COURSE 02457

Signal Processing in Non-linear Systems:

Lecture 9

- Density estimation
- Maximum likelihood estimation
- Gaussian mixtures
- K-means clustering
- Learning GM's
- The EM algorithm
- Classification with mixtures
- Radial Basis Function network
- Training the RBF net
- Exercise 9

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.

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

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

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

Multivariate normal distribution

- In d dimensions, the multivariate normal probability density function is given by

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

  where $\boldsymbol{\mu}$ is a d-dimensional vector, and $\boldsymbol{\Sigma}$ is a $d \times d$ covariance matrix.

- The covariance matrix has a set of eigenvectors

$$\boldsymbol{\Sigma}\mathbf{u}_j = \lambda_j \mathbf{u}_j, \quad j = 1, ..., d$$

  or in matrix notation

$$\boldsymbol{\Sigma}\mathbf{U} = \boldsymbol{\Lambda}\mathbf{U}$$

  If we define a vector $\mathbf{z} = \mathbf{U}^\top \mathbf{x}$, then

$$\mathcal{E}(\mathbf{z}\mathbf{z}^\top) = \mathbf{U}^\top \mathbf{x}\mathbf{x}^\top \mathbf{U} = \mathbf{U}^\top \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}\mathbf{U}^\top = \boldsymbol{\Lambda}$$

  The covariances are zero!, hence, the $\mathbf{z}$ vector has uncorrelated components.

Gaussian mixtures

- The gaussian mixture model is defined

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

- 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 $two\text{-}step$ 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.

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.

- The K-means is equivalent to a "hard assigment" EM is we use a common variance and $P(j) = 1/K$.

Bayes' theorem – multivariate version

$$
\begin{aligned}
P(\mathcal{C}_k, \mathbf{x}) &= p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) \\
P(\mathcal{C}_k, \mathbf{x}) &= P(\mathcal{C}_k|\mathbf{x})p(\mathbf{x})
\end{aligned}
$$

$$
\begin{aligned}
P(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{p(\mathbf{x})} \\
p(\mathbf{x}|\mathcal{C}_k) &= \frac{P(\mathcal{C}_k|X^l)p(\mathbf{x})}{P(\mathcal{C}_k)}
\end{aligned}
$$

$$
\sum_{k=1}^{c} P(\mathcal{C}_k|\mathbf{x}) = 1
$$

$$
\sum_{k=1}^{c} p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) = p(\mathbf{x})
$$

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

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

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

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

- The inequality is based on Jensen's 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

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

and

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

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

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.

Signal detection with mixtures

- Let's recollect Bayes formula

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)P(C_k)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_k)P(C_k)}{\sum_{k'} p(\mathbf{x}|C_{k'})P(C_{k'})}$$

- This could be viewed as a simple network with basis functions

$$\phi_k(\mathbf{x}) = \frac{p(\mathbf{x}|C_k)}{\sum_{k'} p(\mathbf{x}|C_{k'})P(C_{k'})}$$

- weighted by the $P(C_k)$ for each output neuron

Radial Basis Function network for detection

- Using a single Gaussian per class might be over-simplified. So let us propose a Gaussian $mixture$ for each class

$$p(\mathbf{x}|C_k) = \sum_j p(\mathbf{x}|j)P(j|C_k)$$

- The marginal density

$$\begin{aligned} p(\mathbf{x}) &= \sum_k \sum_j p(\mathbf{x}|j)P(j|C_k)P(C_k) \\ &= \sum_j p(\mathbf{x}|j)P(j) \end{aligned}$$

- with priors defined by

$$P(j) = \sum_k P(j|C_k)P(C_k)$$

Radial Basis Function network for detection

- we are interested in a network that gives us the posterior probabilities

$$P(C_k|\mathbf{x}) = \frac{\sum_j p(\mathbf{x}|j)P(j|C_k)P(C_k)}{\sum_{j'} p(\mathbf{x}|j')P(j')}\frac{P(j)}{P(j)}$$
$$= \sum_j w_{k,j}\phi_j(\mathbf{x})$$

- with the definitions

$$\phi_j(\mathbf{x}) = \frac{p(\mathbf{x}|j)P(j)}{\sum_{j'} p(\mathbf{x}|j')P(j')} = P(j|\mathbf{x})$$
$$w_{k,j} = \frac{P(j|C_k)P(C_k)}{P(j)} = P(C_k|j)$$

- so the basis functions are "normalized" by spatially variant functions, hence no longer Gaussians.

