LECTURE 20: MLPs, RBFs and Statistical PR

- Bayes discriminants and MLPs
- The role of MLP hidden units
- Bayes discriminants and RBFs
- Comparison between MLPs and RBFs

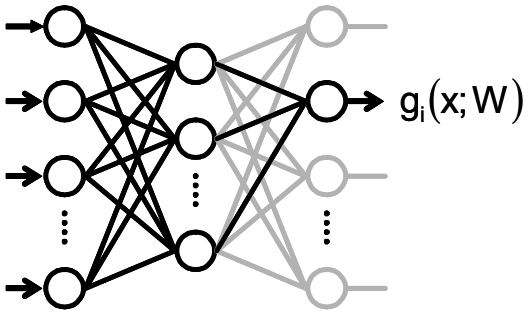
Bayes discriminants and MLPs (1)

- **As we have seen throughout the course, the classifier that minimizes the probability of error is defined by the maximum a posteriori rule**

$$\omega^* = \underset{\omega_i}{\operatorname{argmax}} \{ g_i(x) = P(\omega_i \mid x) \}$$

- **How does the output of an MLP relate to this optimal classifier?**
  - Assume a MLP with a one-of-C encoding for the targets

$$t_i(x) = \begin{cases} 1 & \text{if } x \in \omega_i \\ 0 & \text{otherwise} \end{cases}$$



  - The contribution to the error of the i^{th} output is

$$J(W) = \sum_{\forall x} (g_i(x;W) - t_i)^2 =$$

$$= \sum_{x \in \omega_i} (g_i(x;W) - 1)^2 + \sum_{x \notin \omega_i} (g_i(x;W) - 0)^2 =$$

$$= N \left[\frac{N_i}{N} \frac{1}{N_i} \sum_{x \in \omega_i} (g_i(x;W) - 1)^2 + \frac{N - N_i}{N} \frac{1}{N - N_i} \sum_{x \notin \omega_i} (g_i(x;W) - 0)^2 \right]$$

    - where $g_i(x;W)$ is the discriminant function computed by the MLP for the i^{th} class and the set of weights W

Bayes discriminants and MLPs (2)

- Assuming infinite number of examples, the previous criterion function becomes

$$\lim_{N\to\infty}\frac{1}{N}J(W)=\lim_{N\to\infty}\left[\frac{N_i}{N}\frac{1}{N_i}\sum_{x\in\omega_i}(g_i(x;W)-1)^2+\frac{N-N_i}{N}\frac{1}{N-N_i}\sum_{x\notin\omega_i}(g_i(x;W)-0)^2\right]=$$

$$=\lim_{N\to\infty}\left(\frac{N_i}{N}\right)\lim_{N\to\infty}\left(\frac{1}{N_i}\sum_{x\in\omega_i}(g_i(x;W)-1)^2\right)+\lim_{N\to\infty}\left(\frac{N-N_i}{N}\right)\lim_{N\to\infty}\left(\frac{1}{N-N_i}\sum_{x\notin\omega_i}(g_i(x;W)-0)^2\right)$$

$$=P(\omega_i)\int_x(g_i(x;W)-1)^2P(x\mid\omega_i)dx+P(\omega_{k\neq i})\int_x(g_i(x;W)-0)^2P(x\mid\omega_{k\neq i})dx$$

Bayes discriminants and MLPs (3)

- Expanding the quadratic terms and rearranging

$$\lim_{N\to\infty}\frac{1}{N}J(W)=\int_x\left(g_i^{\,2}(x;W)-2g_i(x;W)+1\right)P(x,\omega_i)dx+\int_x g_i^{\,2}(x;W)P(x,\omega_{k\neq i})dx=$$

$$=\int_x g_i^{\,2}(x;W)\left(P(x,\omega_i)+P(x,\omega_{k\neq i})\right)dx-\int_x 2g_i(x;W)P(x,\omega_i)dx+\int_x P(x,\omega_i)dx$$

$$=\int_x g_i^{\,2}(x;W)P(x)dx-\int_x 2g_i(x;W)P(\omega_i\mid x)P(x)dx+\underbrace{\int_x P^2(\omega_i\mid x)P(x)dx}_{\text{Add this term}}-\underbrace{\int_x P^2(\omega_i\mid x)P(x)dx}_{\text{Subtract this term}}+\int_x P(x,\omega_i)dx=$$

$$=\int_x\left(g_i(x;W)-P(\omega_i\mid x)\right)^2P(x)dx-\underbrace{\int_x P^2(\omega_i\mid x)P(x)dx+\int_x P(x,\omega_i)dx}_{\text{independent of } W}$$

