NEURAL NETWORKS

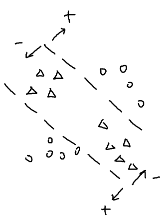
2009年11月3日
上午 12:53

Previously

Linear combinations of fixed basis functions

pros : analytic and computational convenience
cons : curse of dimensionality
hard to apply to large-scale problems

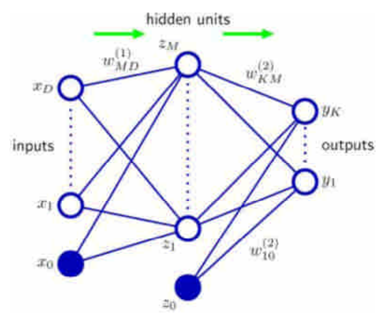
Altenatives : ① Support Vector Machines (SVMs)
data points → basis functions
convex objective functions



② Neural Networks
fixed number of basis functions in parametric forms
adapt the parameters during training

We will focus on
feed-forward neural networks
a.k.a. multilayer perceptrons

training algorithm :
error – backpropagation



$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \qquad j = 1, \dots, M$$

$$z_j = h(a_j)$$

↑ hidden units

$h(\cdot)$: nonlinear activation function
sigmoid or tanh

2009年11月3日
上午 01:48

$$\tilde{a}_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \qquad k = 1, \dots, K$$

$$y_k = \sigma(\tilde{a}_k) \qquad \sigma(a) = \frac{1}{1+\exp(-a)}$$

output units

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

two-layer

define additional units $x_0 = 1$, $z_0 = 1$

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right)$$

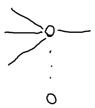
Properties :
① differentiable
② if the activation functions of all the hidden units are linear → we can find an equivalent network without hidden units
③ if the number of hidden units is smaller than the number of input units or output units → dimensionality reduction

④ A two-layer network with linear outputs can uniformly approximate any continuous function on a compact input domain to arbitrary accuracy provided that the network has a sufficiently large number of hidden units

Fig. 5.3

2009年11月3日
上午 02:24

Weight-space symmetries



2^M combinations of sign-flips

$M!$ orderings of hidden units

$M!\,2^M$ factors of symmetries

ERROR BACKPROPAGATION (BACKPROP)

a local message passing scheme

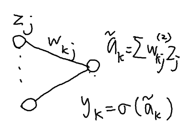
① derivatives of the error function w.r.t. the weights need to be evaluated
(backprop : an efficient way to evaluate the derivatives)

② the derivatives are used to adjust the weights

Evaluation of error-function derivatives :

Consider a simple network having a single layer of sigmoidal hidden units with a cross-entropy error

$$E(w) = \sum_{n=1}^{N} E_n(w)$$ N data points



$$y_k = \sigma(\tilde{a}_k) = \sigma\left(\sum_{j=0}^{M} w^{(2)}_{kj} z_j\right)$$

$$E_n = -\sum_{k=1}^{K} \{ t_{nk} \ln y_{nk} + (1-t_{nk}) \ln (1-y_{nk}) \}$$

$y_{nk} = y_k(x_n, w)$

2009年11月3日
上午 02:46

$$\frac{\partial E_n}{\partial w^{(2)}_{kj}} = \underbrace{\frac{\partial E_n}{\partial \tilde{a}_k}}_{\delta_k} \underbrace{\frac{\partial \tilde{a}_k}{\partial w^{(2)}_{kj}}}_{z_{nj}} = \delta_k \cdot z_{nj}$$

$$\delta_k = \frac{\partial E_n}{\partial \tilde{a}_k} = -\left(\frac{t_{nk}}{y_{nk}} \frac{\partial y_{nk}}{\partial \tilde{a}_k} - \frac{1-t_{nk}}{1-y_{nk}} \frac{\partial y_{nk}}{\partial \tilde{a}_k} \right)$$

$$= -\left(t_{nk}(1-y_{nk}) - (1-t_{nk}) y_{nk} \right)$$

$$= y_{nk} - t_{nk}$$

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

$$\frac{\partial \sigma}{\partial a} = \frac{1}{(1+e^{-a})^2} \cdot e^{-a} = \sigma(1-\sigma)$$

$$\frac{\partial y_{nk}}{\partial \tilde{a}_k} = y_{nk}(1-y_{nk})$$

$$\frac{\partial E_n}{\partial w^{(1)}_{ji}} = \underbrace{\frac{\partial E_n}{\partial a_j}}_{\delta_j} \underbrace{\frac{\partial a_j}{\partial w^{(1)}_{ji}}}_{x_{ni}}$$

