Neural Network Training

Sargur Srihari

Topics in Network Training

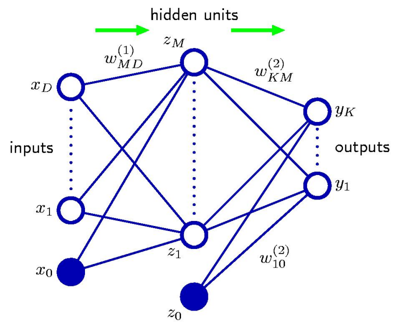
0. Neural network parameters
   - Probabilistic problem formulation
   - Specifying the activation and error functions for
     - Regression
     - Binary classification
     - Multi-class classification
1. Parameter optimization
2. Local quadratic approximation
3. Use of gradient optimization
4. Gradient descent optimization

Neural Network parameters

- Linear models for regression and classification can be represented as

$$y(\boldsymbol{x},\boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x})\right)$$

  - which are linear combinations of basis functions $\phi_j(\boldsymbol{x})$
- In a neural network the basis functions $\phi_j(\boldsymbol{x})$ depend on parameters
  - During training allow these parameters to be adjusted along with the coefficients w_j

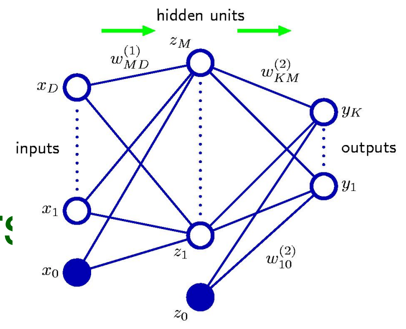


Specifying Error Function for Network Training

- Neural networks perform a transformation
  - vector $\boldsymbol{x}$ of input variables to vector $\boldsymbol{y}$ of output variables
  - For sigmoid activation function

$$\boxed{y_k(\boldsymbol{x},\boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)\right)}$$

D input variables
M hidden units



  - Where vector $\boldsymbol{w}$ consists of all weight and bias parameters
- To determine $\boldsymbol{w}$, simple analogy with curve fitting
  - minimize sum-of-squared errors function
  - Given set of input vectors $\{\boldsymbol{x}_n\}$, $n=1,..,N$ and target vectors $\{\boldsymbol{t}_n\}$ minimize the error function

$$\boxed{E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} || \boldsymbol{y}(\boldsymbol{x}_n,\boldsymbol{w}) - \boldsymbol{t}_n ||^2}$$

N training vectors

- We can provide a more general view of network training
  - by first giving a probabilistic interpretation to the network outputs
    - We have seen many advantages of providing probabilistic predictions
    - It will also provide a clearer motivation for choice of output nonlinearity and choice of error function

Probabilistic View of Network: Activation f, Error E

$$\boxed{y_k(\boldsymbol{x},\boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)\right)}$$

1. Regression
   - f : identity
   - t: Gaussian, with mean $y(\boldsymbol{x},\boldsymbol{w})$, precision β
   - E: Maximum Likelihood (sum of squares)

$$\boxed{y(\boldsymbol{x},\boldsymbol{w}) = \sum_{j=1}^{M} w_j^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)}$$

$$\boxed{E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} \| \boldsymbol{y}(\boldsymbol{x}_n,\boldsymbol{w}) - \boldsymbol{t}_n \|^2}$$

2. Binary Classification
   - f: Logistic sigmoid
   - t: Bernoulli $(0,1)$. With mean $\sigma(\boldsymbol{w}^T\boldsymbol{x})$
   - E: Maximum likelihood (cross-entropy)

$$\boxed{y(\boldsymbol{x},\boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_j^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)\right)}$$

$$\boxed{E(\boldsymbol{w}) = -\sum_{n=1}^{N}\left\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\right\}}$$

3. Multiclass(K binary classifications)
   - f: Logistic sigmoid
   - E: MLE-Bernoulli

$$\boxed{y_k(\boldsymbol{x},\boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)\right)}$$

$$\boxed{E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\left\{t_{nk} \ln y_{nk} + (1-t_{nk})\ln(1-y_{nk})\right\}}$$

