Lecture 12: Multilayer perceptrons II

- Bayes discriminants and MLPs
- The role of hidden units
- An example

Bayes discriminants and MLPs (1)

- **As we have seen throughout the course, the classifier that minimizes the probability of error could be expressed as a family of discriminant functions defined by the maximum a posteriori**

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

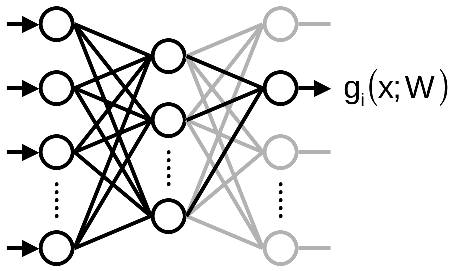
- **How does the output of a MLP relates to this optimal classifier?**
  - Assume a MLP with a one-of-C encoding for the targets

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

  - The contribution to the error of the ith output neuron is

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

  - Where $g_i(x;W)$ is the discriminant function computed by the MLP for the ith class and the set of weights W



Bayes discriminants and MLPs (2)

- **For an infinite number of examples, the previous criterion function becomes**

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

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

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

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

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

$$=\int_x g_i^2(x;W)P(x)dx-\int_x 2g_i(x;W)P(\omega_i\mid x)P(x)dx+\int_x P^2(\omega_i\mid x)P(x)dx-\int_x P^2(\omega_i\mid x)P(x)dx+\int_x P(x,\omega_i)dx=$$

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

Bayes discriminants and MLPs (3)

- **The back propagation rule changes W to minimize J(W), so in fact it is minimizing**

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

  - Summing over all classes (output neurons), then we conclude that back-prop also minimizes

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

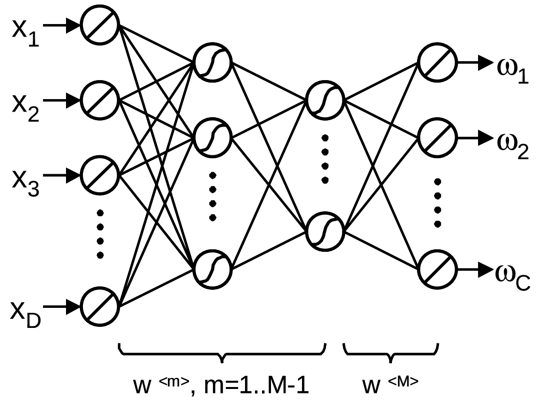
  - So, in the limit of infinite examples, the outputs of the MLP will approximate (in a least-squares sense) the true a posteriori probabilities

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

  - 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
- **This result will be true if and only if**
  - 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.)

The role of 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**



$$W = \arg\min_{W}\left\{\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}\left(y_k^{<M>(n} - t_k^{(n}\right)^2\right\}$$

- **If we hold constant the set of hidden weights $w_{ij}^{<m>}$ m=1..M-1, the minimization of the objective function J(W) with respect of the output weights $w_{ij}^{<M>}$ becomes a linear optimization problem and can, therefore, be solved in closed form**
  - It can be shown [Bishop, 1995] that the role of the output biases is to compensate for the difference between the averages (over the data set) of the target values and the weighted sum of the averages of the hidden unit outputs

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

  - This allows us to ignore the mean of the outputs and targets and express the objective function as (dropping index $^{(n}$ for clarity)

$$J(W) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}\left(\sum_{j=1}^{N_{H_{M-1}}} w_{jk}^{<M>} \underbrace{\tilde{y}_j^{<M-1>}}_{\text{zero-mean}} - \underbrace{\tilde{t}_i}_{\text{zero-mean}}\right)^2$$

$$\text{where }\begin{cases}\tilde{y}_j^{<M-1>} = y_j^{<M-1>} - E\left[y_j^{<M-1>}\right] \\ \tilde{t}_i = t_i - E[t_i]\end{cases}$$

The role of hidden units (2)

- To find the optimal output weights $w_{ij}^{<M>}$ we form the partial derivative

$$\frac{\partial J(W)}{\partial w_{jk}^{<M>}} = \sum_{n=1}^{N} \left(\sum_{j'=1}^{N_{H_{M-1}}} w_{j'k}^{<M>} \tilde{y}_{j'}^{<M-1>} - \tilde{t}_i \right) \tilde{y}_{j'}^{<M-1>} = 0$$

