PRML 5. Neural Networks

To adapt basis function to data.

- SVM : Select basis functions during training. ··· Convex.
- RVM : 〃 ··· sparser but non convex.
- NN : Fixed number, adapting during training. ··· More compact. non convex.

§ 5.1

Feed-forward neural network

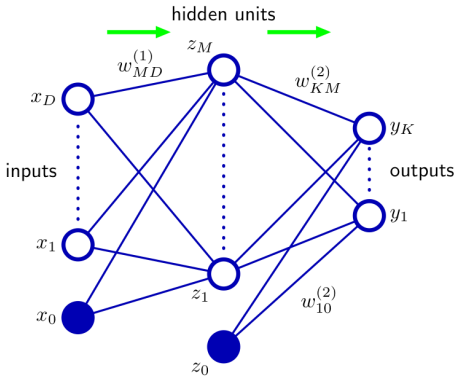
$$y_k(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right), \quad a_j = \sum_{i=0}^{D} w_{ji}^{(1)} x_i .$$

($a^{(1)}$ = the whole sum; $z^{(1)}$ = $h(a_j)$; activation = a_j)

- $f(x) = x$ for regression
- $\sigma(x)$ for classification (or softmax)
- h: non-linear activation function

※ FNN with linear activation function
≡ linear transformation
(possibly with dimensionality reduction)

※ FNN may have skip-layers.
··· sigmoidal hidden units can mimic them, but including them can be better in practice.



$\mathbf{x} \xrightarrow{\mathbf{W}^{(1)}} a^{(1)} \xrightarrow{h} z^{(1)} \xrightarrow{\mathbf{W}^{(2)}} a^{(2)} \xrightarrow{\text{id.}} \text{output}$
(σ for classification)
bias, bias

- Two-layer FNN (except polynomial activation functions)
  can uniformly approximate any continuous functions over a compact region.

- M hidden units have $2^M \times M!$-wise symmetry. $(+ \leftrightarrow -,\ i \leftrightarrow j)$
  → at least $2^M M!$ global minima.

§ 5.2 Natural choice of F and h

$$p(\mathbf{t}\mid \mathbf{X}, w, \beta) = \prod_{n=1}^{N} p(t_n \mid x_n, w, \beta)$$

ML approach: $\nabla_w E(w) = -\sum \nabla_w \ln p_n = 0$

$y_k = F(a_k^{(2)})$

$a_k^{(2)} = \sum w_{kj}^{(2)} z_j^{(1)}$

$z_j^{(1)} = h(a_j^{(1)})$

$a_j^{(1)} = \sum_{i=0} w_{ji}^{(1)} x_i.$

- Regression with Gaussian-noised data $p(t\mid x, w) = \mathcal{N}(t \mid y(x,w), \beta^{-1})$

  $-\nabla_w \ln p = \beta(y-t)\nabla_w y$

- Classification $p(t\mid x, w) = y^t (1-y)^{1-t}$

  $-\nabla_w \ln p = \dfrac{y-t}{y(1-y)} \nabla_w y$

- Multi-class classification $p(t_k \mid x, w) = \prod y_k^{t_k}$

  $-\nabla_w \ln p = -\sum_k t_k \nabla_w \ln y_k$

$$\nabla_w y = \frac{\partial F}{\partial a^{(2)}} \nabla_w a^{(2)}$$

$\sim \dfrac{\partial E}{\partial a_k}$

Regression: $F = \mathrm{id}$ $\rightarrow -\nabla_w \ln p = \beta(y-t)\nabla_w a^{(2)}$

Classification: $F = \sigma \rightarrow F' = F(1-F)$ $\rightarrow -\nabla_w \ln p = (y-t)\nabla_w a^{(2)}$

Multi-class: $F_k = \mathrm{softmax}$ $\rightarrow -\nabla_w \ln p = \sum_k (y_k - t_k)\nabla_w a_k^{(2)}$

… Choosing a suitable F gives $\dfrac{\partial E}{\partial a_k} \propto y_k - t_k$.

