COURSE 02457

Signal Processing in Non-linear Systems:

Lecture 6

- Perceptrons
- Gradient descent
- Computing gradients
- Gradient descent and line searches
- Conjugate gradients
- Local quadratic approximation to the cost function
- Newtons method
- Outer product (Gauss) approximation
- Diagonal (pseudo-Gauss) approximation
- Test error estimation

The Perceptron

- The Perceptron is due to Rosenblatt (1958):

$$y(\mathbf{x}) = \tanh\left(\mathbf{w}^\top \mathbf{x} + w_0\right)$$

- Note the similarity to the logistic regression model which was derived as the discriminant function for discrimination between to normal distribution with identical covariance matrices

The Multi-layer Perceptron

- The Multi-layer Perceptron (MLP) was considered by Minsky and Papert in the early sixties:

$$
\begin{aligned}
y(\mathbf{x}) &= \tanh\left(\sum_{j=0}^{n_H} W_j h_j(\mathbf{x})\right) \\
z_j(\mathbf{x}) &= \tanh\left(\mathbf{w}^\top \mathbf{x} + w_0\right) \\
z_0 &= 1
\end{aligned}
$$

- The hidden units represent the inputs so that the output unit can solve a linear discrimination problem

Feed-forward and feed-back nets

- Feed-forward nets have no loops, hence their evaluation is uniquely defined.
- General MLP's can have loops, hence need an associated evaluation rule. Only relevant for dynamic models (time series modelling). Sometimes also called recursive models.
- For time series models: Feed-forward models have finite memory, while recursive networks implement (arbitrary) long memory.

The MLP learning problem

- Let a training set be given by $\mathcal{D} = \{(t^1, \mathbf{x}^1), ..., (t^N, \mathbf{x}^N)\}$.
- The mean square error of the model $y(\mathbf{x}; \mathbf{w})$ is given by

$$E = \frac{1}{2} \sum_{n=1}^{N} (y(\mathbf{x}^n; \mathbf{w}) - t^n)^2$$

- Weight decay is a means of soft capacity control, augmented cost function

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{1}{2} \nu \mathbf{w}^T \mathbf{w}$$

Approximation capabilities

- Linear output MLP is

$$\begin{aligned} y(\mathbf{x}) &= \sum_{j=0}^{n_H} W_j h_j(\mathbf{x}) \\ z_j(\mathbf{x}) &= \tanh\left(\mathbf{w}^\top \mathbf{x} + w_{0,j}\right) \\ z_0 &= 1 \end{aligned}$$

- is a universal approximation tool for continuous functions.
- The set of linear output MLP's is dense in the continuous functions on a compact subset of a vector space: If given an ϵ and a continuous target function $f(\mathbf{x})$ on the set Ω, we can find an MLP network for which

$$|y(\mathbf{x}; \mathbf{w}) - f(\mathbf{x})| < \epsilon, \forall \mathbf{x} \in \Omega$$

The Multi-layer Perceptron

- The Multi-layer Perceptron (MLP)

$$
\begin{aligned}
y_l(\mathbf{x}) &= \sum_{j=0}^{n_H} \mathbf{w}_{l,j} z_j(\mathbf{x}) \\
z_j(\mathbf{x}) &= \tanh\left(\mathbf{w}_j^\top \mathbf{x} + w_{j,0}\right) \\
z_0 &= 1
\end{aligned}
$$

- The hidden units represent the inputs so that the output unit can solve a linear discrimination problem

The MLP learning problem

- Let a training set be given by $\mathcal{D} = \{(t^1, \mathbf{x}^1), ..., (t^N, \mathbf{x}^N)\}$.
- The mean square error of the model $y(\mathbf{x}; \mathbf{w})$ is given by

$$E = \frac{1}{2} \sum_{n=1}^{N} (y(\mathbf{x}^n; \mathbf{w}) - t^n)^2$$

Gradient descent optimization

- Objective: to solve the equation $\nabla E = 0$

$$\begin{aligned} \mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)} \\ \Delta\mathbf{w}^{(\tau)} &= -\eta \nabla E|_{\mathbf{w}^{(\tau)}} \end{aligned}$$