- We introduce the following matrix notation
  - $W^{<M>}$ denotes the weights of the linear layer
  - $Y^{<M-1>}$ denotes the zero-mean outputs of the last hidden layer (each column is an example, each row is an output)
  - T denotes the zero-mean targets (each column is an example, each row is an output)
- So the previous minimization problem becomes

$$Y^{<M-1>^T} Y^{<M-1>} W^{<M>^T} - Y^{<M-1>^T} T = 0$$

- And the optimal set of weights $W^{<M>}$ becomes

$$W^{<M>^T} = \underbrace{\left(Y^{<M-1>^T} Y^{<M-1>} \right)^{-1} Y^{<M-1>^T}}_{\text{pseudo-inverse of } Y^{<M-1>}} T = \left(Y^{<M-1>} \right)^{\dagger} T$$

  - It is important to notice that this solution can be calculated explicitly, no iterative procedure (i.e. steepest descent) is necessary

We now turn our attention to the hidden layer(s)

- Using matrix notation we can again express the objective function as

$$J(W) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{C} \left(\sum_{j=1}^{N_{H_{M-1}}} w_{jk}^{<M>} \underbrace{\tilde{y}_j^{<M-1>}}_{\text{zero-mean}} - \underbrace{\tilde{t}_i}_{\text{zero-mean}} \right)^2$$

$$= \frac{1}{2} \mathrm{Tr}\left[\left(Y^{<M-1>} W^{<M>^T} - T \right) \left(Y^{<M-1>} W^{<M>^T} - T \right)^T \right]$$

The role of hidden units (3)

- Substituting the optimal value of $W^{<M>}$ yields

$$J(W)=\frac{1}{2}\mathrm{Tr}\left[\left(Y^{<M>}Y^{<M>\dagger}T-T\right)\left(Y^{<M>}Y^{<M>\dagger}T-T\right)^T\right]=\frac{1}{2}\mathrm{Tr}\left[\underbrace{T^TT}_{\substack{\text{independent}\\ \text{of } W}}-S_BS_T^{-1}\right]$$

$$\text{where}\quad \begin{cases} S_T=Y^{<M>^T}Y^{<M>} \\ S_B=Y^{<M>^T}TT^TY^{<M>^T} \end{cases}$$

- Since the product T^TT is independent of W, the minimization of J(W) is equivalent to maximizing J'(W)

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

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

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

  - Notice that this S_B differs from the conventional between-class covariance matrix by having N_k^2 instead of N_k.
    - This means that the MLP will have a strong bias in favor of classes that have a large number of examples

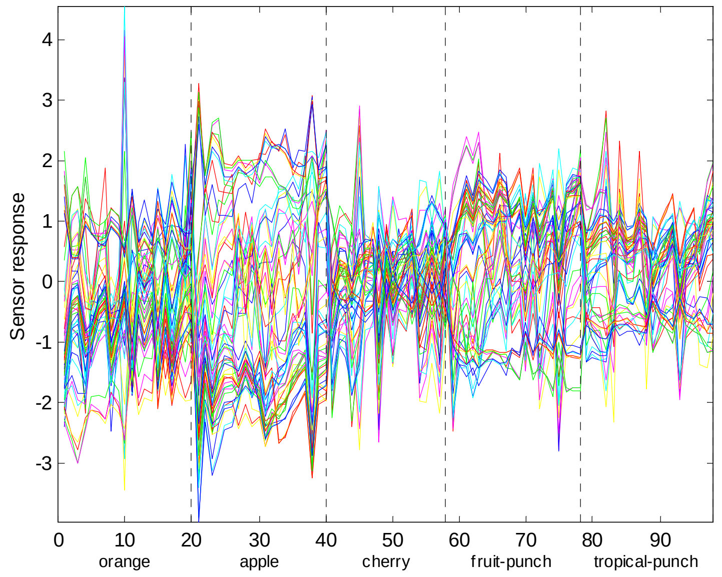
- **CONCLUSION**

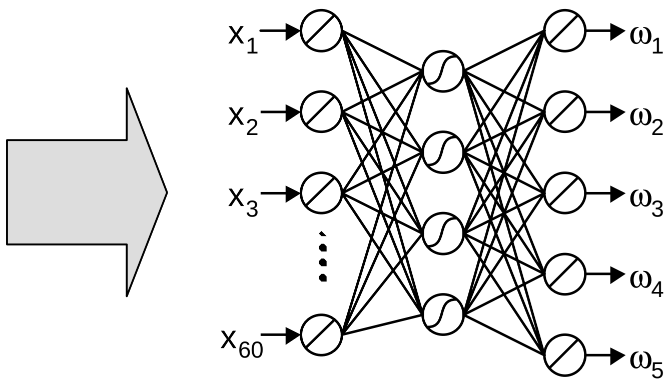
Choosing the optimum weights of a MLP to minimize the square error at the output layer forces the weights of the hidden layer(s) to be chosen so that the transformation from the input data to the output of the (last) hidden layer maximizes the discriminant function **$\mathrm{Tr}[S_BS_T^{\dagger}]$** measured at the output of the (last) hidden layer