4. Standard Multiclass
   - f: Softmax t: multinoulli
   - E: Cross-entropy error function

$$\boxed{y_k(\boldsymbol{x},\boldsymbol{w}) = \frac{\exp(a_k(\boldsymbol{x},\boldsymbol{w}))}{\sum_j \exp(a_j(\boldsymbol{x},\boldsymbol{w}))}}$$

$$\boxed{E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y_k(\boldsymbol{x}_n,\boldsymbol{w})}$$

$$\boxed{a_k = \sum_{i=1}^{M} w_{ki}^{(2)} x_i + w_{k0}^{(2)}}$$

1. Probabilistic Error Function for Regression

- Output is a single target variable t that can take any real value
- Assuming t is Gaussian distributed with an $\boldsymbol{x}$-dependent mean

$$\boxed{p(t \mid \boldsymbol{x}, \boldsymbol{w}) = N(t \mid y(\boldsymbol{x}, \boldsymbol{w}), \beta^{-1})}$$

- Likelihood function

$$p(\mathrm{t} \mid \boldsymbol{x}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} N(t_n \mid y(\boldsymbol{x}_{\mathrm{n}}, \boldsymbol{w}), \beta^{-1})$$

- Taking negative logarithm, we get the error function

$$\boxed{\frac{\beta}{2} \sum_{n=1}^{N} \left\{ y(\boldsymbol{x}_n, \boldsymbol{w}) - t_n \right\}^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln(2\pi)}$$

  - which is minimized to learn parameters $\boldsymbol{w}$ and β
- This maximum likelihood approach is considered first
  - The Bayesian treatment is considered later

Probabilistic Error Function for Regression

- Maximizing likelihood is same as minimizing sum-of-squares

$$\boxed{E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w}) - \boldsymbol{t}_n\right\}^2}$$

  - In neural network literature minimizing error is used
  - Note that we have discarded additive and multiplicative constants
- The nonlinearity of the network function $y(\boldsymbol{x}_n, \boldsymbol{w})$ causes the error function $E(\boldsymbol{w})$ to be non-convex
  - So in practice local minima is found
- Solution $\boldsymbol{w}_{\mathrm{ML}}$ is found using iterative optimization

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} + \Delta\boldsymbol{w}^{(\tau)}$$

  - Gradient descent is discussed more fully later in this lecture
    - Section on Parameter Optimization
    - The gradient is determined efficiently using back-propagation

Determining Noise Parameter β

- Having found $\boldsymbol{w}_{\mathrm{ML}}$ the value of β_{ML} can also be found by minimizing the negative log-likelihood to give

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left\{y(\boldsymbol{x}_n,\boldsymbol{w}_{ML}) - t_n\right\}^2$$

- This can be evaluated once the iterative optimization required to find $\boldsymbol{w}_{\mathrm{ML}}$ is completed
- If we have multiple target variables
  - assume that they are independent conditional on $\boldsymbol{x}$ and $\boldsymbol{w}$ with shared noise precision β, then conditional distribution on target is

$$p(\boldsymbol{t} \mid \boldsymbol{x},\boldsymbol{w}) = N(\boldsymbol{t} \mid \boldsymbol{y}(\boldsymbol{x},\boldsymbol{w}), \beta^{-1} I)$$

  - The max likelihood weight are determined by the same: $\boxed{E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\boldsymbol{y}(\boldsymbol{x}_n,\boldsymbol{w}) - \boldsymbol{t}_n\right\}^2}$
  - The noise precision is given by

$$\boxed{\frac{1}{\beta_{ML}} = \frac{1}{NK}\sum_{n=1}^{N}\left\{\boldsymbol{y}(\boldsymbol{x}_n,\boldsymbol{w}_{ML}) - \boldsymbol{t}_n\right\}^2}$$

where K is the no of target variables
Assumption of independence can be dropped with a slightly more complex optimization

Property useful for derivative computation

- There is a natural pairing between the error function (negative of the log-likelihood) and the output unit activation function
  - In linear regression with Gaussian noise, derivative wrt $\boldsymbol{w}$ of the contribution to the error function from a data point n
    - Error is y_n-t_n × ϕ_n
    - Output activation: $y_n = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}_n$ and activation function is $\mathrm{identity}$
  - Logistic regression: sigmoid activation function and cross-entropy error
  - Multiclass logistic regression: softmax activation, multiclass cross entropy
- In the regression case, output is identity, so that $y_k = a_k$
- Corresponding sum-of-squares error has the property $\boxed{\frac{\partial E}{\partial a_k} = y_k - t_k}$
  - We shall make use of this in discussing backpropagation

