

# Lecture 13.

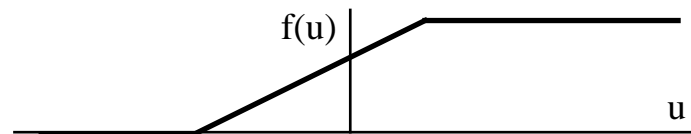
## MLP (V): Speed Up Learning

# Outline

- Dynamic range control
- Local minimum avoidance
- Encoding symbolic features
- Modular learning

# Dynamic Range Control

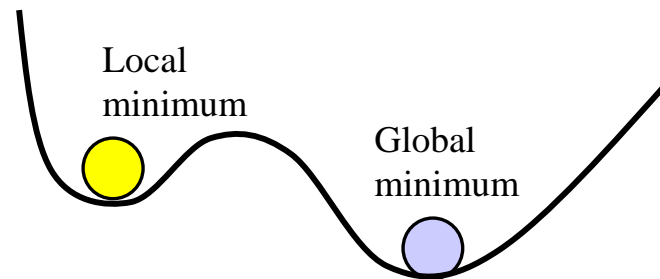
- Saturation of the nonlinear activation function



- To limit the magnitude of net-function:  $u_i = \sum_{j=1}^N w_{ij} a_j$ 
  - Initialize weights to small magnitude.
  - Scale dynamic range of inputs and outputs
  - Use  $\tanh(u)$  in hidden layer
  - Do not use too many neurons (limit N) per layer.

# Local Minimum

- Error Surface:  $E(W)$  is a nonlinear, continuously differentiable function of  $W$ .
- Gradient search will converge to a local minimum.



- However, there are often many local minima on the error surface. Dipping into an inferior local minima too early in the learning process is often undesirable!

# Tips to Avoid Local Minima

- Randomize weights initially. Reinitialize weights whenever deemed necessary.
- Randomize the order of presentation of data samples
  - Notion of "re-sampling" or boot-strapping.
- Invoke random search step when gradient search (BP) fail to reduce error function. One way is to simply use a larger step size  $\eta$ . Another way is to perturb the weights with small noise.
- Add small noise explicitly to samples, or to the net function.

# Learning Rate and Momentum

- For gradient based search,  $\eta$  should be kept small ( $< 0.1$  or even smaller). Need many more search steps.
- Initially, large  $\eta$  can be used.  $\eta$  is too large if the RMS error fluctuates violently.
- A "schedule" can be made to use different values of  $\eta$  at different stages of learning. E.g.  $\eta = A/(1+n/B)$ .
- One purpose of the momentum term with  $\mu$  is to stabilize the search direction upon convergence. Hence a larger  $\mu$  (say, 0.9) can be used.

# Epoch Size

- The number of training samples to be used per epoch certainly affect the speed of training.
- $K$  too large: Each epoch takes too long to compute.
- $K$  too small: weights are updated very frequently and error among epochs may fluctuate violently.

# Data Partitioning

- **Tuning set:**
  - a portion of training set data should be reserved as a tuning set. During training, the weights are updated using training set and the generalization error is estimated by evaluating over tuning set.
- **Cross-validation:**
  - Partition data into  $M$  portions. Use  $M-1$  portions as the training set and the remaining portion as tuning set. Then rotate so that each portion will be the tuning set once. The averaged tuning set error will be a good estimate of the generalization error.

# Stopping Criteria

- If there is tuning set, then stop when the tuning set error reaches its minimum. This requires remembering the best set of weights.
- Too much training may cause problem of over-fitting.
- Testing set should not be used during training even for checking stop conditions.

# Non-numerical Features Encoding

- **Input coding** (Converting symbolic features to numerical features)

One-in-N coding

{R, G, Y} -> {100, 010, 001}

Thermometer coding: 1 -> 0 0 0 0 1, 2 -> 0 0 0 1 1, 3 -> 0 0 1 1 1, 4 -> 0 1 1 1 1, 5 -> 1 1 1 1 1

- **Output coding**

One-class-one-output (one in N coding): e.g. for 3 classes, 3 outputs, one has 1 0 0, 0 1 0, 0 0 1

Error-correcting code (Max. Hamming distance): e.g. for 4 classes, 3 outputs: 1 0 0, 0 1 0, 0 0 1, 1 1 1

# Modular Learning

- Partition the entire network into modular piece and train them separately.

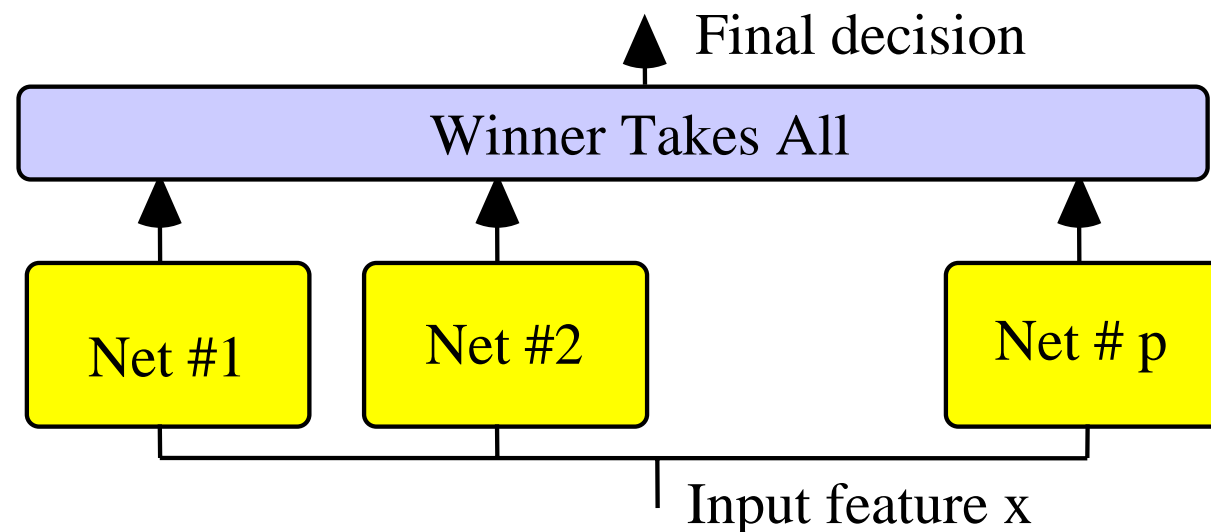
Output Partitioning – E.g. one network focuses on one output only.

