COURSE 02457

Signal Processing in Non-linear Systems:

Lecture 3

- Learning problem
- The likelihood function and Bayesian learning
- Maximum likelihood estimation of parameters in the normal distribution
- Learning by least squares
- Linear models
- Neural network models
- Neural classifiers
- Probability density function estimation
- Classifiers from density estimators
- Markov processes

The learning problem

- Supervised learning: Learning relations between sets of variables e.g. between input and output variables, conditional distributions $p(\text{output}|\text{input})$.
- Unsupervised learning: Learning the distribution of a set of variables $p(\text{input})$.

The Bayesian paradigm

- The density of the measured signals is modelled by a parameterized density: $p(x) \sim p(x|\theta)$.
- Let $\chi = \{x_1, x_2, x_3, ..., x_N\}$ be a *training set*
- Objective: Find the distribution of the parameter vector, $p(\theta|\chi)$, hence the parameters are considered stochastic.

The likelihood function

- Let $\chi = \{x_1, x_2, x_3, ..., x_N\}$ be a *training set*
- We use Bayes theorem

$$p(\theta|\chi) = \frac{p(\chi|\theta)p(\theta)}{p(\chi)}$$

- The function $p(\chi|\theta)$ is called the likelihood function (more correct the likelihood of the parameter vector θ). The density $p(\theta)$ is called the *a priori* or *prior* parameter distribution.
- If the prior is "flat" in the neighborhood of the peak of $p(\chi|\theta)$, we have

$$p(\theta|\chi) \propto p(\chi|\theta)$$

- ...and finding the most probable parameters is equivalent to finding the maximum likelihood parameters.

Maximum likelihood & optimization

- For independent examples, $\chi = \{x_1, x_2, x_3, ..., x_N\}$, the likelihood function factorise

$$p(\chi|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$

- Many algorithms are based on minimizing an index or costfunction

$$E(\theta) = -\log p(\chi|\theta) = \sum_{n=1}^{N} -\log p(x_n|\theta)$$

1D normal distribution

- Let the parameterized density be a 1D normal distribution

$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

- For independent examples, $\chi = \{x_1, x_2, x_3, ..., x_N\}$, the likelihood function becomes

$$\begin{aligned} p(\chi|\mu, \sigma^2) &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right) \\ E(\mu, \sigma^2) &= \frac{N}{2}\log 2\pi\sigma^2 + \frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 \end{aligned}$$

1D normal distribution

- Costfunction for maximum likelihood estimation of mean and variance

$$E(\mu, \sigma^2) = \frac{N}{2} \log 2\pi\sigma^2 + \frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2$$

- Derivatives are zero as minimum:

$$\begin{aligned}
\frac{\partial E(\mu, \sigma^2)}{\partial \mu} &= \frac{1}{\sigma^2} \sum_{n=1}^{N} -(x_n - \mu) \\
\frac{\partial E(\mu, \sigma^2)}{\partial \sigma^2} &= \frac{N}{2} \frac{1}{\sigma^2} - \frac{1}{(\sigma^2)^2} \sum_{n=1}^{N} (x_n - \mu)^2
\end{aligned}$$

$$\begin{aligned}
0 &= \frac{1}{\widehat{\sigma^2}} \sum_{n=1}^{N} -(x_n - \widehat{\mu}) \\
0 &= \frac{N}{2} \frac{1}{\widehat{\sigma^2}} - \frac{1}{(\widehat{\sigma^2})^2} \sum_{n=1}^{N} (x_n - \widehat{\mu})^2
\end{aligned}$$

$$\begin{aligned}
\widehat{\mu} &= \frac{1}{N} \sum_{n=1}^{N} x_n \\
\widehat{\sigma^2} &= \frac{1}{N} \sum_{n=1}^{N} (x_n - \widehat{\mu})^2
\end{aligned}$$

Multivariate normal distribution

- For independent examples, $\chi = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_N\}$, the likelihood function becomes

$$
\begin{aligned}
p(\chi|\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \left(\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}}\right)^N \exp\left(-\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})\right) \\
E(\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \frac{N}{2}\log|2\pi\boldsymbol{\Sigma}| + \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu})
\end{aligned}
$$

- We need two rules:

$$
\begin{aligned}
\frac{\partial \log|\mathbf{A}|}{\partial u} &= \mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial u} \\
\frac{\partial \mathbf{A}^{-1}}{\partial u} &= -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial u}\mathbf{A}^{-1}
\end{aligned}
$$

Multivariate normal distribution

$$
\begin{aligned}
\frac{\partial E(\boldsymbol{\mu}, \sigma^2)}{\partial \boldsymbol{\mu}} &= \sum_{n=1}^{N} -(\mathbf{x}_n - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1} \\
\frac{\partial E(\boldsymbol{\mu}, \sigma^2)}{\partial \boldsymbol{\Sigma}} &= \frac{N}{2}\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\left(\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})'\right)\boldsymbol{\Sigma}^{-1}
\end{aligned}
$$

$$
\begin{aligned}
0 &= \sum_{n=1}^{N} -(\mathbf{x}_n - \widehat{\boldsymbol{\mu}})\widehat{\boldsymbol{\Sigma}}^{-1} \\
0 &= \frac{N}{2}\widehat{\boldsymbol{\Sigma}}^{-1} - \widehat{\boldsymbol{\Sigma}}^{-1}\sum_{n=1}^{N}\left((\mathbf{x}_n - \widehat{\boldsymbol{\mu}})(\mathbf{x}_n - \widehat{\boldsymbol{\mu}})'\right)\widehat{\boldsymbol{\Sigma}}^{-1}
\end{aligned}
$$

$$
\begin{aligned}
\widehat{\boldsymbol{\mu}} &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \\
\widehat{\boldsymbol{\Sigma}} &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \widehat{\boldsymbol{\mu}})(\mathbf{x}_n - \widehat{\boldsymbol{\mu}})'
\end{aligned}
$$