2. Binary Classification

- Single target variable t where t=1 denotes C_1 and t =0 denotes C_2
- Consider network with single output whose activation function is logistic sigmoid

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

  - so that $0 \leq y(\boldsymbol{x},\boldsymbol{w}) \leq 1$
- Interpret $y(\boldsymbol{x},\boldsymbol{w})$ as conditional probability $p(C_1|\boldsymbol{x})$
- Conditional distribution of targets given inputs is then a Bernoulli distribution

$$\boxed{p(t \mid \boldsymbol{x},\boldsymbol{w}) = y(\boldsymbol{x},\boldsymbol{w})^t \{1 - y(\boldsymbol{x},\boldsymbol{w})\}^{1-t}}$$

Binary Classification Error Function

- If we consider a training set of independent observations, then the error function is negative log-likelihood which in this case is a Cross-Entropy error function

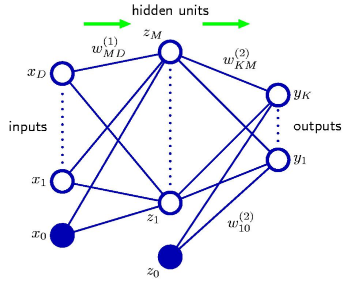
$$\boxed{E(\boldsymbol{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1-t_n)\ln(1-y_n) \right\}}$$

  - where y_n denotes $y(\boldsymbol{x}_n, \boldsymbol{w})$
- Note that there is no analog of the noise precision β since the target values are assumed to be labeled correctly
- Efficiency of cross-entropy
  - Using cross-entropy error function instead of sum of squares leads to faster training and improved generalization

2. K Separate Binary Classifications

- If we have K separate binary classifications to perform
  - Network has K outputs each with a logistic sigmoid activation function
  - Associated with each output is a binary class label $t_k \in \{0,1\},\ k=1,..,K$

$$p(\mathbf{t} \mid \boldsymbol{x},\boldsymbol{w}) = \prod_{k=1}^{K} y_k(\boldsymbol{x},\boldsymbol{w})^{t_k}\left[1-y_k(\boldsymbol{x},\boldsymbol{w})\right]^{1-t_k}$$



- Taking negative logarithm of likelihood function gives

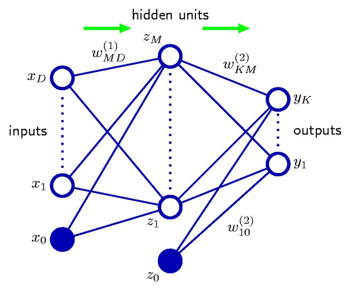
$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\left\{t_{nk}\ln y_{nk} + (1-t_{nk})\ln(1-y_{nk})\right\}$$

  - where y_{nk} denotes $y_k(\boldsymbol{x_n},\boldsymbol{w})$
- Again the derivative of the error function wrt the activation function for a particular unit takes the form

$$\boxed{\frac{\partial E}{\partial a_k} = y_k - t_k}$$

Comparison with K-class Linear Classification

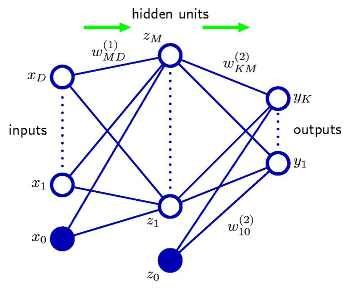
- If we are using a standard two-layer network
  - weight parameters in the first layer of the network are shared between the various outputs
- Whereas in a linear network each classification problem is solved independently
  - In multiclass linear regression we determine $\boldsymbol{w}_1,..,\boldsymbol{w}_K$
- The first layer of the network can be viewed as performing a nonlinear feature extraction and the sharing of features between the different outputs can save on computation and can also lead to improved generalization