$\dfrac{\partial}{\partial a_j} \ln F_k = \delta_{jk} - F_j$

$$-\nabla_w \ln p = -\sum_k t_k \sum_j \left(\frac{\partial}{\partial a_j^{(2)}} \ln y_k\right) \nabla_w a_j^{(2)}$$

$$= \sum_{j,k} t_k (y_j - \delta_{jk}) \nabla_w a_j^{(2)}$$

$$= \sum_j \Big[\underbrace{\sum_k t_k}_{=1} y_j - t_j\Big] \nabla_w a_j^{(2)}$$

✡ Gradient method

https://www.slideshare.net/t_koshikawa/prml-5-pp227pp247
…but a few mistakes.

- global convergence … converge to a solution regardless of the starting point

- convergence rate

p-th order convergence $\Leftrightarrow \|x_{k+1} - x^*\| \le C \|x_k - x^*\|^p$. $(C<1)$

super-linear convergence $\Leftrightarrow \|x_{k+1} - x^*\| \le C_k \|x_k - x^*\|$

with $\lim_{k\to\infty} C_k = 0$.

linear << superlinear << quadratic.

- gradient descent

$w := w - \mu \nabla E(w)$ … global, linear

← difficult to tune

- Newton-Raphson

$w := w - \alpha^{(\tau)} H^{-1} \nabla E(w)$ … non-global, quadratic (!)

any parameter ($\exists 1$, or determined by linear search.) No hyperparameter

H^{-1}: tough, unstable

- Quasi-Newton … global, super-linear

$w^{(\tau+1)} = w^{(\tau)} - \alpha^{(\tau)} B^{(\tau)^{-1}} \nabla E(w)$ with $B^{(\tau)} \simeq H$,

where $s = w^{(\tau)} - w^{(\tau-1)}$, $y = \nabla E(w^{(\tau)}) - \nabla E(w^{(\tau-1)})$, $\rho = (y^T s)^{-1}$,

$$B := B - \frac{B s s^T B}{s^T B s} + \rho y y^T$$

$$\left[\text{or } B^{-1} := (I - \rho s y^T) B^{-1} (I - \rho y s^T) + \rho s s^T \right] \quad \text{(BFGS)}$$

B^{-1}: stable!

- Conjugate descent.

§ 5.3 Back propagation to evaluate $\nabla E(w)$.

$y_k = F(a_k^{(2)})$

$a_k^{(2)} = \sum w_{kj}^{(2)} z_j^{(1)}$

$z_j^{(1)} = h(a_j^{(1)})$

$a_j^{(1)} = \sum_{i=0} w_{ji}^{(1)} x_i$.

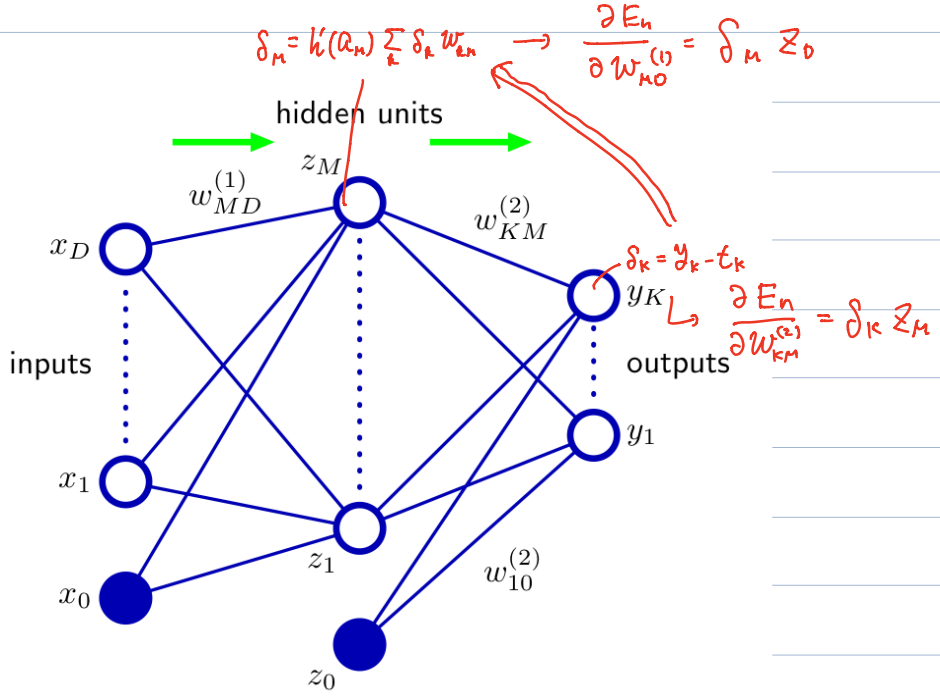
Define $\delta_j^{(m)} \equiv \frac{\partial E_n}{\partial a_j^{(m)}}$ where $E(w) = \sum_n E_n(w)$.