Bayes discriminants and MLPs (4)

- **Therefore, by minimizing J(W) we also minimize**

$$\int_x \left(g_i(x;W) - P(\omega_i \mid x)\right)^2 P(x)dx$$

- Summing over all classes (output neurons), we can then conclude that back-prop attempts to minimize the following expression

$$\sum_{i=1}^{N_C} \int_x \left(g_i(x;W) - P(\omega_i \mid x)\right)^2 P(x)dx$$

- **Therefore, in the large sample limit , the outputs of the MLP will approximate (in a least-squares sense) the true a posteriori probabilities**

$$g_i(x;W) \cong P(\omega_i \mid x)$$

- Notice that nothing said here is specific to MLPs
  - Any discriminant function with adaptive parameters trained to minimize the sum squared error at the output of a 1-of-C encoding will approximate the a posteriori probabilities

Bayes discriminants and MLPs (5)

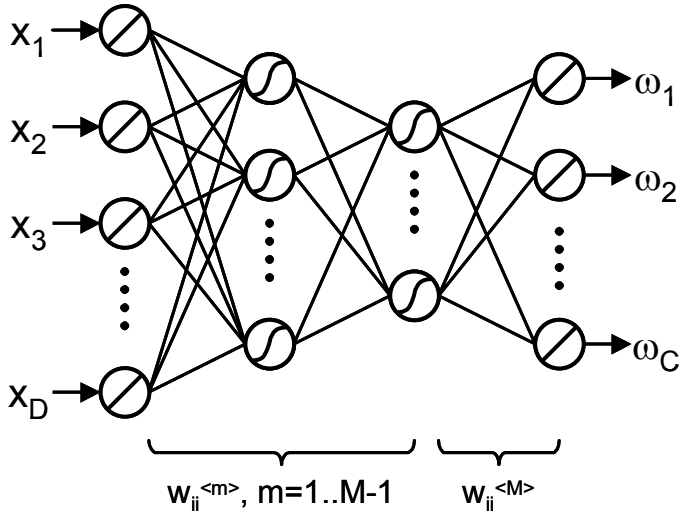
- **This result will be true if and only if**
  - We use a 1-of-C encoding with [0,1] outputs and
  - The MLP has enough hidden units to represent the a posteriori densities and
  - We have an infinite number of examples, and
  - The MLP does not get trapped in a local minima
- **In practice we will have a limited number of examples so**
  - The outputs will not always represent probabilities
    - For instance, there is no guarantee that they will sum up to 1
  - We can use this result to determine if the network has trained properly
    - If the sum of the outputs differs significantly from 1, it will be an indication that the MLP is not modeling the a posteriori densities properly and that we may have to change the MLP (topology, number of hidden units, etc.)
  - If we still want to interpret MLP outputs as probabilities, we must then enforce that they sum up to 1. This can be achieved by choosing the output's non-linearity to be exponential, and normalizing across the output layer

$$y_k^{<2>} = \frac{\exp\left(net_k^{<2>}\right)}{\sum_{k=1}^{N_O} \exp\left(net_k^{<2>}\right)}$$

    - This is known as the **soft-max** method, which receives its name from the fact that it can be interpreted as a smoothed version of the *winner-take-all* rule

The role of MLP hidden units (1)

- **Let us assume a MLP with non-linear activation functions for the hidden layer(s) and linear activation function for the output layer**