3. Standard Multiclass Classification

- Each input assigned to one of K mutually exclusive classes
- One-hot vector coding (1-of-K coding scheme $t_k \in \{0,1\}$)
  - Network outputs are interpreted as $y_k(\boldsymbol{x},\boldsymbol{w}) = p(t_k = 1 \mid \boldsymbol{x})$
  - Leads to following error function

$$\boxed{E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y_k(\boldsymbol{x}_n,\boldsymbol{w})}$$



- Following multiclass linear models $\boxed{p(C_k \mid \boldsymbol{x}) = y_k(\boldsymbol{x},\boldsymbol{w}_k) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \quad a_k = \boldsymbol{w}_k^T \boldsymbol{x}}$
  - Output unit activation function is given by softmax

$$\boxed{y_k(\boldsymbol{x},\boldsymbol{w}) = \frac{\exp(a_k(\boldsymbol{x},\boldsymbol{w}))}{\sum_j \exp(a_j(\boldsymbol{x},\boldsymbol{w}))}}$$

    - which satisfies $0 \le y_k \le 1$ and $\Sigma_k y_k = 1$
    - Note: $y_k(\boldsymbol{x},\boldsymbol{w})$ are unchanged if some constant is added to all $a_k(\boldsymbol{x},\boldsymbol{w})$ causing error function to be constant for some directions in weight space
    - This degeneracy is removed if a regularization term is added to the error
- Derivative of error function wrt activation for a particular output takes the form $\boxed{\frac{\partial E}{\partial a_k} = y_k - t_k}$

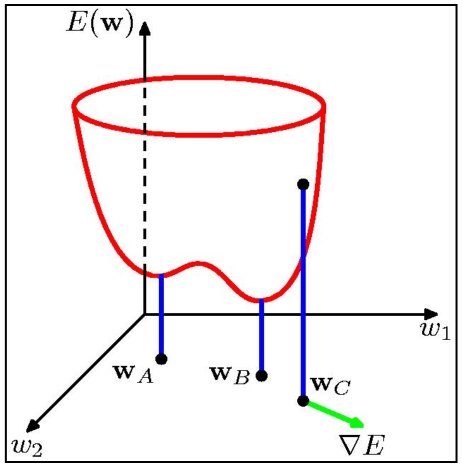
Summary of Neural Network models, activations and error functions

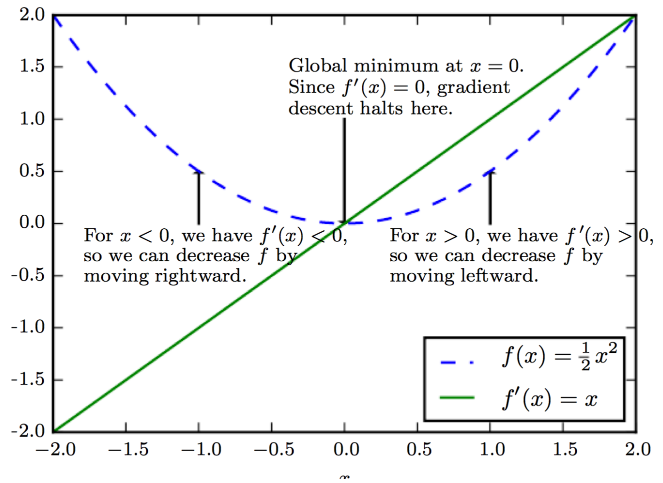
- There is a natural choice of both output unit activation function and matching error function according to the type of problem being solved

1. Regression
   - We use linear outputs and a sum-of-squares error
2. Multiple independent binary classifications:
   - We use logistic sigmoid outputs and a cross-entropy error function
3. Multiclass classification:
   - Softmax outputs with corresponding multiclass cross-entropy error
4. Classification problems involving only two classes:
   - We can use a single logistic sigmoid output, or alternatively we can use a network with two outputs having a softmax output activation function

Parameter Optimization

- Task: Find $\boldsymbol{w}$ which minimizes the chosen error function $E(\boldsymbol{w})$
- Useful to have a geometrical picture of error function
  - $E(\boldsymbol{w})$: surface sitting over weight space
    - $\boldsymbol{w}_{\mathrm{A}}$: a local minimum
    - $\boldsymbol{w}_{\mathrm{B}}$ global minimum
    - At point $\boldsymbol{w}_{\mathrm{C}}$ local gradient is given by vector $\nabla E(\boldsymbol{w})$
    - It points in direction of greatest rate of increase of $E(\boldsymbol{w})$
    - Negative gradient points to rate of greatest decrease
    - One-dimensional illustration:





