Machine Learning Srihari

# Soft-weight sharing

Sargur Srihari

Machine Learning Srihari

## **Sharing of Weights**

- One way of reducing complexity of a neural network with a large number of weights is:
  - Encouraging groups of weights to have similar values
- What we saw with Conv Nets is only applicable when form of the network can be specified in advance
  - Hard weight sharing: Constraint of equal weights
- Alternative approach is Soft weight sharing:
  - Groups of weights encouraged to have similar values
- In soft-weight sharing, learning process determines
  - Division of weights into groups
  - Mean weight value for each group, and
  - Spread of values

Machine Learning Srihari

# Mixture of Gaussians for Weights

Simple weight decay regularizer for neural networks:

$$\tilde{E}(w) = E(w) + \frac{\lambda}{2} w^T w$$
 where  $\lambda$  is the regularization coefficient

- This regularizer can be interpreted as negative log of a zeromean Gaussian prior over w
  - Useful for translation invariance
- Instead, encourage weight values to form several groups rather than a single group
  - Group weights into M groups, each with a Gaussian prior distribution over the weights
    - A mixture distribution
    - Centers, variances and mixing coefficients are learnt from data

#### Machine Learning Srihari

### Learning weight groups

Probability density of weight groups

$$p(\mathbf{w}) = \prod_{i} p(w_i)$$
 
$$p(w_i) = \sum_{j=1}^{M} \pi_j \mathcal{N}(w_i | \mu_j, \sigma_j^2)$$

Negative log leads to regularization function

$$\Omega(\mathbf{w}) = -\sum_{i} \ln \left( \sum_{j=1}^{M} \pi_{j} \mathcal{N}(w_{i} | \mu_{j}, \sigma_{j}^{2}) \right)$$

- Total error function  $\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \Omega(\mathbf{w})$
- Error minimized wrt weights  $w_i$  and wrt parameters  $\{\pi_j, \mu_j, \sigma_j\}$
- Optimization method
  - If  $w_i$  were fixed, we could use EM algorithm, but they are not
    - If  $w_k$  were fixed this is a standard mixture of Gaussians problem
  - So use conjugate gradients or quasi Newton optimization
    - Conjugate gradients is faster than SGD but is sensitive

Machine Learning Srihari

### Derivatives for Minimization of total error function

$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \Omega(\mathbf{w})$$

$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda \Omega(\mathbf{w})$$
  $\Omega(\mathbf{w}) = -\sum_{i} \ln \left( \sum_{j=1}^{M} \pi_{j} \mathcal{N}(w_{i} | \mu_{j}, \sigma_{j}^{2}) \right)$ 

Define

$$\left| \boldsymbol{\gamma_{\scriptscriptstyle j}} = \frac{\boldsymbol{\pi_{\scriptscriptstyle j}} \boldsymbol{N} \left( \boldsymbol{w} \,|\, \boldsymbol{\mu_{\scriptscriptstyle j}}, \boldsymbol{\sigma_{\scriptscriptstyle j}^2} \right)}{\sum_{\boldsymbol{k}} \boldsymbol{\pi_{\scriptscriptstyle k}} \boldsymbol{N} \left( \boldsymbol{w} \,|\, \boldsymbol{\mu_{\scriptscriptstyle k}}, \boldsymbol{\sigma_{\scriptscriptstyle k}^2} \right)} \right|$$

Derivatives of total error function wrt weights  $w_i$  is given by

$$\boxed{\frac{\partial \tilde{E}}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda \sum_j \gamma_j \Big(w_i\Big) \frac{(w_i - \mu_j)}{\sigma_j^2}}$$

• Derivatives of error wrt the centers of Gaussians  $\mu_i$ 

$$\boxed{\frac{\partial \tilde{E}}{\partial \mu_{_{j}}} = \lambda {\sum_{_{j}}} \, \gamma_{_{j}} \Big(w_{_{i}}\Big) \frac{(\mu_{_{i}} - w_{_{j}})}{\sigma_{_{j}}^{2}}}$$

Derivatives of error wrt the variances  $\sigma_i$ 

$$\left| \frac{\partial \tilde{E}}{\partial \sigma_j} = \lambda \sum_j \gamma_j \Big( w_i \Big) \!\! \left[ \frac{1}{\sigma_j} - \! \frac{\left( w_i - \mu_j \right)^2}{\sigma_j^2} \right] \right|$$

Derivatives of error wrt mixing coefficients

$$\boxed{\frac{\partial \tilde{E}}{\partial \eta_{_{j}}} = \lambda {\sum_{_{i}}} \Big\{ \pi_{_{j}} - \gamma_{_{j}} \Big( w_{_{i}} \Big) \hspace{0.1cm} \Big\}} \hspace{1cm} \text{where}$$

$$\pi_{_{j}} = \frac{\exp\left(\eta_{_{j}}\right)}{\sum_{_{k=1}^{M}}^{^{M}} \exp\left(\eta_{_{k}}\right)}$$

5