- **Let us also hold constant the set of hidden weights $w_{ij}^{<m>}$ m=1..M-1**
  - In this case, the minimization of J(W) w.r.t. the output weights $w_{ij}^{<M>}$ becomes a linear optimization problem, which can solved through the pseudo-inverse

$$W^{<M>} = \underset{W}{\operatorname{argmin}}\left\{\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{N_O}\left(y_k^{<M>(n} - t_k^{(n}\right)^2\right\} = \underset{W}{\operatorname{argmin}}\left\{\frac{1}{2}\left\|Y^{<M-1>}W^{<M>} - T\right\|^2\right\}$$

$$\Rightarrow W^{<M>} = \left(Y^{<M-1>^T}Y^{<M-1>}\right)^{-1}Y^{<M-1>^T}T = Y^{<M-1>^\dagger}T$$

- $W^{<M>}$ denotes the weights of the (output) linear layer and
- $Y^{<M-1>}$ denotes the outputs of the last hidden layer and
- T denotes the targets

From [Bishop, 1995]

The role of MLP hidden units (2)

- It can be shown that the role of the output biases is to compensate for the difference between the averages of the target values and the hidden unit outputs

$$w_{0k}^{<M>} = E[t_k] - \sum_{j=1}^{N_{H_{M-1}}} w_{jk}^{<M>} E[y_j^{<M-1>}]$$

  - Knowing that the output biases can be computed in this manner, we will assume for convenience that both T and $Y^{<M-1>}$ are zero-mean. As we will see in a minute, this will allow us to interpret certain terms as scatter matrices

- Plugging the pseudo-inverse into the objective function J(W) we obtain

$$J(W) = \frac{1}{2}\left\| Y^{<M-1>} \underline{W^{<M>}} - T \right\|^2 = \frac{1}{2}\left\| Y^{<M-1>} \underline{Y^{<M-1>\dagger} T} - T \right\|^2$$

  - Remember that $YY^{\dagger}Y \neq I$, the pseudo-inverse only works the other way ($Y^{\dagger}Y=I$) !

- And realizing that the SSE is the sum of ONLY the diagonal terms in $||\cdot||^2$

$$J(W) = \frac{1}{2} Tr\left\{ \left(Y^{<M-1>} Y^{<M-1>\dagger} T - T \right) \left(Y^{<M-1>} Y^{<M-1>\dagger} T - T \right)^t \right\}$$

- which, after some matrix manipulation [Bishop, 1995], can be expressed as

$$J(W) = \frac{1}{2} Tr\left[T^t T - S_B S_{TOT}^{-1} \right] \text{ where } \begin{cases} S_{TOT} = Y^{<M>t} Y^{<M>} \\ S_B = Y^{<M>t} T\, T^t Y^{<M>t} \end{cases}$$

The role of MLP hidden units (3)

- Now, since the product T^tT is independent of W, the minimization of J(W) is equivalent to maximizing J'(W)

$$J'(W) = \frac{1}{2} Tr\left[S_B S_{TOT}^{-1}\right]$$

- Since we are using 1-of-C encoding in the output layer, it can be shown that S_B becomes

$$S_B = \sum_{k=1}^{C} N_k^2 \left(\bar{y}_k^{<M-1>} - \bar{y}^{<M-1>}\right)\left(\bar{y}_k^{<M-1>} - \bar{y}^{<M-1>}\right)^T \qquad \text{with} \quad \begin{cases} \bar{y}_k^{<M-1>} = E\left[y_k^{<M-1>}\right]_{x \in \omega_k} \\ \bar{y}^{<M-1>} = E\left[y^{<M-1>}\right]_{\forall x} \end{cases}$$

  - $E[y_k^{<M-1>}]_{x \in \omega k}$ is the mean activation vector at the last hidden layer for all examples of class ω_k, and $E[y^{<M-1>}]$ is the mean activation vector regardless of class
  - Notice that this S_B only differs from the conventional between-class covariance matrix by having N_k^2 instead of N_k.
    - This means that this scatter matrix will have a strong bias in favor of classes that have a large number of examples
  - and S_{TOT} is the total covariance matrix at the output of the final hidden later

The role of MLP hidden units (4)