Definitions of Gradient and Hessian

- First derivative of a scalar function $E(\boldsymbol{w})$ with respect to a vector $\boldsymbol{w}=[w_1,w_2]^{\mathrm{T}}$ is a vector called the *Gradient* of $E(\boldsymbol{w})$

$$\nabla E(\boldsymbol{w}) = \frac{d}{d\boldsymbol{w}}E(\boldsymbol{w}) = \begin{bmatrix} \dfrac{\partial E}{\partial w_1} \\ \dfrac{\partial E}{\partial w_2} \end{bmatrix}$$

If there are M elements in the vector then Gradient is a M x 1 vector

- Second derivative of $E(\boldsymbol{w})$ is a matrix called the *Hessian*

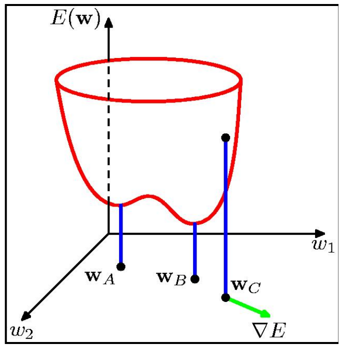
$$H = \nabla\nabla E(\boldsymbol{w}) = \frac{d^2}{d\boldsymbol{w}^2}E(\boldsymbol{w}) = \begin{bmatrix} \dfrac{\partial^2 E}{\partial w_1^2} & \dfrac{\partial^2 E}{\partial w_1 \partial w_2} \\ \dfrac{\partial^2 E}{\partial w_2 \partial w_1} & \dfrac{\partial^2 E}{\partial w_2^2} \end{bmatrix}$$

Hessian is a matrix with M^2 elements

Finding $\boldsymbol{w}$ where $E(\boldsymbol{w})$ is smallest

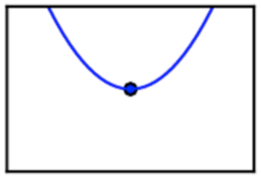
If we take a small step in weight space from $\boldsymbol{w}$ to $\boldsymbol{w}+\delta\boldsymbol{w}$ then the change in error is $\boxed{\delta E \approx \delta \boldsymbol{w}^T \nabla E(\boldsymbol{w})}$

- where the vector $\nabla E(\boldsymbol{w})$ points in the direction of greatest rate of increase of the error function

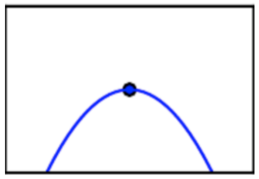


- Because $E(\boldsymbol{w})$ is a smooth continuous function of $\boldsymbol{w}$, its smallest value will occur at a point in weight space such that its gradient vanishes, so that $\boxed{\nabla E(\boldsymbol{w}) = 0}$
  - Points where gradient vanishes are called stationary points
    - Further classified into minima, maxima, and saddle points