Least squares as maximum likelihood

- Let $(\chi_{\mathbf{x}}, \chi_y) = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), ..., (\mathbf{x}_N, y_N)\}$,
- We seek a conditional density model of the form

$$
\begin{aligned}
y &= f_\theta(\mathbf{x}) + \nu \\
p(y|\mathbf{x}, \sigma^2, \theta) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(y - f_\theta(\mathbf{x}))^2\right) \\
p(\chi_y|\chi_{\mathbf{x}}, \sigma^2, \theta) &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - f_\theta(\mathbf{x}_n)^2\right) \\
E(\theta, \sigma^2) &= \frac{N}{2}\log 2\pi\sigma^2 + \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - f_\theta(\mathbf{x}_n))^2
\end{aligned}
$$

- Hence, maximizing the likelihood for Gaussian noise leads to a least squares problem (for θ).

Least squares techniques

- Let a training set be given by $\mathcal{D} = \{(t^1, \mathbf{x}^1), ..., (t^N, \mathbf{x}^N)\}$, the sum-of-squares approximation error is given by

$$E = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}^n + w_0 - t^n)^2 \tag{1}$$

- The optimal parameters are found by gradient based minimization,

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{w}} &= \sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}^n + w_0 - t^n)\mathbf{x}^n \\
\frac{\partial E}{\partial w_0} &= \sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}^n + w_0 - t^n)
\end{aligned}$$

Least squares techniques cont'd

- equations to solve

$$\begin{aligned}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}^n + w_0 - t^n)\mathbf{x}^n &= 0\\ \sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}^n + w_0 - t^n) &= 0\end{aligned}$$

- the solution is given by in terms of $\boldsymbol{\mu} = (1/N)\sum \mathbf{x}^n$, and $\tau = (1/N)\sum t^n$

$$\begin{aligned}\mathbf{w} &= -\left(\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^n - \boldsymbol{\mu})(\mathbf{x}^n - \boldsymbol{\mu})^T\right)^{-1}\left(\frac{1}{N}\sum_{n=1}^{N}(t^n - \tau)\mathbf{x}^n\right)\\ w_0 &= -\mathbf{w}^T\boldsymbol{\mu} + \tau\end{aligned}$$

- This can be used to model any linear input-output relation

The generalization error

- 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

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

- Now consider the limit of large sets, the error per example is

$$\begin{aligned} E &= \lim_{N \to \infty} \frac{1}{2N} \sum_{n=1}^{N} (y(\mathbf{x}^n; \mathbf{w}) - t^n)^2 \\ &= \frac{1}{2} \int \int (y(\mathbf{x}; \mathbf{w}) - t)^2 p(t, \mathbf{x}) dt d\mathbf{x} \end{aligned}$$

- This is the average (or expected) error on a test datum $(\mathbf{x}, t)$, which we call the generalization error.

The generalization error contd

- The generalization error

$$E = \frac{1}{2} \int \int (y(\mathbf{x};\mathbf{w}) - t)^2 p(t|\mathbf{x}) p(\mathbf{x}) dt d\mathbf{x}$$

- can be rewritten using the definitions

$$\begin{aligned} < t|\mathbf{x} > \; &= \int t p(t|\mathbf{x}) dt \\ < t^2|\mathbf{x} > \; &= \int t^2 p(t|\mathbf{x}) dt \end{aligned}$$

$$\begin{aligned} \{y - t\}^2 \; &= \; \{y - < t|\mathbf{x} > + < t|\mathbf{x} > - t\}^2 \\ &= \; \{y - < t|\mathbf{x} >\}^2 + 2\{y - < t|\mathbf{x} >\}\{< t|\mathbf{x} > - t\} \\ &+ \; \{< t|\mathbf{x} > - t\}^2 \end{aligned}$$

Regressions

- Then the generalization error becomes

$$\begin{aligned} E &= \frac{1}{2}\int\int (y(\mathbf{x};\mathbf{w}) - t)^2 p(t|\mathbf{x})p(\mathbf{x})dtd\mathbf{x} \\ &= \frac{1}{2}\int\int \{y - < t|\mathbf{x} >\}^2 + 2\{y - < t|\mathbf{x} >\}\{< t|\mathbf{x} > -t\} \\ &+ \{< t|\mathbf{x} > -t\}^2 p(t|\mathbf{x})p(\mathbf{x})dtd\mathbf{x} \end{aligned}$$

