying several transformations to the original input data.
- Alternatively, preprocess the input data to remove the transformation.
- Or use **convolutional neural networks** (e.g. in image processing where close pixels are more correlated than far away pixels; therefore extract local features first and later feed into a network extracting higher-order features).

Review

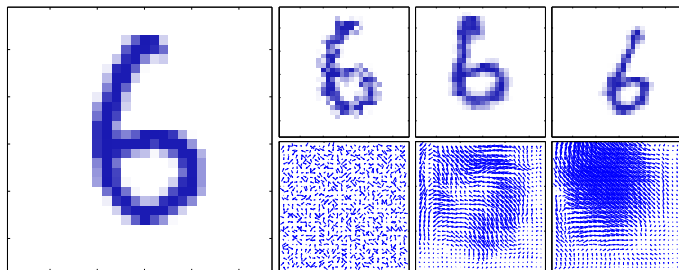
Error Backpropagation

*Regularisation in Neural
Networks*

*Bayesian Neural
Networks*



- Create synthetic data by warping handwritten digits.



Left: Original digitised image. Right : Examples of warped images (above) and their corresponding displacement fields (below).

Review

Error Backpropagation

*Regularisation in Neural
Networks*

*Bayesian Neural
Networks*



- Predict a single target t from a vector of inputs \mathbf{x}
- Assume conditional distribution to be Gaussian with precision β

$$p(t | \mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t | y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

- Prior distribution over weights \mathbf{w} is also assumed to be Gaussian

$$p(\mathbf{w} | \alpha) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} \mathbf{I})$$

- For an i.i.d training data set $\{\mathbf{x}_n, t_n\}_{n=1}^N$, the likelihood of the targets $\mathcal{D} = \{t_1, \dots, t_N\}$ is

$$p(\mathcal{D} | \mathbf{w}, \beta) = \prod_{n=1}^N \mathcal{N}(t_n | y(\mathbf{x}_n, \mathbf{w}), \beta^{-1})$$

- Posterior distribution

$$p(\mathbf{w} | \mathcal{D}, \alpha, \beta) \propto p(\mathbf{w} | \alpha) p(\mathcal{D} | \mathbf{w}, \beta)$$



- But $y(\mathbf{x}, \mathbf{w})$ is **nonlinear**, and therefore we can no longer calculate the posterior in closed form.
- Use Laplace approximation
 - 1 Find a (local) maximum \mathbf{w}_{MAP} of the posterior via **numerical optimisation**.
 - 2 Evaluate the matrix of second derivatives of the negative log posterior distribution.
- Find a (local) maximum using the log-posterior

$$\ln p(\mathbf{w} | \mathcal{D}, \alpha, \beta) = -\frac{\alpha}{2} \mathbf{w}^T \mathbf{w} - \frac{\beta}{2} \sum_{n=1}^N (y(\mathbf{x}, \mathbf{w}) - t_n)^2 + \text{const}$$

- Find the matrix of second derivatives of the negative log posterior distribution

$$\mathbf{A} = -\nabla \nabla \ln p(\mathbf{w} | \mathcal{D}, \alpha, \beta) = \alpha \mathbf{I} + \beta \mathbf{H}$$

where \mathbf{H} is the Hessian matrix of the sum-of-squares error function with respect to the components of \mathbf{w} .

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks



- Having \mathbf{w}_{MAP} , and \mathbf{A} , we can approximate the posterior by a Gaussian

$$q(\mathbf{w} | \mathcal{D}, \alpha, \beta) = \mathcal{N}(\mathbf{w} | \mathbf{w}_{MAP}, \mathbf{A}^{-1})$$

- Similarly for the predictive distribution (without proof)

$$p(t | \mathbf{x}, \mathcal{D}, \alpha, \beta) = \mathcal{N}(t | y(\mathbf{x}, \mathbf{w}_{MAP}), \sigma^2(\mathbf{x}))$$

where

$$\sigma^2(\mathbf{x}) = \beta^{-1} + \mathbf{g}^T \mathbf{A}^{-1} \mathbf{g}$$

and

$$\mathbf{g} = \nabla_{\mathbf{w}} y(\mathbf{x}, \mathbf{w}) |_{\mathbf{w}=\mathbf{w}_{MAP}}.$$

(Remember predictive distribution in the linear regression case?)

- uncertainty because of intrinsic noise on the target: β^{-1}
- uncertainty in the model parameter \mathbf{w} : $\mathbf{g}^T \mathbf{A}^{-1} \mathbf{g}$

Review

Error Backpropagation

Regularisation in Neural
Networks

Bayesian Neural
Networks