$$\frac{\partial E_n}{\partial w_{ji}^{(m)}} = \frac{\partial E_n}{\partial a_j^{(m)}} \frac{\partial a_j^{(m)}}{\partial w_{ji}^{(m)}} = \delta_j^{(m)} z_i^{(m-1)}.$$

For output units, $\delta_j^{(out)} \equiv \frac{\partial E_n}{\partial a_j^{(out)}} = y_j - t_j$

while for hidden units, $\delta_j^{(m)} \equiv \frac{\partial E_n}{\partial a_j^{(m)}} = \sum_k \frac{\partial E_n}{\partial a_k^{(m+1)}} \frac{\partial a_k^{(m+1)}}{\partial z_j^{(m)}} \frac{\partial z_j^{(m)}}{\partial a_j^{(m)}}$

$$= \sum_k \delta_k^{(m+1)} w_{kj}^{(m+1)} h'(a_j^{(m)}).$$



※ Numerical differenciation of $\frac{\partial E}{\partial w_{ji}}$ is slower than Back propagation but useful to debug.

§ 5.4 Hessian

$$H_{(ji)(lk)} = \frac{\partial^2 E}{\partial w_{ji}\, \partial w_{lk}}$$

§ 5.4.1 Diagonal approximation

Useful to calculate H^{-1}, but not good approximation.

$$\frac{\partial^2 E_n}{\partial w_{ji}^2} = \frac{\partial^2 E_n}{\partial a_j^2} z_i^2 ;$$

$$\frac{\partial^2 E_n}{\partial a_j^2} = \frac{\partial}{\partial a_j}\left(\sum_k \frac{\partial E_n}{\partial a_k} w_{kj}\, h'(a_j) \right)$$

($\frac{\partial^2 E_n}{\partial a_j^2}$: m-th layer; $\frac{\partial E_n}{\partial a_k}$: NEXT LAYER; w_{kj}: CONNECT to NEXT LAYER)

$$= \sum_k h''(a_j)\, w_{kj} \frac{\partial E_n}{\partial a_k} + h'(a_j)^2 \sum_{k,k'} w_{kj} w_{k'j} \frac{\partial^2 E_n}{\partial a_k \partial a_{k'}}$$

($\frac{\partial E_n}{\partial a_k}$: $m+1$-th; $\frac{\partial^2 E_n}{\partial a_k \partial a_{k'}}$: $m+1$-th → neglecting off-diagonal)

$$\approx \sum_k h''(a_j)\, w_{kj} \frac{\partial E_n}{\partial a_k} + h'(a_j)^2 \sum_k w_{kj}^2 \frac{\partial^2 E_n}{\partial a_k^2}$$

§ 5.4.2 Outer product approximation (Levenberg-Marquardt approximation)

Valid only for appropriately-trained network.

For $E_n = \frac{1}{2}(y_n - t_n)^2$,

$$H = \nabla_w \nabla_w E = \sum_{n=1}^{N} \left[(\nabla y_n)(\nabla y_n)^T + (y_n - t_n) \nabla \nabla y_n \right].$$

(second term: should be zero-mean → smaller)

$$\approx \sum b_n b_n^T \quad \text{w.} \quad b_n \equiv \nabla a_n = \nabla y_n.$$

(output layer → $F = id.$ for regression)

For cross-entropy error $E_n = -t_n \log y_n$ with $\sigma(a)$,

$$H \approx \sum_{n=1}^{N} y_n (1 - y_n)\, b_n b_n^T .$$

§5.4.3

$$H_N = \sum_{n=1}^{N} b_n b_n^T$$

$$\Rightarrow H_N^{-1} = H_{N-1}^{-1} - \frac{H_{N-1}^{-1} b_N b_N^T H_{N-1}^{-1}}{1 + b_N^T H_{N-1}^{-1} b_N}$$