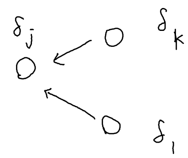
$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_{k=1}^{K} \underbrace{\frac{\partial E_n}{\partial \tilde{a}_k}}_{\delta_k} \frac{\partial \tilde{a}_k}{\partial a_j}$$

$$= h'(a_j) \sum_{k=1}^{K} w^{(2)}_{kj} \delta_k$$

$$\tilde{a}_k = \sum_{j=0}^{M} w^{(2)}_{kj} z_{nj} = \sum_{j=0}^{M} w^{(2)}_{kj} h(a_j)$$

$$\frac{\partial \tilde{a}_k}{\partial a_j} = w^{(2)}_{kj} h'(a_j)$$

2009年11月3日
上午 03:15



① $a_j = \sum_i w_{ji}^{(1)} x_{ni}$

$z_{nj} = h(a_j)$

$\tilde{a}_k = \sum_k w_{kj}^{(2)} z_{nj}$

$y_{nk} = \sigma(\tilde{a}_k)$

② $\delta_k = y_{nk} - t_{nk}$

$$\boxed{\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k \cdot z_{nj}}$$

③ $\delta_j = h'(a_j) \sum_k w_{kj}^{(2)} \delta_k$

$$\boxed{\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j \, x_{ni}}$$

THE HESSIAN MATRIX $\frac{\partial^2 E}{\partial w_{ji} \partial w_{kl}}$

2009年11月3日
上午 03:26

① second-order properties of the error surface
② fast retraining a feed-forward network following a small change in the training data
③ inverse of the Hessian
→ least significant weights
④ Laplace approximation for a Bayesian neural network

How to compute the Hessian ?

approximately or exactly

(outer product, finite difference) (backprop)

Outer Product Approximation

Sum-of-squares error function $E = \frac{1}{2} \sum_{n=1}^{N} (y_n - t_n)^2$

the Hessian matrix

$$H = \nabla\nabla E = \sum_{n=1}^{N} \nabla y_n (\nabla y_n)^T + \sum_{n=1}^{N} (y_n - t_n) \nabla\nabla y_n$$

$\nabla E = \sum_{n=1}^{N} (y_n - t_n) \nabla y_n$

Assume the network has been trained on the data set : $y_n \simeq t_n$, we may neglect the second term
or
$(y_n - t_n)$ is a random variable zero mean and $(y_n - t_n)$ is uncorrelated with $\nabla\nabla y_n$, $\sum_{n=1}^{N} (y_n - t_n) \nabla\nabla y_n$ will sum to zero .

2009年11月3日
上午 09:32

outer product approximation (Levenberg-Marquardt approximation)

$$H \simeq \sum_{n=1}^{N} \mathbf{b}_n \mathbf{b}_n^T$$

$\mathbf{b}_n \equiv \nabla a_n = \nabla y_n$

assume the output units are linear : $y_n = a_n$

This approximation is only likely to be valid for a well trained network.

the cross-entropy error function for a network with logistic sigmoid output-unit activation functions

$$H \simeq \sum_{n=1}^{N} y_n(1-y_n)\, \mathbf{b}_n \mathbf{b}_n^T$$

$$\mathbf{b}_n \equiv \nabla_w a_n$$

$$\nabla E = \sum_{n=1}^{N} \frac{\partial E}{\partial a_n} \nabla a_n = \sum_{n=1}^{N} (y_n - t_n) \nabla a_n$$

$$\nabla\nabla E = \sum_{n=1}^{N} \left\{ \frac{\partial y_n}{\partial a_n} \nabla a_n (\nabla a_n)^T + (y_n - t_n) \nabla\nabla a_n \right\} \simeq \sum_{n=1}^{N} y_n(1-y_n) \nabla a_n (\nabla a_n)^T$$

Compute the inverse of the Hessian

$$H_N = \sum_{n=1}^{N} \mathbf{b}_n \mathbf{b}_n^T \quad \Rightarrow \quad H_{L+1} = H_L + \mathbf{b}_{L+1} \mathbf{b}_{L+1}^T$$

$$(M + vv^T)^{-1} = M^{-1} - \frac{(M^{-1}v)(v^T M^{-1})}{1 + v^T M^{-1} v} \qquad \text{(rank-one update)}$$

$$H_{L+1}^{-1} = H_L^{-1} - \frac{H_L^{-1} \mathbf{b}_{L+1} \mathbf{b}_{L+1}^T H_L^{-1}}{1 + \mathbf{b}_{L+1}^T H_L^{-1} \mathbf{b}_{L+1}}$$