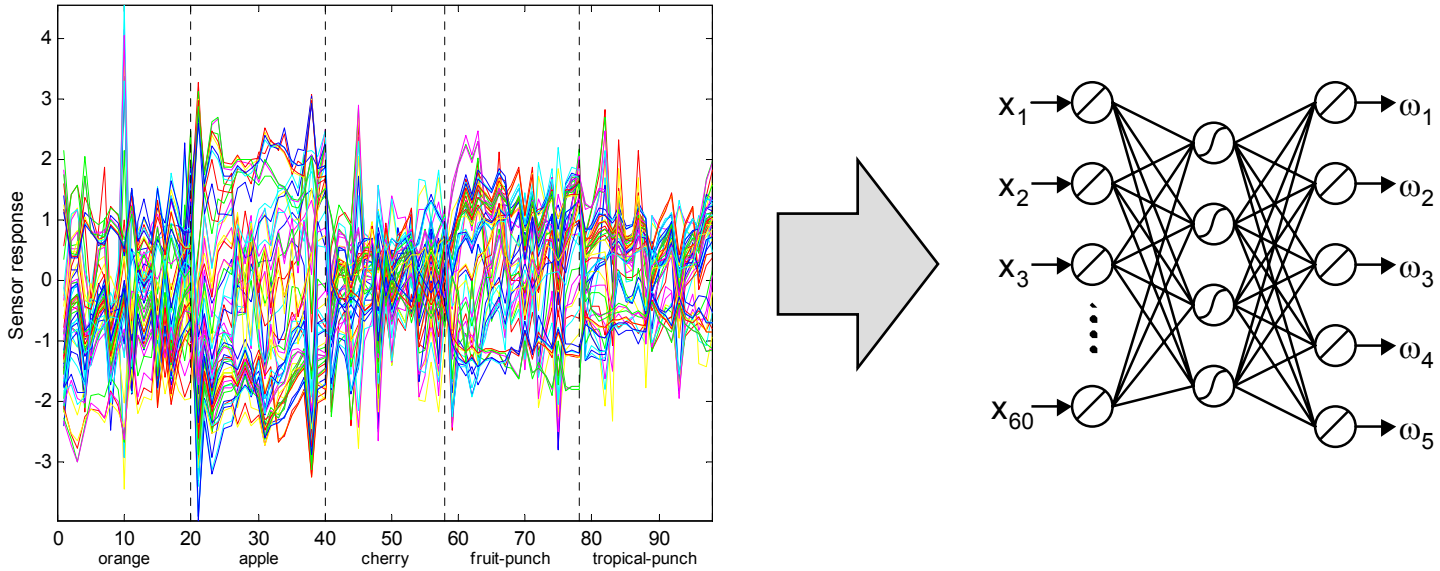
- **CONCLUSION**

> Minimization of the sum squared error between the desired targets and the outputs of an MLP with linear output neurons forces the hidden layer(s) to perform a non-linear transformation of the inputs that maximizes the discriminant function $\mathbf{Tr[S_B S_{TOT}^{-1}]}$ measured at the output of the (last) hidden layer