- η is the learning parameter
- η can be too small: convergence very slow
- η can be too large: oscillatory behavior

Backprop for the two layer network

- We compute the gradient w.r.t. any weight in first or second layer u

$$\begin{aligned} E &= \frac{1}{2}\sum_{n=1}^{N}(y(\mathbf{x}^n;\mathbf{w}) - t^n)^2 \\ \frac{\partial E}{\partial u} &= \frac{1}{2}\sum_{n=1}^{N}\frac{\partial}{\partial u}(y(\mathbf{x}^n;\mathbf{w}) - t^n)^2 \\ &= \sum_{n=1}^{N}(y(\mathbf{x}^n;\mathbf{w}) - t^n)\frac{\partial y(\mathbf{x}^n;\mathbf{w})}{\partial u} \end{aligned}$$

- The network derivative for an output unit weight $w_{j'}$is given by

$$\begin{aligned} \frac{\partial y(\mathbf{x}^n;\mathbf{w})}{\partial w_{j'}} &= \frac{\partial}{\partial w_{j'}}\sum_{j=0}^{n_H}\mathbf{w}_j z_j(\mathbf{x}) \\ &= \sum_{j=0}^{n_H}\frac{\partial}{\partial w_{j'}} w_j z_j(\mathbf{x}) \\ &= z_{j'}(\mathbf{x}) \end{aligned}$$

Backprop for the two layer net cont'd

- The network derivative for a hidden unit weight $w_{j',k'}$is given by

$$\begin{aligned}
\frac{\partial y(\mathbf{x}^n;\mathbf{w})}{\partial w_{j',k'}} &= \frac{\partial}{\partial w_{j',k'}} \sum_{j=0}^{n_H} w_j z_j(\mathbf{x}) \\
&= \sum_{j=0}^{n_H} w_j \frac{\partial}{\partial w_{j',k'}} z_j(\mathbf{x}) \\
&= w_{j'} \frac{\partial}{\partial w_{j',k'}} \tanh\left(\sum_{k=0}^{n_I} w_{j,k} x_k^n\right) \\
&= w_{j'} \left(1 - \tanh^2\left(\sum_{k=0}^{n_I} w_{j,k} x_k^n\right)\right) \frac{\partial}{\partial w_{j',k'}} \sum_{k=0}^{n_I} w_{j,k} x_k^n \\
&= w_{j'} \left(1 - z_{j'}^2\right) x_{k'}^n
\end{aligned}$$

Backprop for the two layer net cont'd

- Combining we get for the output weight

$$\begin{aligned}\frac{\partial E}{\partial w_j} &= \sum_{n=1}^{N}(y(\mathbf{x}^n) - t^n)z_j(\mathbf{x}^n) \\ &\equiv \sum_{n=1}^{N}\delta^n z_j(\mathbf{x}^n)\end{aligned}$$

- and for the hidden weight

$$\begin{aligned}\frac{\partial E}{\partial w_{j,k}} &= \sum_{n=1}^{N}(y(\mathbf{x}^n) - t^n)w_j\left(1 - z_j^2(\mathbf{x}^n)\right)x_k^n \\ &\equiv \sum_{n=1}^{N}\delta_j^n x_k^n\end{aligned}$$

The general Backprop rule

- Consider a hidden unit $z_j = g(a_j)$, where $a_j^n = \sum_i w_{j,i} z_i(\mathbf{x}^n)$
- ... then the derivative can be expressed

$$\begin{aligned}\frac{\partial E}{\partial w_{ji}} &= \sum_{j',n} \frac{\partial E^n}{\partial a_{j'}^n} \frac{\partial a_{j'}^n}{\partial w_{ji}} \\ &= \sum_n \frac{\partial E^n}{\partial a_j^n} \frac{\partial a_j^n}{\partial w_{ji}}\end{aligned}$$