- leading to the simplification

$$\begin{aligned} E &= \frac{1}{2}\int (y(\mathbf{x};\mathbf{w}) - < t|\mathbf{x} >)^2 p(\mathbf{x})d\mathbf{x} \\ &+ \frac{1}{2}\int \{< t^2|\mathbf{x} > - < t|\mathbf{x} >^2\} p(\mathbf{x})d\mathbf{x} \end{aligned}$$

- we see that the generalization error is minimal (as function of $y(\mathbf{x};\mathbf{w})$) if

$$y(\mathbf{x};\mathbf{w}) = < t|\mathbf{x} >$$

- The model should output the conditional mean, hence be a "regression"

The bias-variance trade-off

- When training a model we have only a finite training set, hence the model can only find the best approximation minimization the training error

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

- leading to $\mathbf{w}_{\text{optimal}} = \mathbf{w}(\mathcal{D})$, hence the error in a particular test point

$$(y(\mathbf{x};\mathbf{w}(\mathcal{D})) - t)^2$$

will depend on the particular training set.

- The expected value of this quantity is,

$$\mathcal{E}_{\mathcal{D}}[(y(\mathbf{x};\mathbf{w}(\mathcal{D})) - t)^2]$$

The bias-variance trade-off cont'd

- The expectation

$$\mathcal{E}_{\mathcal{D}}\left[\left(y(\mathbf{x};\mathbf{w}(\mathcal{D})) - < t|\mathbf{x} >\right)^2\right]$$

can be rewritten

$$\begin{aligned}
(y(\mathbf{x};\mathbf{w}(\mathcal{D})) &- < t|\mathbf{x} >)^2 = \\
& \{y(\mathbf{x}) - \mathcal{E}_{\mathcal{D}}[y(\mathbf{x})] + \mathcal{E}_{\mathcal{D}}[y(\mathbf{x})] - < t|\mathbf{x} >\}^2 \\
&= \{y(\mathbf{x}) - \mathcal{E}_{\mathcal{D}}[y(\mathbf{x})]\}^2 + \{\mathcal{E}_{\mathcal{D}}[y(\mathbf{x})] - < t|\mathbf{x} >\}^2 \\
&+ 2\{y(\mathbf{x}) - \mathcal{E}_{\mathcal{D}}[y(\mathbf{x})]\}\{\mathcal{E}_{\mathcal{D}}[y(\mathbf{x})] - < t|\mathbf{x} >\}
\end{aligned}$$

- taking expectations w.r.t. $\mathcal{E}_{\mathcal{D}}$,

$$\begin{aligned}
\mathcal{E}_{\mathcal{D}}\left[\left(y(\mathbf{x};\mathbf{w}(\mathcal{D})) - < t|\mathbf{x} >\right)^2\right] &= \mathcal{E}_{\mathcal{D}}\left[\{y(\mathbf{x}) - \mathcal{E}_{\mathcal{D}}[y(\mathbf{x})]\}^2\right] \\
&+ \{\mathcal{E}_{\mathcal{D}}[y(\mathbf{x})] - < t|\mathbf{x} >\}^2
\end{aligned}$$

Test sets

- Generalization errors can not be measured, but can be estimated using a finite $test\ set$
- The bias-variance trade-off quantities can be estimated by drawing multiple training sets (can in fact be overlapping i.e. cross-validation)

Model capacity and test errors

- The Generalization error depends on the interplay between model flexibility and training set size

Weight decay

- Weight decay is a means of soft capacity control

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:

$$
\begin{aligned}
y(\mathbf{x}) &= \tanh\left(\sum_{j=0}^{n_H} W_j h_j(\mathbf{x})\right) \\
h_j(\mathbf{x}) &= \tanh\left(\mathbf{w}^\top \mathbf{x} + w_0\right) \\
h_0 &= 1
\end{aligned}
$$

- 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

$$
\begin{aligned}
y(\mathbf{x}) &= \sum_{j=0}^{n_H} W_j h_j(\mathbf{x}) \\
h_j(\mathbf{x}) &= \tanh\left(\mathbf{w}^\top \mathbf{x} + w_{0,j}\right) \\
h_0 &= 1
\end{aligned}
$$

- 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 $a_j = \sum_i w_{j,i} z_i$
- then the derivative can be expressed

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

- Let $\delta_j = \frac{\partial E}{\partial a_j}$, note also $\frac{\partial a_j}{\partial w_{ji}} = z_i$, this leads to

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

- Computing the $\delta' s$

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

Local quadratic approximation

- Second order Taylor expansion of the costfunction

$$
\begin{aligned}
E(\mathbf{w}) \;&\approx\; E(\mathbf{w}_0) + \sum_j \frac{\partial E}{\partial w_j}\left(w_j - w_{0,j}\right) \\
&\quad + \frac{1}{2}\sum_{j,k} \frac{\partial^2 E}{\partial w_j \partial w_k}\left(w_j - w_{0,j}\right)\left(w_k - w_{0,k}\right) \\
E(\mathbf{w}) \;&\approx\; E(\mathbf{w}_0) + \frac{\partial E}{\partial \mathbf{w}}\left(\mathbf{w} - \mathbf{w}_0\right) \\
&\quad + \frac{1}{2}\left(\mathbf{w} - \mathbf{w}_0\right)^T \frac{\partial^2 E}{\partial \mathbf{w} \partial \mathbf{w}^T}\left(\mathbf{w} - \mathbf{w}_0\right)
\end{aligned}
$$

