Neural Network Training

Sargur Srihari

Topics

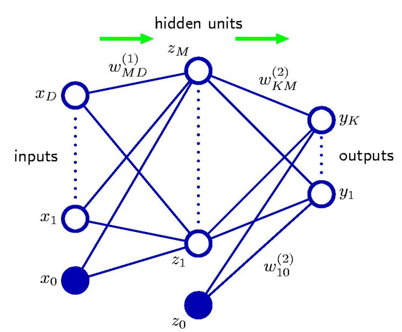
- Neural network parameters
- Probabilistic problem formulation
- Determining the error function
  - Regression
  - Binary classification
  - Multi-class classification
- Parameter optimization
- Local quadratic approximation
- Use of gradient optimization
- Gradient descent optimization

Neural Network parameters

- Linear models for regression and classification can be represented as

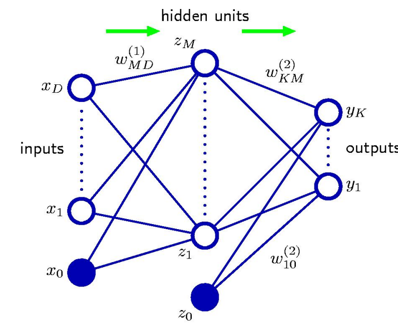
$$y(\boldsymbol{x}, \mathrm{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x})\right)$$

  - 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



Network Training: Sum of squared errors

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



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

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

N training vectors

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

- Consider a more general probabilistic interpretation

Probabilistic View: From activation function f determine Error Function E (as defined by likelihood function)

1. Regression
   - f: activation function is identity $\quad y(\boldsymbol{x},\boldsymbol{w}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) = \sum_{j=1}^{M} w_j\phi_j(\boldsymbol{x})$
   - E: Sum-of-squares error/Maximum Likelihood $\quad E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} \| \boldsymbol{y}(\boldsymbol{x}_n,\boldsymbol{w}) - \boldsymbol{t}_n \|^2$
2. (Multiple Independent) Binary Classifications
   - f: activation function is Logistic sigmoid $\quad y(\boldsymbol{x},\boldsymbol{w}) = \sigma(\boldsymbol{w}^T\phi(\boldsymbol{x})) = \frac{1}{1+\exp(-\boldsymbol{w}^T\phi(\boldsymbol{x}))}$
   - E: Cross-entropy error function $\quad E(\boldsymbol{w}) = -\sum_{n=1}^{N}\left\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\right\}$
3. Multiclass Classification
   - f: Softmax outputs $\quad y_k(\boldsymbol{x},\boldsymbol{w}) = \frac{\exp(\boldsymbol{w}_k\phi(\boldsymbol{x}))}{\sum_j \exp(\boldsymbol{w}_j\phi(\boldsymbol{x}))}$
   - E: Cross-entropy error function

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

1. Probabilistic View: Regression

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

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

- Likelihood function

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

- Taking negative logarithm, we get the negative log-likelihood

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

  - which can be used to learn parameters w and β

Regression Error Function

- Likelihood Function could be used to learn parameters $\boldsymbol{w}$ and $\boldsymbol{\beta}$
  - Usually done in a Bayesian treatment
- In neural network literature minimizing error is used
  - They are equivalent here. Sum of squared errors is

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

  - Its smallest value occurs when $\nabla E(\boldsymbol{w})=0$
- Since $E(\mathrm{w})$ is non-convex:
  - Solution $\boldsymbol{w}_{\mathrm{ML}}$ found using iterative optimization $\boldsymbol{w}^{t+1}=\boldsymbol{w}^{t}+\underline{\Delta}\,\boldsymbol{w}^{t}$
    - Gradient descent (discussed later in this lecture))
  - Another solution is back-propagation
    - Since regression output is same as activation $y_k=a_k$, so

$$\frac{\partial E}{\partial a_k} = y_k - t_k \qquad a_k = \sum_{i=1}^{M} w_{ki}^{(2)} x_i + w_{k0}^{(2)} \quad \text{where } k = 1,..,K$$