(H_0 can be set $H_0 = \alpha I$ with small α)

§5.4.4 Numerical differentiation

for $f(x)$,

$$\frac{\partial f}{\partial x} = \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon} + O(\varepsilon^2)$$

similarly, for $f(x,y)$,

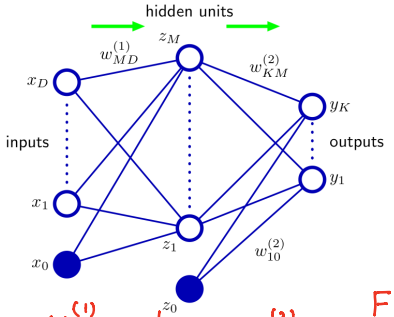
$$\frac{\partial^2 f}{\partial x \partial y} = \frac{1}{4\varepsilon^2}\Big[f(x+\varepsilon,y+\varepsilon) - f(x+\varepsilon,y-\varepsilon) - f(x-\varepsilon,y+\varepsilon) + f(x-\varepsilon,y-\varepsilon)\Big] + O(\varepsilon^2)$$

$$= \frac{1}{2\varepsilon}\left[\frac{\partial f}{\partial x}(x,y+\varepsilon) - \frac{\partial f}{\partial x}(x,y-\varepsilon)\right] + O(\varepsilon^2)$$

§5.4.5

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

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



$x_i \xrightarrow{w_{ji}^{(1)}} a_j \xrightarrow{h} z_j \xrightarrow{w_{kj}^{(2)}} y_k \xrightarrow{F} \text{output}$

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

where $\delta_k = \dfrac{\partial E_n}{\partial a_k}$, $\quad M_{kk'} \equiv \dfrac{\partial^2 E_n}{\partial a_k \partial a_{k'}}$

§5.4.6 Fast calculation of Hv

Denoting $v^T \nabla_w = \mathcal{R}$, ($v \in \mathbb{R}^W$) ← Because $\nabla : \mathbb{R} \to \mathbb{R}^W$, $\mathcal{R} : \mathbb{R} \to \mathbb{R}$

$$v^T H = v^T \nabla(\nabla E) = \mathcal{R}(\nabla E).$$ Note that $\mathcal{R}(w) = v$.

E.g. for 2-layer network with $F = id.$,

$$\begin{cases} y_k = \sum w_{kj} z_j \\ z_j = h(a_j) \\ a_j^{(1)} = \sum w_{ji} x_i. \end{cases}$$

∇E is given by backpropagation as

$$\frac{\partial E}{\partial w_{kj}} = \delta_k z_j = z_j (y_k - t_k),$$

$$\frac{\partial E}{\partial w_{ji}} = \delta_j x_i = x_i h'(a_j) \sum_{k'} \delta_{k'} w_{k'j}.$$

While, $\mathcal{R}(y_k) = \sum_j w_{kj} \mathcal{R}(z_j) + \sum_j v_{kj} z_j$. $\quad \mathcal{R}(z_j) = h'(a_j) \mathcal{R}(a_j)$

$\mathcal{R}(\delta_k) = \mathcal{R}(y_k) \quad \mathcal{R}(a_j) = \sum_i v_{ji} x_i,$

$$\mathcal{R}(\delta_j) = \sum_k \left[h''(a_j) \mathcal{R}(a_j) w_{kj} \delta_k + h'(a_j) v_{kj} \delta_k + h'(a_j) w_{kj} \mathcal{R}(\delta_k) \right]$$

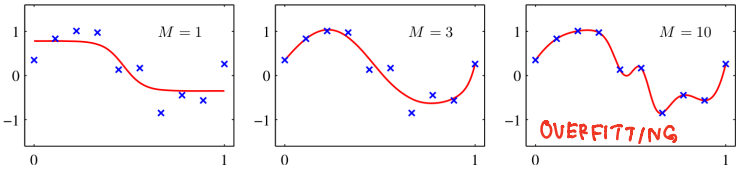
Thus $v^T H = \mathcal{R}(\nabla E)$ is given by