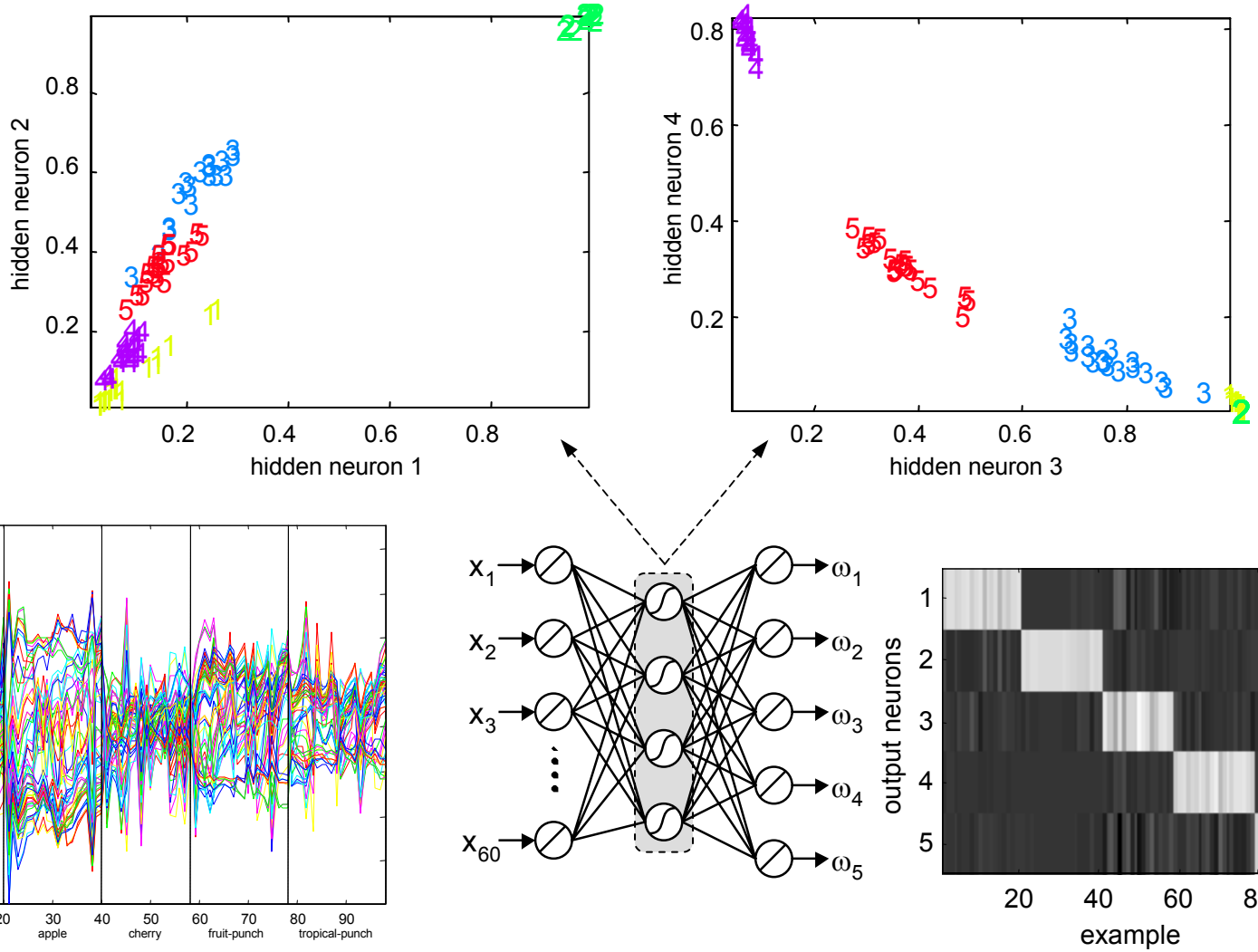
- In other words, the hidden layers of an MLP perform a non-linear version of Fisher's discriminant analysis (Lecture 10)
- This is precisely why MLPs have been demonstrated to perform so well in pattern classification tasks

An example

- **We train an MLP to classify five odors from an array of sixty gas sensors**
  - The MLP has sixty inputs, one for each gas sensor
  - There are five outputs, one for each odor
    - Output neurons use the 1-of-C encoding of classes
    - The output layer has linear activation function
  - Four hidden neurons are used (as many as LDA projections)
    - The hidden layer has the logistic sigmoidal activation function
  - Training
    - Hidden weights and biases trained with steepest descent rule
    - Output weights and biases trained with the pseudo-inverse rule



An example: results



Bayes discriminants and RBFs (1)

- **Recall from previous lectures that the posterior can be expressed as**

$$p(\omega_k \mid x) = \frac{p(x \mid \omega_k) p(\omega_k)}{\sum_{k'=1}^{C} p(x \mid \omega_{k'}) p(\omega_{k'})}$$

  - If we model each class with a single kernel function $p(x|\omega_k)$, this can be viewed as a simple network with normalized basis functions given by

$$\varphi_k(x) = \frac{p(x \mid \omega_k)}{\sum_{k'=1}^{C} p(x \mid \omega_{k'}) p(\omega_{k'})}$$

  - and hidden-to-output weights defined by

$$w_k = p(\omega_k)$$

  - This provides a very simple interpretation of RBFs as Bayesian classifiers and vice versa