- The matrix $\mathbf{H} = \frac{\partial^2 E}{\partial \mathbf{w} \partial \mathbf{w}^T}$ is called the $Hessian$
- The local approximation to the gradient (∇E) is given by

$$
\frac{\partial E}{\partial \mathbf{w}}(\mathbf{w}) \approx \frac{\partial E}{\partial \mathbf{w}}(\mathbf{w}_0) + \mathbf{H}(\mathbf{w}_0)\left(\mathbf{w} - \mathbf{w}_0\right)
$$

Expansion around a minimum

- At a minimum $\nabla E = 0$

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

- The Hessian is real and symmetric, hence a has a set of orthonormal eigenvectors

$$\mathbf{H}\mathbf{u}_j = \lambda_j \mathbf{u}_j$$
$$\mathbf{u}_i^\top \mathbf{u}_j = \delta_{ij}$$

- At a minimum the Hessian is positive: $\mathbf{v}^\top \mathbf{H} \mathbf{v} > 0$
- in particular for all eigenvectors

$$\mathbf{u}_j^\top \mathbf{H} \mathbf{u}_j = \lambda_j > 0$$

- hence, when the Hessian is positive all eigenvalues are positive

The generalization error

- 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

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

- Now consider the limit of large sets, the error per example is

$$\begin{aligned} E &= \lim_{N \to \infty} \frac{1}{2N} \sum_{n=1}^{N} (y(\mathbf{x}^n; \mathbf{w}) - t^n)^2 \\ &= \frac{1}{2} \int \int (y(\mathbf{x}; \mathbf{w}) - t)^2 p(t, \mathbf{x}) dt d\mathbf{x} \end{aligned}$$

- This is the average (or expected) error on a test datum $(\mathbf{x}, t)$, which we call the generalization error.

Crossvalidation

- Generalization errors can not be measured, but can be estimated using a finite *test set*
- We would like to use as many examples as possible for training.
- Crossvalidation: Split the data set in V subsets $\mathcal{D}_v$

$$\mathcal{D} = \cup_{v=1}^{V} \mathcal{D}_v \tag{2}$$

- For $v = 1, ..., V$ train on $\mathcal{D}/\mathcal{D}_v$ and estimate the test error by

$$\begin{aligned} E_{\text{test}} &= \frac{1}{V} \sum_{v=1}^{V} E_v \\ E_v &= \frac{1}{2} \sum_{n \in \mathcal{D}_v} (y(\mathbf{x}^n; \mathbf{w}) - t^n)^2 \end{aligned}$$

- Extreme (and likely optimal): $V = N$, aka leave-one-out crossvalidation

Model capacity and test errors

- The Generalization error depends on the interplay between model flexibility and training set size
- The learning curve is the relation between generalization and training set size: $E_{\mathrm{test}}(N)$ vs. N.
- The generalization error is determined by the complexity of the model and the amount of data N.
- The model complexity is controlled by regularization and by pruning

Regularization by weight decay

- Weight decay is a means of soft capacity control

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

- Analysis of weight decay: Second order Taylor expansion of the costfunction

$$\begin{aligned}
E(\mathbf{w}) &\approx E(\mathbf{w}_0) + \sum_j \frac{\partial E}{\partial w_j}(w_j - w_{0,j}) \\
&+ \frac{1}{2}\sum_{j,k}\frac{\partial^2 E}{\partial w_j \partial w_k}(w_j - w_{0,j})(w_k - w_{0,k}) \\
E(\mathbf{w}) &\approx E(\mathbf{w}_0) + \frac{\partial E}{\partial \mathbf{w}}(\mathbf{w} - \mathbf{w}_0) \\
&+ \frac{1}{2}(\mathbf{w} - \mathbf{w}_0)^T \frac{\partial^2 E}{\partial \mathbf{w}\partial \mathbf{w}^T}(\mathbf{w} - \mathbf{w}_0)
\end{aligned}$$

- The matrix $\mathbf{H} = \frac{\partial^2 E}{\partial \mathbf{w}\partial \mathbf{w}^T}$ is called the $Hessian$

Weight decay

- Analysis of weight decay:

$$
\begin{aligned}
E(\mathbf{w}) \;\approx\; & E(\mathbf{w}_0) + \frac{\partial E}{\partial \mathbf{w}} (\mathbf{w} - \mathbf{w}_0) \\
& + \frac{1}{2} (\mathbf{w} - \mathbf{w}_0)^T \, \mathbf{H} \, (\mathbf{w} - \mathbf{w}_0)
\end{aligned}
$$

- Hence the minimum solves

$$
\frac{\partial E}{\partial \mathbf{w}} + \mathbf{H} (\mathbf{w} - \mathbf{w}_0) = 0
$$

$$
\mathbf{w}^* = \mathbf{w}_0 - \mathbf{H}^{-1} \frac{\partial E}{\partial \mathbf{w}}
$$

Weight decay

- Now if there is a non-zero weight decay

$$\frac{\partial \tilde{E}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial E}{\partial \mathbf{w}} + \nu \mathbf{w}$$

- Hence the new minimum solves

$$\frac{\partial E}{\partial \mathbf{w}_0} + \mathbf{H}_0(\mathbf{w}^* - \mathbf{w}_0) + \nu(\mathbf{w}^* - \mathbf{w}_0) = 0$$

