Regularization in Neural Networks

1. Bayesian Interpretation of Regularization (Extension of Gaussian Priors)

$\widehat{E} = E_D(w) + \lambda E_w(w)$

where $E_D(w)$ is the error of misclassification on the training set and $E_w(w)$ is the penalty form. Interpreted in the Bayesian way:

$P(w \mid D) \propto P(D \mid w)P(w)$

considering:

$P(D \mid w) = Z_D^{-1} \exp(-E_D)$ and $P(w) = Z_W^{-1} \exp(-\lambda E_W)$

It follows that the process of minimization is equivalent to

$\widehat{E} = -\log P(D \mid w)P(w) + const$

2. Approaches to Invariance

Let the transformation denoted by $s(x_n, \xi)$, which is parameterized by ξ. $s(x,0) = x$. The tangent vector at the point x_n is given by:

$$\tau_n = \frac{\partial s(x_n, \xi)}{\partial \xi} \Big|_{\zeta=0} \tag{5.125}$$

2.1 Regularization through Tangent Propagation

If the input vector x is transformed with parameter ξ, in general the network output will change. The derivative of network output with respect to ξ is given by:

$$\frac{\partial y_k}{\partial \xi} \Big|_{\zeta=0} = \sum \frac{\partial y_k}{\partial x_i} \frac{\partial x_i}{\partial \xi} \Big|_{\zeta=0} = \sum J_{ki} \tau_i \tag{5.126}$$

By adding this part into the error function to penalized variance to the transform, ***the total error function*** is modified to be:

$$\widehat{E} = E_D(w) + \lambda \Omega \tag{5.127}$$

where

$$\Omega = \frac{1}{2} \sum_n \sum_k \left(\frac{\partial y_{nk}}{\partial \xi} \Big|_{\xi=0} \right)^2 = \frac{1}{2} \sum_n \sum_k \left(\sum_{i=1:D} J_{ki} \tau_i \right)^2 \tag{5.128}$$

2.2 Training with transformed data

By considering the average error function under a distribution $p(\xi)$ of the transformation parameter ξ, ***the average error function*** becomes:

$$\widehat{E} = \frac{1}{2} \iiint \{ y(s(x,\xi) - t \}^2 \, p(t \mid x) \, p(x) \, p(\xi) dx dt d\xi \tag{5.130}$$

By taking the Taylor expansion of y with respect to x and ξ, in the average error function

$$\widehat{E} = E_D + \lambda\Omega \tag{5.131}$$

$$\Omega = \int \left[\{y(x) - E[t \mid x]\} \frac{1}{2} \{(\tau')^T \nabla y(x) + \tau^T \nabla\nabla y(x)\tau\} + (\tau^T \nabla y(x))^2 \right] p(x)dx \tag{5.132}$$

2.3 Convolutional Networks

The convolutional network is composed of (1) receptive layer, (2) convolutional layer and (3) sub-sampling layer.

Weight sharing: in the convolutional layer all the units in a feature map are constrained to share the same weight.

Subsampling: each unit takes input from a small receptive field in the corresponding feature map of the convolutional layer. The average value of those input are calculated to be the input of a sigmoidal function.

2.4 Soft Weight Sharing

Soft weight sharing is designed to replace the hard constraint of equal weight in the convolutional network with a form of regularization.

Instead of considering a single Gaussian prior distribution of the weight, in the soft weight sharing the distribution is taken as a mixture of Gaussians:

$$p(w) = \prod_i p(w_i) \tag{5.136}$$

$$p(w_i) = \sum_{j=1:M} \pi_j N(w_i \mid \mu_j, \sigma_j^2) \tag{5.138}$$

So the regularization function takes the form:

$$\Omega(w) = -\sum_i \ln\left(\sum_{j=1}^{M} \pi_j N(w_i \mid \mu_j, \sigma_j^2)\right) \tag{5.139}$$

The minimization of the total error function

$$\widehat{E} = E_D(w) + \lambda\Omega$$

is performed with the help of considering the mixing coefficient $\{\pi_j\}$ as prior probabilities, so the constraint:

$$\sum_j \pi_j = 1 \tag{5.145}$$

is added.