- Let $\delta_j^n = \frac{\partial E^n}{\partial a_j^n}$, note also $\frac{\partial a_j^n}{\partial w_{ji}} = z_i(\mathbf{x}^n)$, this leads to

$$\frac{\partial E}{\partial w_{ji}} = \sum_n \delta_j^n z_i(\mathbf{x}^n)$$

- Computing the $\delta' s$

$$\begin{aligned}\delta_j^n &\equiv \frac{\partial E^n}{\partial a_j} = \sum_k \frac{\partial E^n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \\ \delta_j^n &= g'(a_j^n) \sum_k w_{k,j} \delta_k^n\end{aligned}$$

Gradient descent optimization revisited

- Objective: to solve the equation $\nabla E = 0$

$$\begin{aligned} \mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} + \Delta\mathbf{w}^{(\tau)} \\ \Delta\mathbf{w}^{(\tau)} &= -\eta \nabla E|_{\mathbf{w}^{(\tau)}} \end{aligned}$$

- η is the learning parameter
- η can be too small: convergence very slow
- η can be too large: oscillatory behavior
- Find η by line search along the search direction $\mathbf{d}^{(\tau)} = -\nabla E|_{\mathbf{w}^{(\tau)}}$:

$$\begin{aligned} \mathbf{w}^{(\tau+1)} &= \mathbf{w}^{(\tau)} + \eta^{(\tau)}\mathbf{d}^{(\tau)} \\ E(\eta) &= E(\mathbf{w}^{(\tau)} + \eta\mathbf{d}^{(\tau)}) \end{aligned}$$

Conjugate gradient method

- Objective: to solve the equation $\nabla E = 0$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta \mathbf{d}^{(\tau)}$$

- Let $\mathbf{g}^{(\tau)} \equiv \nabla E(\mathbf{w}^{(\tau)})$, the optimal η solves

$$\begin{aligned} \frac{\partial}{\partial \eta} E(\mathbf{w}^{(\tau)} + \eta \mathbf{d}^{(\tau)}) &= 0 \\ \nabla E(\mathbf{w}^{(\tau)} + \eta \mathbf{d}^{(\tau)})^{\top} \mathbf{d}^{(\tau)} &= 0 \\ \mathbf{g}^{(\tau+1)\top} \mathbf{d}^{(\tau)} &= 0 \end{aligned}$$

- Gradient at optimal point is orthogonal to the search direction!

$$\mathbf{g}(\mathbf{w}^{(\tau+1)})^{\top} \mathbf{d}^{(\tau)} = 0$$

- Choose the search direction so that this property also holds between new and old search direction:

$$\mathbf{g}(\mathbf{w}^{(\tau+1)} + \eta \mathbf{d}^{(\tau+1)})^{\top} \mathbf{d}^{(\tau)} = 0$$

Conjugate gradient method cont'd

- Gradient at optimal point is orthogonal to the search direction! If we search such orthogonal directions we keep minimal interference.

$$\mathbf{g}(\mathbf{w}^{(\tau+1)})^\top \mathbf{d}^{(\tau)} = 0$$

- Choose the search direction so that this property also holds between new and old search direction:

$$\mathbf{g}(\mathbf{w}^{(\tau+1)} + \eta \mathbf{d}^{(\tau+1)})^\top \mathbf{d}^{(\tau)} = 0$$

- expand to second order:

$$\mathbf{g}(\mathbf{w}^{(\tau+1)} + \eta \mathbf{d}^{(\tau+1)}) \approx \mathbf{g}(\mathbf{w}^{(\tau+1)}) + \eta \mathbf{H} \mathbf{d}^{(\tau+1)}$$

$$\left(\mathbf{g}(\mathbf{w}^{(\tau+1)}) + \eta \mathbf{H} \mathbf{d}^{(\tau+1)}\right)^\top \mathbf{d}^{(\tau)} = 0$$

$$\mathbf{d}^{(\tau+1)\top} \mathbf{H} \mathbf{d}^{(\tau)} = 0$$

- This defines the conjugate directions.