- <u>This is precisely why MLPs have been demonstrated to perform classification tasks so well</u>

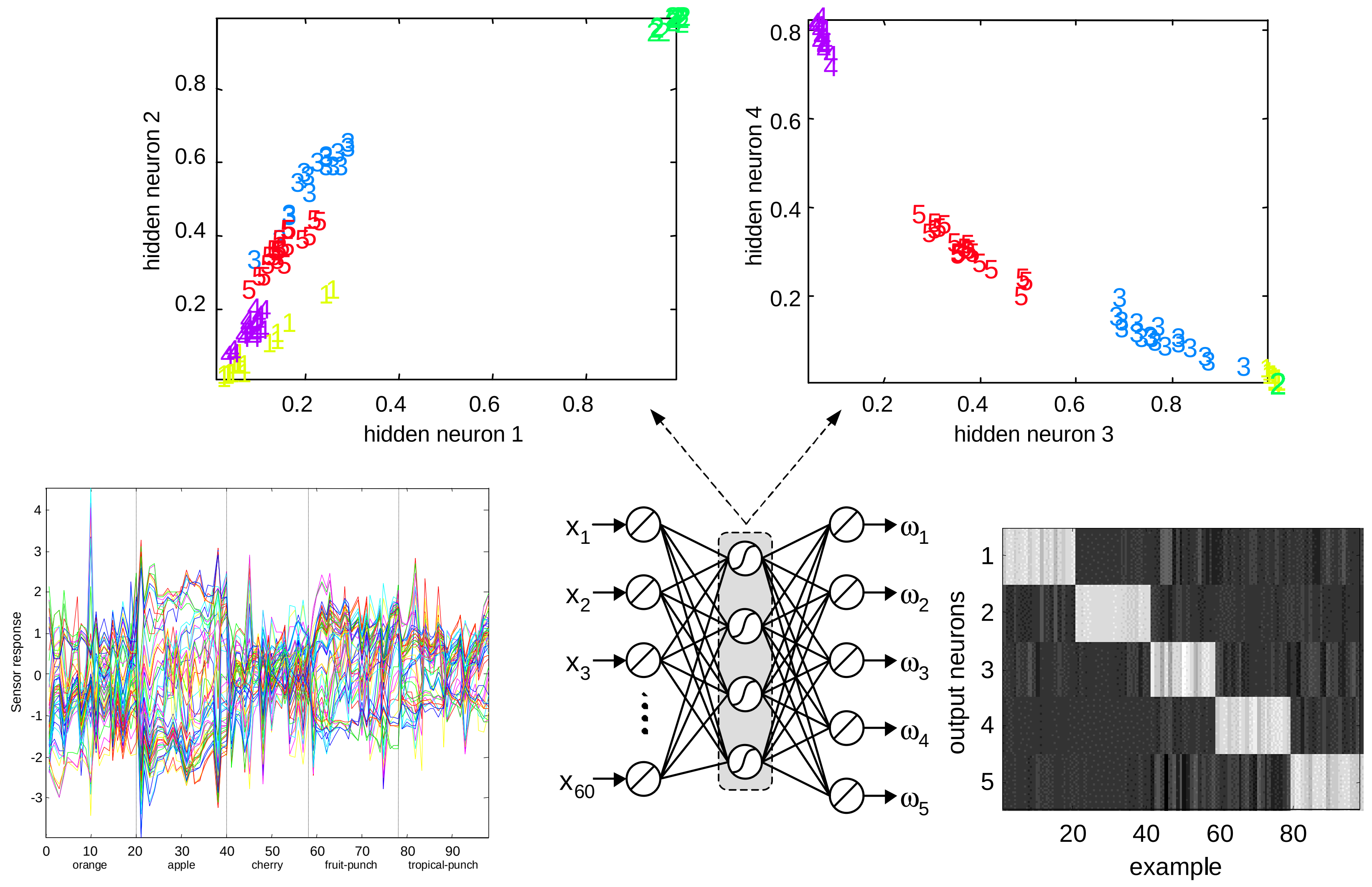
An example

- **We train a two-layer MLP to classify five odors from an array of sixty gas sensors**
  - The MLP has sixty inputs, one for each gas sensor
  - The MLP has five outputs, one for each odor
    - Output neurons use the 1-of-C encoding of classes
  - Four hidden neurons are used (as many as LDA projections)
  - The hidden layer has the logistic sigmoidal activation function
  - The output layer has linear 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


hidden neuron 2
hidden neuron 1
hidden neuron 4
hidden neuron 3
Sensor response
orange
apple
cherry
fruit-punch
tropical-punch
x_1
x_2
x_3
x_{60}
ω_1
ω_2
ω_3
ω_4
ω_5
output neurons
example