- while the old minimum solves

$$\frac{\partial E}{\partial \mathbf{w}_0} + \mathbf{H}_0(\mathbf{w} - \mathbf{w}_0) = 0$$

- this means that the new and the old minima are related as

$$\mathbf{w}^* - \mathbf{w}_0 = (\mathbf{H}_0 + \nu \mathbf{1})^{-1} \mathbf{H}_0(\mathbf{w} - \mathbf{w}_0)$$

Saliency: Optimal Brain Damage

- How much does the training error increase if we delete a weight
- Second order expansion:

$$
\begin{aligned}
E(\mathbf{w}) \;&\approx\; E(\mathbf{w}^*) + \frac{\partial E}{\partial \mathbf{w}}\left(\mathbf{w} - \mathbf{w}^*\right) \\
&\quad + \frac{1}{2}\left(\mathbf{w} - \mathbf{w}^*\right)^T \mathbf{H} \left(\mathbf{w} - \mathbf{w}^*\right)
\end{aligned}
$$

- Deletion of the j'th weight: $\mathbf{w} - \mathbf{w}^* = w_j \mathbf{e}_j$

$$
\begin{aligned}
E(\mathbf{w}) \;&\approx\; E(\mathbf{w}^*) + \frac{\partial E}{\partial \mathbf{w}} w_j \mathbf{e}_j \\
&\quad + \frac{1}{2} w_j \mathbf{e}_j^T \mathbf{H} w_j \mathbf{e}_j
\end{aligned}
$$

$$
\begin{aligned}
E(\mathbf{w}) \;&\approx\; E(\mathbf{w}^*) + \frac{\partial E}{\partial w_j} w_j \\
&\quad + \frac{1}{2} \mathbf{H}_{j,j} w_j^2
\end{aligned}
$$

Saliency: Optimal Brain Damage

- However, in the minimum the first derivative is zero, hence

$$\Delta E(\mathbf{w})_{\rm obd} \approx \frac{1}{2}\mathbf{H}_{jj} w_j^2$$

  defining the OBD *saliency*

- If the retraining contribution is included (the un-pruned weights are not optimal after pruning) we get instead the OBS saliency

$$\Delta E(\mathbf{w})_{\rm obs} \approx \frac{1}{2}\frac{w_j^2}{(\mathbf{H}^{-1})_{jj}}$$

Estimating Generalization: Penalties

- The training set error is a downward biased estimate

$$E_{\text{test}} \approx E_{\text{train}} + d\sigma^2$$

- If we have used regularization (by weight decay)

$$E_{\text{test,GPE}} \approx E_{\text{train}} + d_{\text{eff}}\sigma^2$$

$$d_{\text{eff}} = \sum_{j=1}^{d} \frac{\lambda_j}{\lambda_j + \nu}$$

- The noise variance can be estimated

$$\widehat{\sigma^2} \approx \frac{2E_{\text{train}}}{N - d_{\text{eff}}}$$

- Combining we find

$$E_{\text{test,GPE}} \approx \frac{N + d_{\text{eff}}}{N - d_{\text{eff}}} E_{\text{train}}$$

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
- Find η by line search along the search direction $\mathbf{d}^{(\tau)} = -\nabla E|_{\mathbf{w}^{(\tau)}}$:

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

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

Signal detection: Bayes decision theory

- A signal detection system (or pattern classifier) provides a rule for assigning a measurement to a given signal category (class)
- Hence, a classifier divides measument space (feature space) into disjoint regions $\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_c$, such that measurements that fall into region $\mathcal{R}_k$ are assigned with class $\mathcal{C}_k$.
- Boundaries between regions are denoted decision surfaces or decision boundaries

Signal Detection: Bayes decision theory

Figure 1: Schematic plot of the densities for a measured signal drawn from either of two populations $\mathcal{C}_1, \mathcal{C}_2$

$$\begin{aligned} P(\text{error}) &= P(x \in \mathcal{R}_2, \mathcal{C}_1) + P(x \in \mathcal{R}_1, \mathcal{C}_2) \\ &= P(x \in \mathcal{R}_2 | \mathcal{C}_1) P(\mathcal{C}_1) + P(x \in \mathcal{R}_1 | \mathcal{C}_2) P(\mathcal{C}_2) \\ &= \left(\int_{\mathcal{R}_2} p(x|\mathcal{C}_1) dx \right) P(\mathcal{C}_1) + \left(\int_{\mathcal{R}_1} p(x|\mathcal{C}_2) dx \right) P(\mathcal{C}_2) \end{aligned}$$

- The probability of error is minimized if we assign points to $\mathcal{R}_1$, whenever $p(x|\mathcal{C}_1)P(\mathcal{C}_1) > p(x|\mathcal{C}_2)P(\mathcal{C}_2)$
- Using Bayes' theorem, this is equivalent to assign points to $\mathcal{R}_1$, whenever $p(\mathcal{C}_1|x) > p(\mathcal{C}_2|x)$, since we can divide by $p(\mathbf{x})$ on both sides of the inequality. Hence, the Bayes optimal signal detection system chooses the most probable class given the measurement.