Conjugate gradient method cont'd

- A complete set of conjugate directions can be found for a quadratic problem:

$$\mathbf{d}^{(\tau+1)} = -\nabla E^{(\tau+1)} + \beta^{(\tau)}\mathbf{d}^{(\tau)}$$

- with the three alternative definitions ($\mathbf{g} \equiv \nabla E$) (Hestenes-Stiefel, Polak-Ribiere, Fletcher-Reeves):

$$\beta^{(\tau)} = \frac{\mathbf{g}^{(\tau+1)\top}\left(\mathbf{g}^{(\tau+1)} - \mathbf{g}^{(\tau)}\right)}{\mathbf{d}^{(\tau)\top}\left(\mathbf{g}^{(\tau+1)} - \mathbf{g}^{(\tau)}\right)}$$

$$\beta^{(\tau)} = \frac{\mathbf{g}^{(\tau+1)\top}\left(\mathbf{g}^{(\tau+1)} - \mathbf{g}^{(\tau)}\right)}{\mathbf{g}^{(\tau)\top}\mathbf{g}^{(\tau)}}$$

$$\beta^{(\tau)} = \frac{\mathbf{g}^{(\tau+1)\top}\mathbf{g}^{(\tau)}}{\mathbf{g}^{(\tau)\top}\mathbf{g}^{(\tau)}}$$

- Furthermore, if we perform a perfect line search at every step the algorithm will converge in W steps for the quadratic problem. For the general costfunction nothing definitive is known, but it should work close to the minimum....

Newtons method in 1D

- Let the costfunction be approximated,

$$E(w) = E(w^*) + \frac{1}{2}H(w - w^*)^2$$

- The derivative is given by

$$\frac{\partial E}{\partial w}(w) = \frac{\partial E}{\partial w}(w^*) + H(w - w^*)$$

$$\frac{\partial E}{\partial w}(w) = H(w - w^*)$$

- This means that the distance from w to w^* is

$$w^* = w - H^{-1}\frac{\partial E}{\partial w}(w)$$

- Hence the optimal step is $\Delta w = -H^{-1}\frac{\partial E}{\partial w}(w)$

Newtons method in multiple dimensions

- At a minimum $\nabla E = 0$

$$E(\mathbf{w}) \approx E(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^T \mathbf{H} (\mathbf{w} - \mathbf{w}^*)$$

$$\nabla E(\mathbf{w}) \approx \mathbf{H} (\mathbf{w} - \mathbf{w}^*)$$

- We find the optimal multivariate step is given by

$$\mathbf{w}^* = \mathbf{w} - \mathbf{H}^{-1} \nabla E(\mathbf{w})$$

- this is the Newton direction, for a quadratic problem this solves the optimization problem in one iteration!

Hessian for a least squares problem

- The least squares costfunction

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y^n - d^n)^2$$

- The first derivative is

$$\frac{\partial E}{\partial \mathbf{w}} = \sum_{n=1}^{N}(y^n - d^n)\frac{\partial y^n}{\partial \mathbf{w}}$$

- The second derivative is

$$\frac{\partial^2 E}{\partial \mathbf{w}\partial \mathbf{w}^\top} = \sum_{n=1}^{N}\frac{\partial y^n}{\partial \mathbf{w}}\frac{\partial y^n}{\partial \mathbf{w}}^\top + \sum_{n=1}^{N}(y^n - d^n)\frac{\partial^2 y^n}{\partial \mathbf{w}\partial \mathbf{w}^\top}$$

- The Gauss-Newton or outer product approximation is

$$\frac{\partial^2 E}{\partial \mathbf{w}\partial \mathbf{w}^\top} \approx \sum_{n=1}^{N}\frac{\partial y^n}{\partial \mathbf{w}}\frac{\partial y^n}{\partial \mathbf{w}}^\top$$

- The pseudo-Gauss-Newton approximation is to ignore the off-diagonal terms