- Having found $\boldsymbol{w}_{\mathrm{ML}}$ the value of $\boldsymbol{\beta}_{\mathrm{ML}}$ can also be found using

$$\frac{\ln\beta/2\pi}{\beta} = \frac{1}{N}\sum_{n=1}^{N}\left\{y(\mathrm{x}_n,\mathrm{w}_{ML}) - t_n\right\}^2$$

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

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

Binary Classification Error Function

- Error function is negative log-likelihood which in this case is a Cross-Entropy error function

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

  - where y_n denotes $y(\boldsymbol{x}_n, \boldsymbol{w})$
- Using cross-entropy error function instead of sum of squares leads to faster training and improved generalization

2. K Separate Binary Classifications

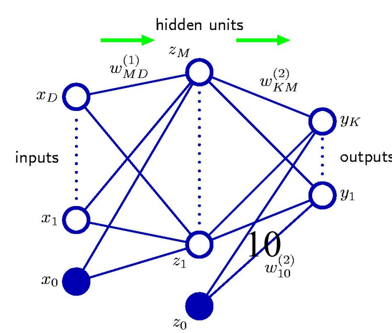
- Network has K outputs each with a logistic sigmoid activation function
- Associated with each output is a binary class label t_k

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

- Taking negative logarithm of likelihood function

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

where y_{nk} denotes $y_k(\boldsymbol{x}_n, \boldsymbol{w})$

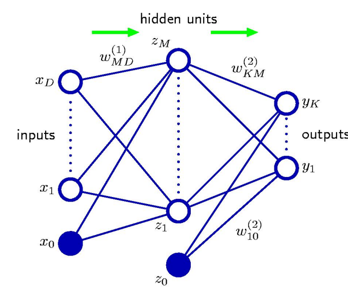


3. Multiclass Classification

- Each input assigned to one of K classes
- Binary target variables have 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



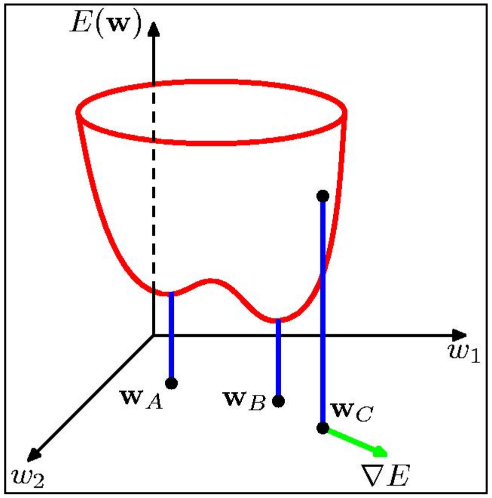
$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y_k(\boldsymbol{x}_n,\boldsymbol{w})$$

- Output unit activation function is given by softmax

$$y_k(\boldsymbol{x},\boldsymbol{w}) = \frac{\exp(a_k(\boldsymbol{x},\boldsymbol{w}))}{\sum_j \exp(a_j(\boldsymbol{x},\boldsymbol{w}))} \qquad y_k(\boldsymbol{x},\boldsymbol{w}) = \frac{\exp(\boldsymbol{w}_k \phi(\boldsymbol{x}))}{\sum_j \exp(\boldsymbol{w}_j \phi(\boldsymbol{x}))}$$

Parameter Optimization

- Task: Find weight vector $\boldsymbol{w}$ which minimizes the chosen function $E(\boldsymbol{w})$
- Geometrical picture of error function
- Error function has a highly nonlinear
- dependence