The Bayesian paradigm

- The density of the measured signals is modelled by a parameterized density: $p(x) \sim p(x|\theta)$.
- Let $\chi = \{x_1, x_2, x_3, ..., x_N\}$ be a *training set*
- Objective: Find the distribution of the parameter vector, $p(\theta|\chi)$, hence the parameters are considered stochastic.

The likelihood function

- Let $\chi = \{x_1, x_2, x_3, ..., x_N\}$ be a *training set*
- We use Bayes theorem

$$p(\theta|\chi) = \frac{p(\chi|\theta)p(\theta)}{p(\chi)}$$

- The function $p(\chi|\theta)$ is called the likelihood function (more correct the likelihood of the parameter vector θ). The density $p(\theta)$ is called the *a priori* or *prior* parameter distribution.

- If the prior is "flat" in the neighborhood of the peak of $p(\chi|\theta)$, we have

$$p(\theta|\chi) \propto p(\chi|\theta)$$

- thus, the most probable parameters are the maximum likelihood parameters.

Maximum likelihood learning

- The training set is $D = (\chi_y, \chi_x)$, with $\chi_t = \{t_1, t_2, t_3, ..., t_N\}$ and $\chi_x = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_N\}$

$$p(\chi_t|\chi_x, \mathbf{w}) = \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w})$$

- The costfunction is then

$$E(\mathbf{w}) = -\log\left(\prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w})\right)$$

$$E(\mathbf{w}) = \sum_{n=1}^{N} -\log p(t_n|\mathbf{x}_n, \mathbf{w})$$

Two class problem

- Let the labels for a two class problem be $t_n = 0$ for one class and $t_n = 1$ for the other.
- Let the network output be $0 \leq y \leq 1$ be the probability of $t = 1$, then
- we can write the likelihood as

$$
\begin{aligned}
p(\chi_t|\chi_x, \mathbf{w}) &= \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w}) \\
&= \prod_{n=1}^{N} y(\mathbf{x}_n|\mathbf{w})^{t_n}[1 - y(\mathbf{x}_n|\mathbf{w})]^{(1-t_n)}
\end{aligned}
$$

- and the costfunction becomes

$$
E(\mathbf{w}) = -\sum_{n=1} t_n \log y(\mathbf{x}_n|\mathbf{w}) + (1 - t_n) \log[1 - y(\mathbf{x}_n|\mathbf{w})]
$$

- this is called the *entropic costfunction*

Properties of the costfunction

- The cost is minimal if $y_n = t_n$

$$E(\mathbf{w}) = -\sum_{n=1} t_n \log y(\mathbf{x}_n|\mathbf{w}) + (1 - t_n) \log[1 - y(\mathbf{x}_n|\mathbf{w})]$$

- the derivative w.r.t. y is

$$\frac{\partial E}{\partial y_n} = -\left[\frac{t_n}{y_n} - \frac{1 - t_n}{1 - y_n}\right]$$

Figure 2: Entropic costfunction: dependence on y_n

Properties of the costfunction cont'd

- The entropic costfunction penalizes wrong decisions more heavily than least squares method:

$$E(\mathbf{w}) = -\sum_{n=1} t_n \log y(\mathbf{x}_n|\mathbf{w}) + (1-t_n)\log[1-y(\mathbf{x}_n|\mathbf{w})]$$

- Let $t_n = 1$ and $y_n = 1 - \epsilon_n$ (correct decision)

$$E^n = -\log y_n = -\log[1-\epsilon_n] \approx \epsilon_n$$

- Let $t_n = 0$ and $y_n = 1 - \epsilon_n$ (very wrong decision!)

$$E^n = -\log[1-y_n] = -\log[1-(1-\epsilon_n)] = \log \epsilon_n$$

- Now for the sum of squares costfunction we have for the same situations

$$E^n = (1-(1-\epsilon_n))^2 = (\epsilon_n)^2$$
$$E^n = (0-(1-\epsilon_n))^2 = (1-\epsilon_n)^2$$

Properties of the costfunction cont'd

- We use $0 \leq y \leq 1$ coding of output unit based on linear standard MLP ($a_o(\mathbf{x}|\mathbf{w})$)

$$y(\mathbf{x}|\mathbf{w}) = \frac{1}{1 + \exp(-a_o(\mathbf{x}|\mathbf{w}))}$$

- Backprop rule

$$\frac{\partial E^n}{\partial \mathbf{w}_{jk}} = \delta_j^n z_k^n$$

- The output weight derivative is given by

$$\delta_o^n = \frac{\partial E^n}{\partial a_o^n} = \frac{\partial E^n}{\partial y^n}\frac{\partial y^n}{\partial a_o^n} = \frac{y^n - t^n}{y^n(1-y^n)}\frac{\partial g(a_o^n)}{\partial a_o^n}$$

  the derivative is given by $g'(a) = g(a)[1 - g(a)]$

- and the output error is then simply

$$\delta_o^n = \frac{y^n - t^n}{y^n(1-y^n)} y^n(1-y^n) = y^n - t^n$$

Multiple classes

- We use $0 \leq y\ 1$ coding for C classes and we want the outputs to be the posterior probabilities $P(C|\mathbf{x})$, hence they "should sum to one"

$$y_k(\mathbf{x}) = \frac{a_k(\mathbf{x})}{\sum_k a_k(\mathbf{x})}$$

- Targets are represented by 0-1 vectors:

$$\mathbf{t}_k = [0, 0, 0, ..., 1, 0, 0]$$

- The likelihood function is given by

$$p(\mathbf{t}|\mathbf{x}) = \prod_{k=1}^{C} y_k(\mathbf{x})^{t_k}$$

Multiple classes cont'd

- The likelihood function is given by

$$
\begin{aligned}
p(\mathbf{t}|\mathbf{x}) &= \prod_{k=1}^{C} y_k(\mathbf{x})^{t_k} \\
E &= -\sum_n \sum_k t_k^n \log y_k^n
\end{aligned}
$$