Bayes discriminants and RBFs (2)

- **In some situations, however, a single kernel per class may not be sufficient to model the input space density**
  - Let us then use M different basis functions, which we will label by an index j
    - The class conditional density is then given by

$$p(x \mid \omega_k) = \sum_{j=1}^{M} p(x \mid j) p(j \mid \omega_k)$$

    - and the unconditional density is given by

$$p(x) = \sum_{k=1}^{C} p(x \mid \omega_k) p(\omega_k) = \sum_{j=1}^{M} p(x \mid j) p(j)$$

    - where the priors p(j) can be defined by

$$p(j) = \sum_{k=1}^{C} p(j \mid \omega_k) p(\omega_k)$$

  - With these expressions in mind, the posterior probabilities become

$$p(\omega_k \mid x) = \frac{p(x \mid \omega_k) p(\omega_k)}{p(x)} = \frac{\sum_{j=1}^{M} p(x \mid j) p(j \mid \omega_k) p(\omega_k)}{\sum_{j'=1}^{M} p(x \mid j') p(j')}$$

Bayes discriminants and RBFs (3)

- Moving the denominator into the summation, and adding the term p(j)/p(j)=1

$$p(\omega_k \mid x) = \frac{\sum_{j=1}^{M} p(x \mid j) p(j \mid \omega_k) p(\omega_k)}{\sum_{j'=1}^{M} p(x \mid j') p(j')} = \sum_{j=1}^{M} \left[\frac{p(x \mid j) p(j \mid \omega_k) p(\omega_k)}{\sum_{j'=1}^{M} p(x \mid j') p(j')} \frac{p(j)}{p(j)} \right]$$

- This operation allows us to regroup the expression as

$$p(\omega_k \mid x) = \sum_{j=1}^{M} w_{jk} \varphi_j(x)$$

  - where

$$\varphi_j(x) = \frac{p(x \mid j) p(j)}{\sum_{j'=1}^{M} p(x \mid j') p(j')} = p(j \mid x) \qquad w_{jk} = \frac{p(j \mid \omega_k) p(\omega_k)}{p(j)} = p(\omega_k \mid j)$$

INTERPRETATION

- The activation of a basis function can be interpreted as the posterior probability of the presence of its prototype pattern in the input space,
- The hidden-to-output weights can be interpreted as the posterior probabilities of the classes given the prototype pattern of the basis function, and
- **The output of the RBF can also be interpreted as the posterior probability of class membership**

Comparison between MLPs and RBFs (1)

- **Similarities**
  - Either model can function as an "**universal approximator**"
    - *MLPs and RBFs can approximate any functional continuous mapping with arbitrary accuracy, provided that the number of hidden units is sufficiently large*
- **Differences**
  - MLPs perform a global and distributed approximation of the target function, whereas RBF perform a local approximation
  - MLP partition feature space with hyper-planes; RBF decision boundaries are hyper-ellipsoids
  - The distributed representation of MLPs causes the error surface to have multiple local minima and nearly flat regions with very slow convergence. As a result training times for MLPs are usually larger than those for RBFs
  - MLPs exhibit better generalization properties than RBFs in regions of feature space outside of the local neighborhoods defined by the training set. On the other hand, extrapolation far from training data is oftentimes unjustified and dangerous
  - MLPs typically require fewer parameters than RBFs to approximate a non-linear function with the same accuracy

Comparison between MLPs and RBFs (2)

- **Differences (cont.)**
  - All the parameters in an MLP are trained simultaneously; parameters in the hidden and output layers of an RBF network are typically trained separately using an efficient, faster hybrid algorithm
  - MLPs may have multiple hidden layers with complex connectivity, whereas RBFs typically have only one hidden layer and full connectivity
  - The hidden neurons of an MLP compute the inner product between an input vector and their weight vector; RBFs compute the Euclidean distance between an input vector and the radial basis centers
  - The hidden layer of an RBF is non-linear, whereas the output layer is linear. In an MLP classifier, all layers are typically non-linear. Only when an MLP is used for non-linear regression, the output layer is typically linear.

Comparison between MLPs and RBFs (3)