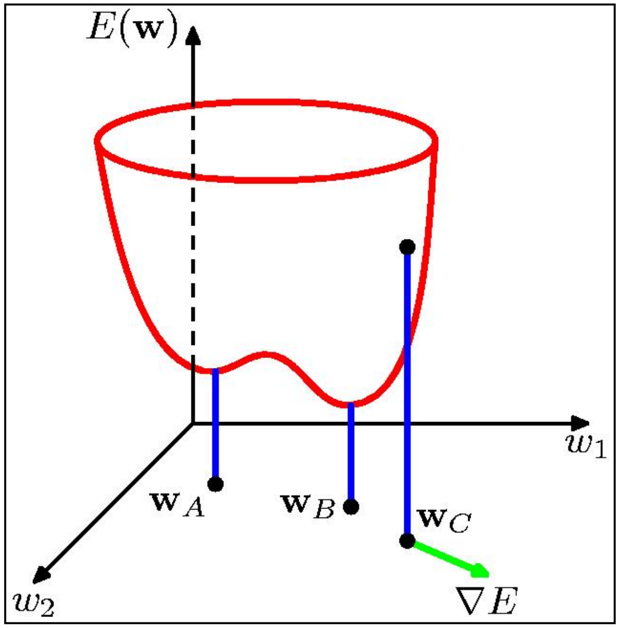


Parameter Optimization: Geometrical View

$E(\boldsymbol{w})$: surface sitting over weight space

- $\boldsymbol{w}_{\mathrm{A}}$: a local minimum
- $\boldsymbol{w}_{\mathrm{B}}$ global minimum
- Need to find minimum

- At point $\boldsymbol{w}_{\mathrm{C}}$ local gradient
  - is given by vector $\nabla E(\boldsymbol{w})$
  - points in direction of greatest rate of increase of $E(\boldsymbol{w})$
  - Negative gradient points to rate of greatest decrease

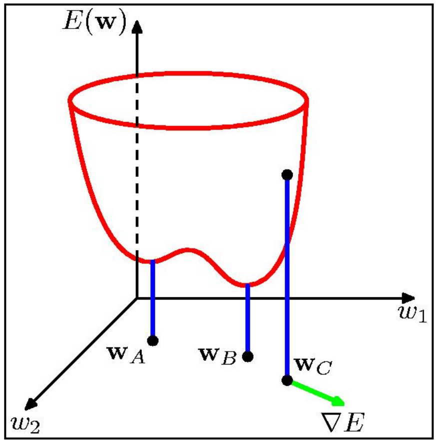


Finding w where E(w) is smallest

Small step from w to w+δw leads to

- change in error function

$$\delta E \approx \delta \mathrm{w}^T \nabla E(\mathrm{w})$$



- Minimum of E(w) will occur when

$$\nabla E(\mathrm{w}) = 0$$

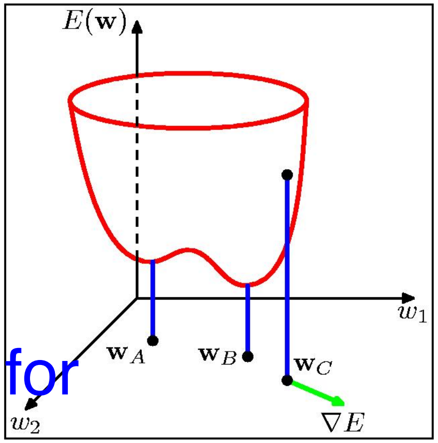
- Points at which gradient vanishes are stationary points: minima, maxima, saddle

Complex surface

No hope of finding analytical solution to equation

$$\nabla E(\mathrm{w}) = 0$$

Iterative Numerical Procedure for Minima



- Since there is no analytical solution choose initial $\mathrm{w}^{(0)}$ and update it using

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

  where τ is the iteration step
- Different algorithms involve different choices for weight vector update $\Delta w^{(\tau)}$

- Weight vector update $\Delta w^{(\tau)}$ is usually based on gradient $\nabla E(\mathrm{w})$ evaluated at weight vector $\mathrm{w}^{(\tau+1)}$
- To understand importance of gradient information consider Taylor's series expansion of error function
  - Leads to local quadratic approximation

Discussion Overview

Preliminary concepts for Backpropagation

1 Local quadratic approximation

- Provides insight into optimization problem
- $O(W^3)$ where W is dimensionality of w
- Based on Taylor's series: $f(x)$ is approximated