$H_0 = \alpha I$

α is small

2009年11月3日
上午 10:47

Finite Differences Approximation

$$\frac{\partial^2 E}{\partial w_{ji} \partial w_{lk}} = \frac{1}{2\epsilon} \left\{ \frac{\partial E}{\partial w_{ji}}(w_{lk} + \epsilon) - \frac{\partial E}{\partial w_{ji}}(w_{lk} - \epsilon) \right\} + O(\epsilon^2)$$

central differences of the first derivatives

$\frac{\partial E}{\partial w_{ji}}$ can be computed by backprop

Exact Evaluation of the Hessian

using backpropagation

Consider a two-layer network

i, i' for input units , j, j' for hidden units

k, k' for output units

$$\delta_k = \frac{\partial E_n}{\partial \tilde{a}_k} , \qquad M_{kk'} \equiv \frac{\partial^2 E_n}{\partial \tilde{a}_k \partial \tilde{a}_{k'}}$$

① $$\frac{\partial^2 E_n}{\partial w_{kj}^{(2)} \partial w_{k'j'}^{(2)}} = z_j z_{j'} M_{kk'}$$

$$\frac{\partial\, \delta_k z_j}{\partial w_{k'j'}^{(2)}} = z_j \frac{\partial E_n}{\partial \tilde{a}_k \partial \tilde{a}_{k'}} \frac{\partial \tilde{a}_{k'}}{\partial w_{k'j'}} = z_j z_{j'} M_{kk'}$$

② $$\frac{\partial^2 E_n}{\partial w_{ji}^{(1)} \partial w_{j'i'}^{(1)}} = x_i x_{i'} h''(a_j) I_{jj'} \sum_k w_{kj'}^{(2)} \delta_k + x_i x_{i'} h'(a_{j'}) \sum_k \sum_{k'} w_{kj}^{(2)} w_{k'j'}^{(2)} M_{kk'}$$

③ $$\frac{\partial^2 E_n}{\partial w_{ji}^{(1)} \partial w_{kj'}^{(2)}} = x_i h'(a_{j'}) \left\{ \delta_k I_{jj'} + z_j \sum_{k'} w_{k'j'}^{(2)} M_{kk'} \right\}$$

Fast Multiplication by the Hessian

2009年11月3日
上午 11:09

In some cases we wish to compute $v^T H$ rather than H

$$v^T H = \underbrace{v^T \nabla}_{} (\nabla E)$$

as a new operator $\mathcal{R}\{\cdot\}$

$$\mathcal{R}\{w\} = v^T \nabla w = \nabla v^T w = \frac{\partial}{\partial w} w^T v = v$$

Consider cross-entropy error with sigmoidal outputs

$$a_j = \sum_i w_{ji}^{(1)} x_i \qquad \mathcal{R}\{a_j\} = \sum_i v_{ji} x_i$$

$$z_j = h(a_j) \qquad \mathcal{R}\{z_j\} = h'(a_j)\mathcal{R}\{a_j\}$$

$$\tilde{a}_k = \sum_j w_{kj}^{(2)} z_j \qquad \mathcal{R}\{\tilde{a}_k\} = \sum_j w_{kj}^{(2)} \mathcal{R}\{z_j\} + \sum_j v_{kj} z_j$$

$$y_k = \sigma(\tilde{a}_k) \qquad \mathcal{R}\{y_k\} = \sigma'(\tilde{a}_k)\mathcal{R}\{\tilde{a}_k\}$$

$$\delta_k = y_k - t_k \qquad \mathcal{R}\{\delta_k\} = \mathcal{R}\{y_k\}$$

$$\delta_j = h'(a_j)\sum_k w_{kj}^{(2)} \delta_k \qquad \mathcal{R}\{\delta_j\} = h''(a_j)\mathcal{R}\{a_j\}\sum_k w_{kj}^{(2)}\delta_k + h'(a_j)\sum_k v_{kj}\delta_k + h'(a_j)\sum_k w_{kj}^{(2)}\mathcal{R}\{\delta_k\}$$

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \delta_k z_j \qquad \mathcal{R}\left\{\frac{\partial E}{\partial w_{kj}^{(2)}}\right\} = \mathcal{R}\{\delta_k\} z_j + \delta_k \mathcal{R}\{z_j\}$$

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \delta_j x_i \qquad \mathcal{R}\left\{\frac{\partial E}{\partial w_{ji}^{(1)}}\right\} = x_i \mathcal{R}\{\delta_j\}$$