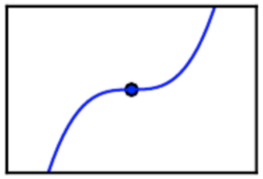
Minimum



Maximum



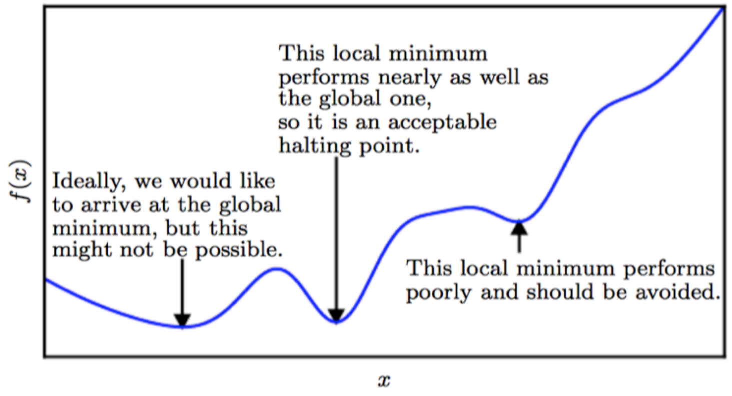
Saddle point



- One goal is to find a $\boldsymbol{w}$ such that $E(\boldsymbol{w})$ is smallest
  - But the error function typically has a highly nonlinear dependence on the weights and bias parameters and there will be many points where the gradient vanishes
    - For any point $\boldsymbol{w}$ that is a local minimum there will be other equivalent minima
    - For a two layer network with M hidden units, each point in weight space is a member of a family of $M!2^M$ equivalent points

Complexity of Surface

- There will be typically:
  - Multiple inequivalent stationary points and
  - Multiple inequivalent minima
- Global minimum
  - Smallest value of E for any $\boldsymbol{w}$
  - Minima with higher E are local minima
- May not need global minimum
  - May not know if global minimum found
  - May need to compare several local minima
- Because there is no hope of finding analytical solution
  - to equation $\nabla E(\boldsymbol{w}) = 0$ we resort to numerical procedures



Iterative Numerical Procedures for Minima

- Most algorithms choose some initial $\boldsymbol{w}^{(0)}$ for the weight vector
- Move through weight space in a succession of steps of the form

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} + \Delta \boldsymbol{w}^{(\tau)}$$

where τ labels the iteration step

- Different algorithms involve different choices for update $\Delta \boldsymbol{w}^{(\tau)}$

- Many algorithms use gradient information
  - Therefore require that after each update:
  - the value of $\nabla E(\boldsymbol{w})$ is evaluated at the new weight vector $\boldsymbol{w}^{(\tau+1)}$

- To understand importance of gradient information
  - It is useful to consider Taylor's series expansion of error function
    - Leads to local quadratic approximation

Overview of Optimization methods

1 Local quadratic approximation

- We learn that evaluating E around minimum is $O(W^2)$
  - where W is dimensionality of $\boldsymbol{w}$
  - And the task of finding the minimum is $O(W^3)$

2 Use of gradient information

- Complexity of locating minimum is reduced using gradient
  - Backpropagation method of evaluating gradient is $O(W)$
  - And the task of finding the minimum is $O(W^2)$

3 Gradient descent optimization

- Overview of first and second order (Newton) methods
- Batch vs On-line (minibatch, SGD)

1. Local Quadratic Optimization

- Taylor's Series
  - Expansion of $f(x)$ at a: $\boxed{f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f^{(3)}(a)}{3!}(x-a)^3+\cdots.}$
  - Expansion of $E(\boldsymbol{w})$ at $\hat{\mathrm{w}}$ (with cubic and higher terms omitted):

$$\boxed{E(\boldsymbol{w}) \cong E(\hat{\mathrm{w}})+(\boldsymbol{w}-\hat{\mathrm{w}})^T\mathrm{b}+\frac{1}{2}(\boldsymbol{w}-\hat{\mathrm{w}})^T H(\boldsymbol{w}-\hat{\mathrm{w}})}$$

  - Where $\mathrm{b} \equiv \nabla E\,|_{\mathrm{w}=\hat{\mathrm{w}}}$ is the gradient of E evaluated at $\hat{\mathrm{w}}$. It has W elements.
  - H is the Hessian matrix $\mathbf{H}=\nabla\nabla\mathbf{E}$. H is a W x W matrix with elements:

$$(\mathrm{H})_{ij}=\left.\frac{\partial E}{\partial w_i\,\partial w_j}\right|_{\mathrm{w}=\hat{\mathrm{w}}}$$

- Corresponding local approximation to the gradient
  - From equation for $E(\boldsymbol{w})$ above is given by

$$\nabla E \simeq \boldsymbol{b}+H(\boldsymbol{w}-\hat{\boldsymbol{w}})$$

  - For points $\boldsymbol{w}$ that at sufficiently close to $\boldsymbol{w}$^ these expressions will give reasonable approximations for the error and its gradient

1. Local Quadratic Optimization

- Consider local quadratic approximation around $\boldsymbol{w}^*$, that is a minimum of the error function
- In this case there is no linear term because $\nabla E(\boldsymbol{w}^*) = 0$