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots.$$

2 Use of gradient information

- Leads to significant improvements in speed of locating minima of error function
- Backpropagation is $O(W^2)$

3 Gradient descent optimization

- Simplest approach of using gradient information

Definitions of Gradient and Hessian

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

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

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

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

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

Hessian is a matrix with M^2 elements

1. Local Quadratic Optimization

- Taylor's Series Expansion of $E(\mathrm{w})$ around some point $\hat{\mathrm{w}}$ in weight space (with cubic and higher terms omitted)

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

  where b is the gradient of E evaluated at $\hat{\mathrm{w}}$ $\quad \mathrm{b} \equiv \nabla E \,|_{\mathrm{w}=\hat{\mathrm{w}}}$
  - b is a vector of W elements

  H is the Hessian matrix $\mathbf{H} = \nabla\nabla \mathrm{E}$ with elements $(\mathrm{H})_{ij} = \left.\frac{\partial E}{\partial w_i \partial w_j}\right|_{\mathrm{w}=\hat{\mathrm{w}}}$
  - H is a W x W matrix

- Consider local quadratic approximation around w^*, a minimum of the error function

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

  - where H is evaluated at 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 $\mathrm{u_i}$ are orthonormal $u_i^T u_i = \delta_{ij}$
  - Expand $(\mathrm{w}\text{-}\mathrm{w}^*)$ as a linear combination the eigenvectors $\mathrm{w} - \mathrm{w}^* = \sum_i \alpha_i \mathrm{u}_i$

Error Function Approximation by a Quadratic

- Error Functions
  - Linear Regression $y(\mathrm{x},\mathrm{w}) = \mathrm{w}^T \phi(\mathrm{x}) = \sum_{j=1}^{M} w_j \phi_j(\mathrm{x})$

    $$E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(\mathrm{x}_n,\mathrm{w}) - t_n\}^2$$

  - Binary Classification $y(\mathrm{x},\mathrm{w}) = \sigma(\mathrm{w}^T\phi(\mathrm{x})) = \frac{1}{1+\exp(-\mathrm{w}^T\phi(\mathrm{x}))}$

    $$E(\mathrm{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

  - Multiclass Classification $y_k(\mathrm{x},\mathrm{w}) = \frac{\exp(\mathrm{w}_k\phi(\mathrm{x}))}{\sum_j \exp(\mathrm{w}_j\phi(\mathrm{x}))}$

    $$E(\mathrm{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y_k(x_n,\mathrm{w})$$

Approximated by quadratic

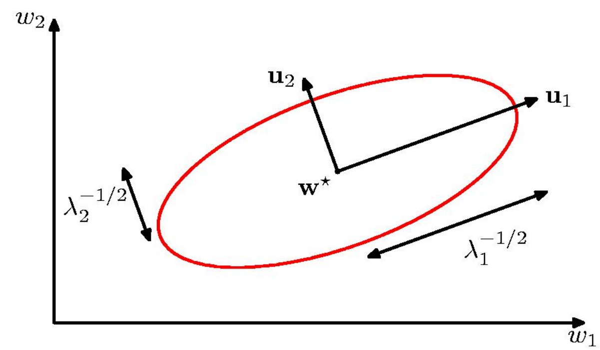
$$E(\mathrm{w}) \cong E(\mathrm{w}^*) + \frac{1}{2}(\mathrm{w}-\mathrm{w}^*)^T H(\mathrm{w}-\mathrm{w}^*)$$