- The derivatives are relatively simple again

$$
\begin{aligned}
\frac{\partial E^n}{\partial a_k} &= \sum_{k'} \frac{\partial E^n}{\partial y_{k'}} \frac{\partial y_{k'}}{\partial a_k} \\
\frac{\partial y_{k'}}{\partial a_k} &= \delta_{kk'} y_k - y_{k'} y_k \\
\frac{\partial E^n}{\partial y_{k'}} &= -\frac{t_{k'}}{y_{k'}}
\end{aligned}
$$

- and we find

$$
\begin{aligned}
\frac{\partial E^n}{\partial a_k} &= \sum_{k'} \frac{t_{k'}}{y_{k'}}(\delta_{kk'}y_k - y_k y_{k'}) \\
&= -(t_k - y_k \sum_{k'} t_{k'}) \\
&= y_k - t_k
\end{aligned}
$$

Pruning and example

- We use pruning by OBD
- Stop pruning based on generalization error

Density estimation

- We want to model the density of a stochastic signal source

$$p(\mathbf{x}) \sim p(\mathbf{x}|\mathbf{w})$$

- where the family $p(\mathbf{x}|\mathbf{w})$ is a given parametric density.
- A density model can e.g. be used for outlier detection: How likely is a data point?

Maximum likelihood learning

- The training set is $D = (\chi)$, with $\chi = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_N\}$
- the likelihood function is given by

$$p(\chi|\mathbf{w}) = \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{w})$$

- The costfunction is then

$$E(\mathbf{w}) = -\log\left(\prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{w})\right)$$

$$E(\mathbf{w}) = \sum_{n=1}^{N} -\log p(\mathbf{x}_n|\mathbf{w})$$

Gaussian mixtures

- The gaussian mixture model is defined

$$p(\mathbf{x}|\mathbf{w}) = \sum_{j=1}^{M} P(j) p(\mathbf{x}|j, \mathbf{w}_j)$$

- Where each component density is a normal distribution with parameters: $\mathbf{w}_j = \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}$
- Think of the stochastic process as a *two-step* process: first draw a component number j with relative probabilities $P(j)$, then draw a random vector from the given component. This is the way to simulate data from the this source.

Maximum likelihood learning for GM

- The costfunction is

$$
\begin{aligned}
E(\mathbf{w}) &= \sum_{n=1}^{N} - \log p(\mathbf{x}_n|\mathbf{w}) \\
&= \sum_{n=1}^{N} - \log \sum_{j=1}^{M} p(\mathbf{x}_n|j)P(j)
\end{aligned}
$$

- We will simplify the family to isotropic Gaussians, al expressions generalize easily to full covariance matrices,

$$
p(\mathbf{x}|\boldsymbol{\mu}_j, \sigma_j^2) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp\left(-\frac{(\mathbf{x} - \boldsymbol{\mu}_j)^2}{2\sigma_j^2} \right)
$$

- The derivative w.r.t. the mean value vector is

$$
\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{\mu}_j} &= -\sum_{n=1}^{N} \frac{\partial/\partial\boldsymbol{\mu}_j \sum_{j'=1}^{M} p(\mathbf{x}_n|j')P(j')}{p(\mathbf{x}_n|\mathbf{w})} \\
&= -\sum_{n=1}^{N} P(j|\mathbf{x}_n) \frac{(\boldsymbol{\mu}_j - \mathbf{x}_n)}{\sigma_j^2}
\end{aligned}
$$

Maximum likelihood learning for GM

- The derivative w.r.t. the mean value vector is

$$\frac{\partial E}{\partial \boldsymbol{\mu}_j} = \sum_{n=1}^{N} P(j|\mathbf{x}_n) \frac{(\boldsymbol{\mu}_j - \mathbf{x}_n)}{\sigma_j^2}$$

- the derivative w.r.t. the widths is given by

$$\frac{\partial E}{\partial \sigma_j} = \sum_{n=1}^{N} P(j|\mathbf{x}_n) \left[\frac{d}{\sigma_j} - \frac{(\boldsymbol{\mu}_j - \mathbf{x}_n)^2}{\sigma_j^3} \right]$$

- We can understand these rules, let us try to solve them by equating the derivative to zero

$$\widehat{\boldsymbol{\mu}_j} = \frac{\sum_{n=1}^{N} P(j|\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} P(j|\mathbf{x}_n)}$$

and

$$\widehat{\sigma_j^2} = \frac{1}{d} \frac{\sum_{n=1}^{N} P(j|\mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_j)^2}{\sum_{n=1}^{N} P(j|\mathbf{x}_n)}$$