Input Partitioning – e.g. one network learns features a, b, the other learns with feature c. Or, each neuron will respond to a particular subset of data samples.

Data Fusion – Glue training and fine tuning. Put all modules together, add more hidden neurons, fine tune the performance.

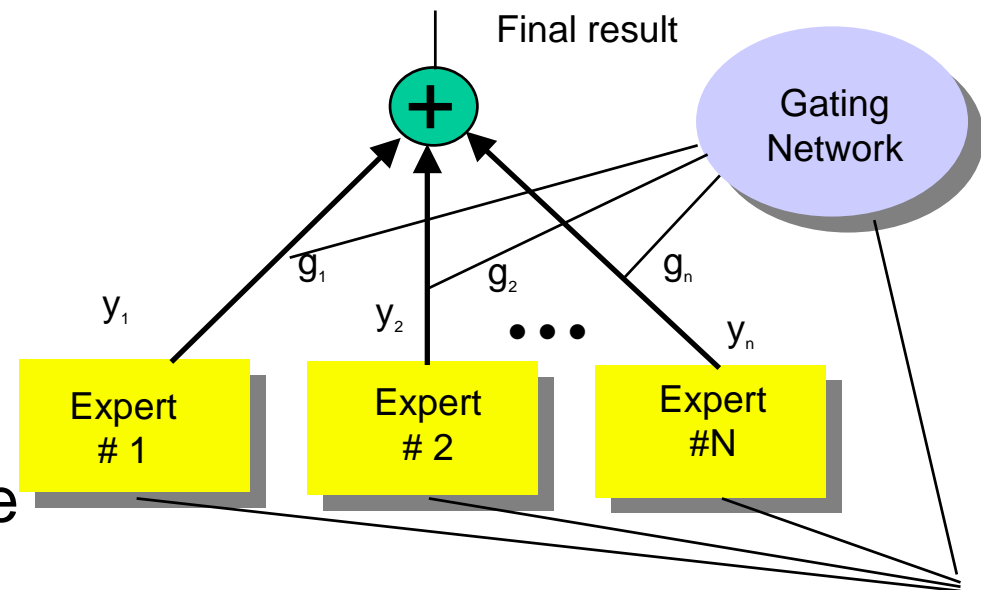
# One Output One Net

Assumption: The output is proportional to the likelihood that the feature is consistent with the class. Thus, the larger the output, the more confident the network is about the input feature.



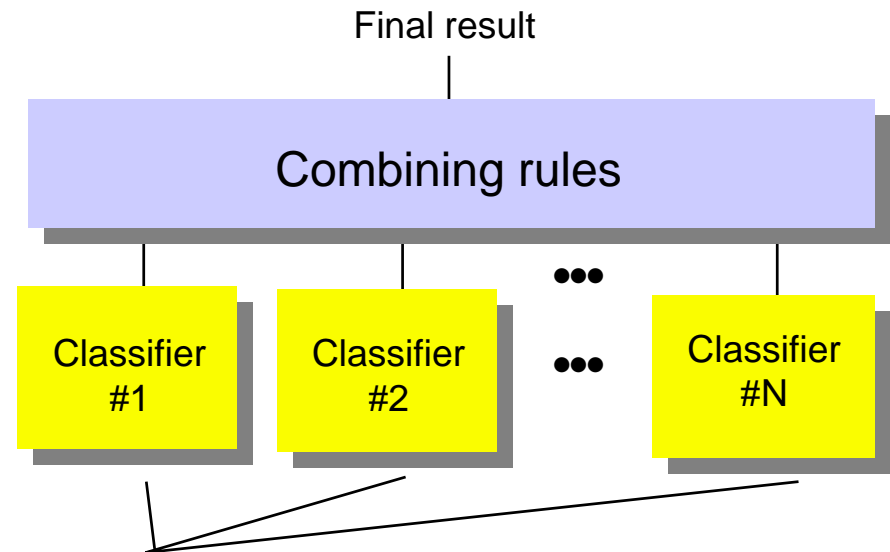
# Mixture of Experts

- Each classifier is considered as an expert in a special region of the feature space.
- A gating network determines which expert the current testing sample should be examined, and assign dynamically the weighting of expert's opinion.



# Ensemble Network

- Instead of competing to give the final output, the outputs of every member classifier are combined to form a collective opinion.
- Linear combining rule: voting, weighted ensemble averaging
- Nonlinear combining rule: Stack generalization machine



# Other Learning Methods

- BP is a steepest gradient descent optimization method.
- Other methods that can be used to find weights of a MLP include:
  - Conjugate gradient method
  - Levenburg-Marquardt method
  - Quasi-Newton method
  - Least square Kalman filtering method