Where $\mathbf{H} = \nabla\nabla \mathrm{E}$ is a W x W matrix

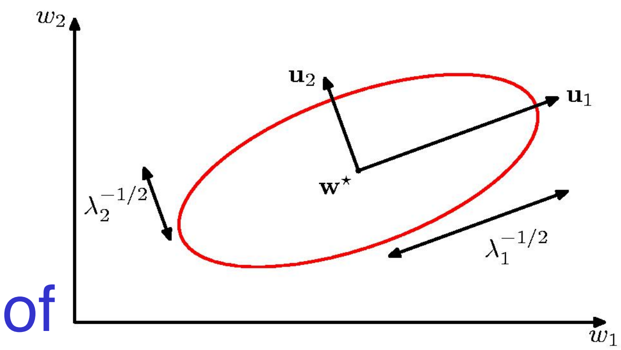
Whose contours of constant error are ellipses

with axes aligned with eigen vectors u_i of H whose

lengths are inversely proportional to sq roots of eigenvalues



Neighborhood of a minimum w*



- w-w* is a coordinate transformation
  - Origin is translated to w*
  - Axes rotated to align with eigenvectors of Hessian
- Error function can be written as

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

- 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 $\mathrm{v} = \sum_i c_i \mathrm{u}_i$
- Then an arbitrary vector v can be written as $\mathrm{v}^T H \mathrm{v} = \sum_i c_i^2 \lambda_i$
- The stationary point w* will be a minimum if the Hessian matrix is positive definite (or all its eigenvalues are positive)

Condition for a point w* to be a minimum

- For a one-dimensional weight space, a stationary point 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 w* is positive definite
  - A matrix H is positive definite iff $\mathrm{v}^T\mathrm{H}\mathrm{v} > 0$ for all v

Complexity of Quadratic Approximation

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

where b is the gradient of E evaluated at $\hat{\mathrm{w}}$

- b is a vector of W elements $\mathrm{b} \equiv \nabla E \,|_{\mathrm{w}=\hat{\mathrm{w}}}$

H is the Hessian matrix with elements

- H is a W x W matrix $\mathbf{H} = \nabla\nabla \mathbf{E}$

- 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
- Minimum depends on $O(W^2)$ parameters
- Need to perform $O(W^2)$ function evaluations, each requiring $O(W)$ steps.
- Computational effort needed is $O(W^3)$
- W in a 10 x 10 x 10 network needs 100+100=200 weights which means 8 million steps

2. Use of Gradient Information

- Gradient of error function can be evaluated efficiently using back-propagation
    - Use of gradient information can lead to significant improvements in speed with which minimum of error function can be located
- In quadratic approximation to error function
    - Computational effort needed is $O(W^3)$
- By using gradient information minimum can be found in $O(W^2)$ steps
    - 4,000 steps for 10 x 10 x 10 network with 200 weights

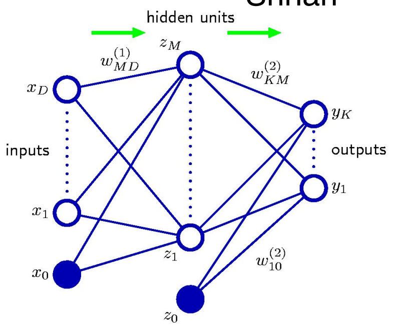
3. Gradient Descent Optimization

- Simplest approach to using gradient information
- Take a small step in the direction of the negative gradient

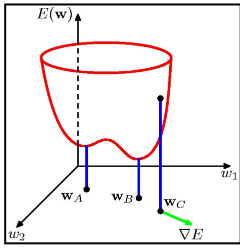
$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla E(\mathrm{w}^{(\tau)})$$

- η is the learning rate
- There are batch and on-line versions

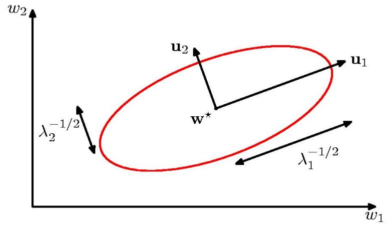
Summary



- Neural network parameters have many parameters
  - can be determined analogous to linear regression parameters
- Probabilistic formulation leads to appropriate error functions for linear regression, binary and multi-class classification



- Parameter optimization can be viewed as minimizing error function in weight space
- At the minimum Hessian is positive definite



- Local quadratic optimization needs $O(W^3)$ steps
- Using gradient information more efficient $O(W^2)$ algorithm can be designed