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

  - where H is evaluated at $\boldsymbol{w}^*$ and the linear term vanishes
- Let us interpret this geometrically
  - Consider eigen value equation for the Hessian matrix

$$\mathrm{Hu}_i = \lambda_i \mathrm{u}_i$$

    - where the eigen vectors u_i form a complete orthonormal set, so that:

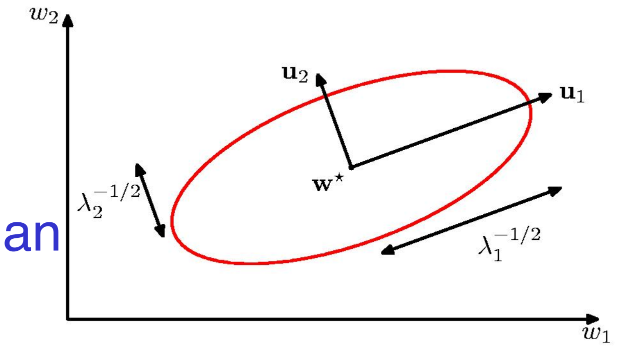
$$u_i^T u_i = \delta_{ij}$$

  - Expand $(\boldsymbol{w}$-$\boldsymbol{w}^*)$ as a linear combination the eigenvectors

$$\boldsymbol{w}-\boldsymbol{w}^* = \sum_i \alpha_i \mathrm{u}_i$$

  - This can be regarded as a transformation of the coordinate system in which the origin is translated to the point $\boldsymbol{w}^*$ and the axes are rotated to align with the eigenvectors
    - Through the orthogonal matrix whose columns are the u_i

Neighborhood of a minimum $\boldsymbol{w}^*$ is quadratic



- $\boldsymbol{w}$-$\boldsymbol{w}^*$ is a coordinate transformation
    - Origin is translated to $\boldsymbol{w}^*$
    - Axes rotated to align with eigenvectors of Hessian
- Error function can be written as

$$E(\boldsymbol{w}) = E(\boldsymbol{w}^*) + \frac{1}{2}\sum_i \lambda_i \alpha_i^2$$

- Matrix $\mathbf{H} = \nabla\nabla \mathbf{E}$ is positive definite iff

$$\mathrm{v}^T H \mathrm{v} > 0 \text{ for all } \mathrm{v}$$

- Since eigenvectors form a complete set then an arbitrary vector v can be written as $\mathrm{v} = \sum_i c_i \mathrm{u}_i$

$$\mathrm{v}^T H \mathrm{v} = \sum_i c_i^2 \lambda_i$$

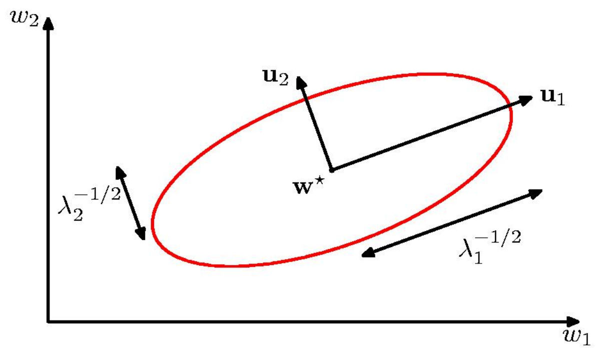
    - And so H will be positive definite (or all its eigenvalues are positive).
- In the new coordinate system, whose basis vectors are given by the eigenvectors $\{\mathrm{u_i}\}$, the contours of constant E are ellipses centered on the origin

Error Function Approximation by a Quadratic

In the neighborhood of a minimum $\boldsymbol{w}^*$,

the error function can be approximated by quadratic

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



Contours of constant error are then ellipses

- Whose axes are aligned with the eigen vectors $\boldsymbol{u}_i$ of the Hessian H with
- lengths that are inversely proportional to square roots of eigenvectors λ_i