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"

Regression from joint density

- Let $p(t, \mathbf{x})$ be a joint input-output density

$$\begin{aligned} y(\mathbf{x}) &= < t|x > \\ &= \int tp(t|\mathbf{x})dt \\ &= \frac{\int tp(t, \mathbf{x})dt}{\int p(t, \mathbf{x})dt} \end{aligned}$$

- where we used $p(a|b)p(b) = p(a, b)$
- If our joint density is of the form with centers (ν, μ)

$$p(t, \mathbf{x}) = \sum_{j=1}^{M} P(j) \frac{1}{(2\pi\sigma_j^2)^{\frac{d+c}{2}}} \exp\left(-\frac{(\mathbf{x} - \boldsymbol{\mu}_j)^2}{2\sigma_j^2} - \frac{(t - \nu_j)^2}{2\sigma_j^2}\right)$$

- then the conditional mean is given by

$$y(\mathbf{x}) = \frac{\sum_{j=1}^{M} P(j)\nu_j \exp\left(-\frac{(\mathbf{x}-\boldsymbol{\mu}_j)^2}{2\sigma_j^2}\right)}{\sum_{j=1}^{M} P(j) \exp\left(-\frac{(\mathbf{x}-\boldsymbol{\mu}_j)^2}{2\sigma_j^2}\right)}$$

Radial Basis Functions for regression

- Let the basis function be a gaussian

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{(\mathbf{x} - \boldsymbol{\mu}_j)^2}{2\sigma_j^2}\right)$$

- then the general RBF network is defined

$$y_j(\mathbf{x}) = \sum_{j=1}^{M} w_{k,j}\phi(\mathbf{x}) + w_{k,0}$$

Training RBF networks

- For fixed basis functions the weight can be trained using least squares for the linear model
- For fixed weights we can using gradients (or conjugate gradients) for the the basis function parameters.

Training RBF networks

- If the cost function is

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\sum_{k}(t_k^n - y_k^n)^2 \\
&= \frac{1}{2}\sum_{n=1}^{N}\sum_{k}\left(t_k^n - \sum_{j=1}^{Nh} w_{kj}\phi_j(\mathbf{x}^n)\right)^2 \\
\phi_j(\mathbf{x}^n) &= \exp\left(-\frac{(\mathbf{x}^n - \boldsymbol{\mu}_j)^2}{2\sigma_j^2}\right)
\end{aligned}
$$

- Then the derivatives w.r.t. $\boldsymbol{\mu}_j, \sigma_j^2$ become

$$
\frac{\partial E}{\partial \boldsymbol{\mu}_j} = \sum_{n=1}^{N}\sum_{k}(y_k^n - t_k^n) w_{kj}\phi_j(\mathbf{x}^n)\frac{(\mathbf{x}^n - \boldsymbol{\mu}_j)}{\sigma_j^2}
$$

$$
\frac{\partial E}{\partial \sigma_j} = \sum_{n=1}^{N}\sum_{k}(y_k^n - t_k^n) w_{kj}\phi_j(\mathbf{x}^n)\frac{(\mathbf{x}^n - \boldsymbol{\mu}_j)^2}{\sigma_j^3}
$$

Training RBF networks cont'd

- For a given set basis functions we can find the optimal weights (linear model)

$$\frac{\partial E}{\partial w_{kj}} = \sum_{n=1}^{N} \sum_{k} (y_k^n - t_k^n) \phi_j(\mathbf{x}^n)$$

- Equating this to zero we find the solution

$$\begin{aligned} \mathbf{W} &= \mathbf{B}\mathbf{A}^{-1} \\ A_{jj'} &= \frac{1}{N} \sum_{n=1}^{N} \phi_j(\mathbf{x}^n) \phi_{j'}(\mathbf{x}^n) \\ B_{kj} &= \frac{1}{N} \sum_{n=1}^{N} t_k^n \phi_j(\mathbf{x}^n) \end{aligned}$$

Training RBF networks cont'd

- If we regularize the weight estimate by weight decay, the augmented cost function is,

$$\tilde{E}(\mathbf{w}) \;=\; \frac{1}{2}\sum_{n=1}^{N}\sum_{k}(t_k^n - y_k^n)^2 + \frac{1}{2}\alpha\sum_{kj} w_{kj}^2$$

- and the derivative

$$\frac{\partial \tilde{E}}{\partial w_{kj}} \;=\; \frac{\partial E}{\partial w_{kj}} + \alpha w_{kj}$$

- Hence the regularized solution

$$\tilde{\mathbf{W}} \;=\; \mathbf{B}(\mathbf{A} + \alpha\mathbf{1})^{-1}$$