Maximum likelihood learning for GM

- Next we want to estimate $P(j)$. Note that the prior probabilities sum to unity:

$$\sum_{j=1}^{M} P(j) \;=\; 1$$

- Use the softmax trick

$$P(j) \;=\; \frac{\exp(\gamma_j)}{\sum_{j'=1}^{M} \exp(\gamma_{j'})}$$

- The derivative of the cost function is

$$\frac{\partial E}{\partial \gamma_j} \;=\; \sum_{k=1}^{M} \frac{\partial E}{\partial P(k)} \frac{\partial P(k)}{\partial \gamma_j}$$

Maximum likelihood learning for GM

- The derivative of the cost function is

$$\frac{\partial E}{\partial \gamma_j} = \sum_{k=1}^{M} \frac{\partial E}{\partial P(k)} \frac{\partial P(k)}{\partial \gamma_j}$$

$$\frac{\partial E}{\partial P(k)} = -\sum_{n=1}^{N} \frac{1}{p(\mathbf{x}_n)} p(\mathbf{x}_n|k) = -\sum_{n=1}^{N} \frac{P(k|\mathbf{x}_n)}{P(k)}$$

$$\frac{\partial P(k)}{\partial \gamma_j} = \delta_{k,j} P(k) - P(k)P(j)$$

- hence,

$$\frac{\partial E}{\partial \gamma_j} = -\sum_{n=1}^{N} [P(j|\mathbf{x}_n) - P(j)] = 0$$

- the solution is

$$\widehat{P(j)} = \frac{1}{N} \sum_{n=1}^{N} P(j|\mathbf{x}_n)$$

The EM algorithm

- The Estimate-Maximize algorithm is a general scheme for maximum likelihood estimation. Note that the change in costfunction that occurs when we iterate the estimates

$$
\begin{aligned}
E^{\mathrm{new}} - E^{\mathrm{old}} &= -\sum_{n=1}^{N} \log \frac{p^{\mathrm{new}}(\mathbf{x}_n)}{p^{\mathrm{old}}(\mathbf{x}_n)} \\
&= -\sum_{n=1}^{N} \log \frac{\sum_{j=1}^{M} p^{\mathrm{new}}(\mathbf{x}_n|j) P^{\mathrm{new}}(j)}{p^{\mathrm{old}}(\mathbf{x}_n)} \frac{P^{\mathrm{old}}(j|\mathbf{x}_n)}{P^{\mathrm{old}}(j|\mathbf{x}_n)} \\
&\leq -\sum_{n=1}^{N} \sum_{j} P^{\mathrm{old}}(j|\mathbf{x}_n) \log \frac{p^{\mathrm{new}}(\mathbf{x}_n|j) P^{\mathrm{new}}(j)}{p^{\mathrm{old}}(\mathbf{x}_n) P^{\mathrm{old}}(j|\mathbf{x}_n)}
\end{aligned}
$$

- The inequality is based on Jensen inequality:

$$
\log\left(\sum_j \lambda_j x_j\right) \geq \sum_j \lambda_j \log(x_j)
$$

- This is an upper bound so that it can be minimized and this give us similar results as for the maximum likelihood, now in iterative form.

The EM algorithm cont'd

- This is an upper bound so that it can be minimized. This gives us similar results as for the maximum likelihood, now in iterative form

$$\widehat{\boldsymbol{\mu}_j^{\mathrm{new}}} = \frac{\sum_{n=1}^{N} P^{\mathrm{old}}(j|\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} P^{\mathrm{old}}(j|\mathbf{x}_n)}$$

and

$$(\widehat{\sigma_j^{\mathrm{new}}})^2 = \frac{1}{d}\frac{\sum_{n=1}^{N} P^{\mathrm{old}}(j|\mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_j^{\mathrm{new}})^2}{\sum_{n=1}^{N} P^{\mathrm{old}}(j|\mathbf{x}_n)}$$

$$P_j^{\mathrm{new}} = \frac{1}{N}\sum_{n=1}^{N} P^{\mathrm{old}}(j|\mathbf{x}_n)$$

The K-means algorithm

- The K-means algorithm is a simple *clustering* algorithm aimed at minimizing the cost function for K clusters,

$$E = \sum_{j=1}^{K} \sum_{n \in S_j} (\mathbf{x}_n - \boldsymbol{\mu}_j)^2$$

- ...where $\boldsymbol{\mu}_j$ is the mean of the data points associated most with the $j'th$ component S_j (i.e. closest to)

$$\boldsymbol{\mu}_j = \frac{\sum_{n \in S_j} \mathbf{x}_n}{\sum_{n \in S_j} 1}$$

- Initialization is rather important, e.g., a cluster component which is never assigned any points will not be updated

The K-means algorithm cont'

- If we calculate the variance associated with the $j'th$ cluster

$$\sigma_j^2 = \frac{1}{d}\frac{\sum_{n\in S_j}(\mathbf{x}_n - \boldsymbol{\mu}_j)^2}{\sum_{n\in S_j} 1}$$

- and let the assignments be

  $P(j) = 1/K$ ...or even

  $P(j) = \sum_{n\in S_j} 1/N$

  we can actually use the parameters to define a density estimate as for the Gaussian mixture.