(Set $v^T = (0, 0, \dots, 1, \dots, 0)$ for evaluating full Hessian)

BAYESIAN NEURAL NETWORKS

2009年11月3日
上午 11:33

Using Laplace approximation
approximate the posterior by a Gaussian centered
at a mode of the true posterior

Posterior Parameter Distribution

(Regression)

$$p(w \mid D, \alpha, \beta) \propto \underbrace{p(w \mid \alpha)}_{\text{prior}} \; \underbrace{p(D \mid w, \beta)}_{\text{likelihood}}$$

$$p(w \mid \alpha) = \mathcal{N}(w \mid 0, \alpha^{-1} I)$$

$$p(D \mid w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n \mid y(x_n, w), \beta^{-1})$$

$$\ln p(w \mid D) = -\frac{\alpha}{2} w^T w - \frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2 + \text{const}$$

regularized sum-of-squares error function

mode : w_{MAP}

$$A = -\nabla\nabla \ln p(w \mid D, \alpha, \beta) = \alpha I + \beta H$$

(From Taylor expansion)

H is the Hessian matrix comprising the second derivatives of the sum-of-squares error function (we know how to evaluate the Hessian)

$$q(w \mid D) = \mathcal{N}(w \mid w_{MAP}, A^{-1})$$

2009年11月3日
上午 11:53

Predictive Distribution

$$p(t|x,D) = \int p(t|x,w)\, q(w|D)\, dw$$

marginalize w.r.t. the posterior

Make a Tayor expansion of the network function around w_{MAP}

$$y(x,w) \simeq y(x, w_{MAP}) + g^T (w - w_{MAP})$$

$$g = \nabla_w y(x,w) \Big|_{w = w_{MAP}}$$ (linear approximation

$$p(t|x,w,\beta) \simeq \mathcal{N}\left(t \middle| y(x,w_{MAP}) + g^T(w - w_{MAP}),\ \beta^{-1}\right)$$

Finally the predictive distribution is approximated by

$$p(t|x,D,\alpha,\beta) = \mathcal{N}\left(t \middle| y(x, w_{MAP}),\ \sigma^2(x)\right)$$

$$\sigma^2(x) = \beta^{-1} + g^T A^{-1} g$$

β^{-1}: intrinsic noise

$g^T A^{-1} g$: uncertainty in w

See (2.113)–(2.117)

2009年11月3日
下午 12:04

Bayesian Neural Networks for Classification

$$\ln p(D|w) = \sum_{n=1}^{N} \{ t_n \ln y_n + (1 - t_n) \ln (1 - y_n) \}$$

log-likelihood $\quad t_n \in \{0, 1\} \quad y_n \equiv y(x_n, w)$

there is no hyperparameter β because the data points are assumed to be correctly labeled

Apply Laplace approximation:

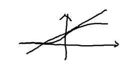
maximizing the log posterior = minimizing the regularized error

$$E(w) = -\ln p(D|w) + \frac{\alpha}{2} w^T w$$

mode: w_{MAP}

evaluating the Hessian H (second derivatives of the negative log likelihood)

Predictive distribution:



linear approximation for logistic sigmoid outputs would be inappropriate

⇒ make a linear approximation for the output unit activation

$$a(x,w) \simeq a_{MAP}(x) + b^T (w - w_{MAP})$$

$$a_{MAP}(x) = a(x, w_{MAP}) \qquad b \equiv \nabla a(x, w_{MAP})$$

by backprop

See section 4.5.2

$$p(a|x,D) = \int \delta\left(a - a_{MAP}(x) - b^T(x)(w - w_{MAP})\right) q(w|D)\, dw$$

Gaussian $\quad$ mean: $a_{MAP} \equiv a(x, w_{MAP})$ $\quad$ variance: $s_a^2(x) = b^T(x) A^{-1} b(x)$

2009年11月3日
下午 12:44

$$p(t=1 \mid \mathbf{x}, D) = \int \sigma(a)\, p(a \mid \mathbf{x}, D)\, da$$

$$= \sigma\left(\kappa(s_a^2)\, b^T w_{MAP} \right)$$

$$\kappa(s^2) = (1 + \pi s^2/8)^{-1/2}$$

b, s_a^2 are functions of $\mathbf{x}$

Bayesian neural networks outperform boosted trees and random forests
NIPS 2003 challenge (two-class classification problems)
Neal & Zhang

Geoffrey Hinton (U. Toronto) Restricted Boltzmann Machines
Yann LeCun (NYU) Convolutional Networks