Condition for a point $\boldsymbol{w}^*$ to be a minimum

- For a one-dimensional weight space, a stationary point $\boldsymbol{w}^*$ will be minimum if

$$\left.\frac{\partial^2 E}{\partial w^2}\right|_{w^*} > 0$$

- Corresponding result in D dimensions is that the Hessian matrix evaluated at $\boldsymbol{w}^*$ is positive definite
  - A matrix H is *positive definite* iff $\mathrm{v}^{\mathrm{T}}\mathrm{H}\mathrm{v} > 0$ for all v

Complexity of Evaluating Error Minimum

- The quadratic approximation to the error function is

$$E(\mathrm{w}) \cong E(\hat{\mathrm{w}}) + (\mathrm{w}-\hat{\mathrm{w}})^T \mathrm{b} + \frac{1}{2}(\mathrm{w}-\hat{\mathrm{w}})^T H(\mathrm{w}-\hat{\mathrm{w}})$$

  where b is the gradient of E evaluated at $\hat{\mathbf{w}}$ or $\mathrm{b} \equiv \nabla E |_{\mathrm{w}=\hat{\mathrm{w}}}$
    - b is a vector of W elements
  - H is the Hessian matrix $\mathbf{H} = \nabla\nabla\mathbf{E}$ with W x W elements
- Error surface is specified by b and H
  - They contain total of $W(W+3)/2$ independent elements
    - W is total number of adaptive parameters in network
- If we did not use gradient information to determine minimum but simply evaluated the error function for different settings of $\boldsymbol{w}$
  - Need to obtain $O(W^2)$ pieces of information, each requiring $O(W)$ steps.
  - Computational effort needed is $O(W^3)$
  - In a 10 x 10 x 10 network $W=100+100=200$ weights which means 8 $\mathrm{million}$ steps

2. Use of Gradient Information

- Without using gradient information
  - Computational effort needed to find minimum is $O(W^3)$
- Gradient of error function can be evaluated efficiently using back-propagation
  - By using error backpropagation minimum can be found in $O(W^2)$ steps
    - $4{,}000$ steps for 10 x 10 x 10 network with 200 weights

3. Gradient Descent Optimization

- The simplest approach to using gradient information:
  - Take a small step in the direction of the negative gradient, so that

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E(\boldsymbol{w}^{(\tau)})$$

  - Where $\eta > 0$ is known as the learning rate
  - After each update, the gradient is re-evaluated for the new weight vector and the process is repeated
  - Note that the error function is defined wrt a training set
  - So each step requires the entire training set to be processed in order to evaluate $\nabla E(\boldsymbol{w})$
  - Techniques that use the whole data set are called batch methods
    - At each step the weight vector is moved in the direction of the greatest rate of decrease of the error function
    - So this method is known as gradient descent or steepest descent
    - But this algorithm turns out to be a poor choice

More efficient gradient descent optimization

- Conjugate gradients and quasi Newton methods
  - They are much more robust and much faster than simple gradient descent
  - Unlike gradient descent the error function always decrease at each iteration unless the minimum has been reached
- To obtain a sufficiently good minimum it may be necessary to run a gradient based algorithm multiple times
  - Each time using a different randomly chosen starting point and comparing results on a validation set

Stochastic Gradient Descent (SGD)

- On-line version of gradient descent useful for large data sets
- Error functions based on maximum likelihood comprise a sum of terms, one for each data point

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

- SGD makes an update to the weight vector based on one data point at a point, so that

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E_n(\boldsymbol{w}^{(\tau)})$$

- This update is repeated by cycling through the data either in sequence or by selecting points at random with replacement
- Updates can also be based on minibatches of data points

Advantage of SGD over Batch methods

- Redundancy in the data is handled more efficiently
  - Consider an extreme example in which we take a data set and double its size by duplicating every data point
  - This simply multiplies the error function by a factor of 2 and so is equivalent to using the original error function
  - Batch methods will require double the computational effort to evaluate the batch error function
  - Whereas SGD will be unaffected
    - Because the same data point will not cause an update of $\boldsymbol{w}$?
- SGD escapes from local minima
  - Since a stationary point wrt E for the whole data set will not be a stationary point for each data point individually

Summary of Network Training

- Neural network activation functions are related to error functions
- Parameter optimization can be viewed as minimizing error function in weight space
- At the minimum Hessian is positive definite
- Evaluating minimum without using gradient is $O(W^3)$
- Using gradient information is efficient $O(W^2)$