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

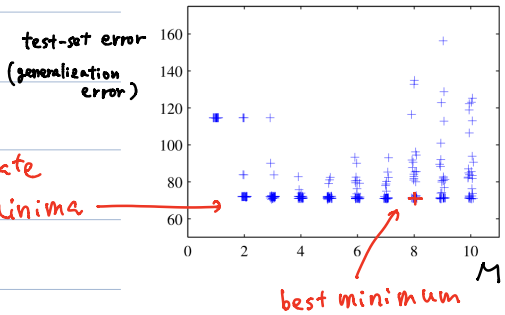
$$\mathcal{R}\left(\frac{\partial E}{\partial w_{ji}}\right) = x_i \mathcal{R}(\delta_j).$$

§ 5.5 Regularization

… How to determine #(hidden units) ≡ M ?



"M giving the best test-set error" is not the optimal M.



§ 5.5.1

Simple weight-decay $E_D = \frac{\lambda}{2}\|w\|^2$ does not work

because it does not respect the symmetry

$$\begin{cases} x \mapsto Ax + b \\ w_{ji} \mapsto w_{ja} A^{-1}_{ai} \\ w_{j0} \mapsto w_{j0} - w_{ja} A^{-1}_{ai} b_i \end{cases} \quad \text{OR} \quad \begin{cases} y \mapsto Ay + b \\ w_{kj} \mapsto A_{ka} w_{aj} \\ w_{k0} \mapsto A_{ka} w_{a0} + b. \end{cases}$$

The text does not discuss this general linear transformation but rather just a scale transformation. I suppose it is because we cannot build a generalizer that respects this symmetry.

We can build a generalizer that respects $A = \alpha I$, a subset of the above symmetry:

$$E_D = \frac{\lambda_1}{2} \sum_{w \in W^{(1)}} w^2 + \frac{\lambda_2}{2} \sum_{w \in W^{(2)}} w^2$$

(without bias weights.)

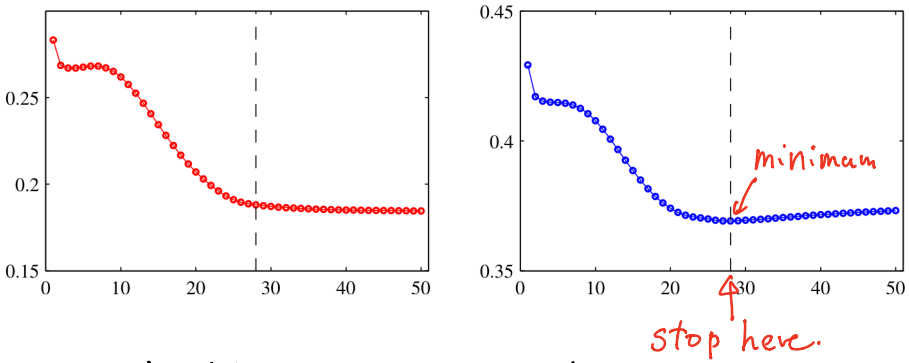
This corresponds to the prior

$$p(w \mid \alpha_1, \alpha_2) \propto \exp\left[-\frac{\alpha_1}{2} \sum_{w \in W^{(1)}} w^2 - \frac{\alpha_2}{2} \sum_{w \in W^{(2)}} w^2 \right].$$

This is improper due to the absence of the biases, (Then no longer b-invariant.)

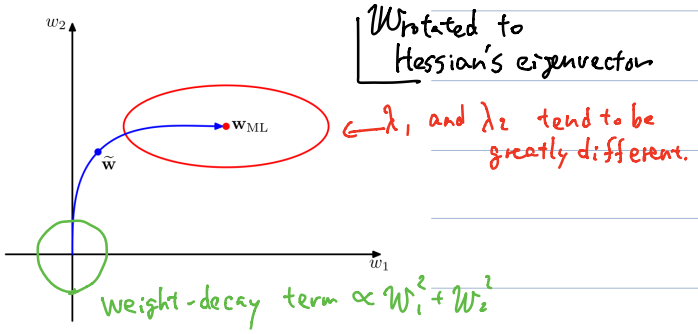
So it is common to include separate priors for biases.

§ 5.5.2 Early stopping



… we need validation set + extra calculation.

For a quadratic error function,



W rotated to Hessian's eigenvector

$\leftarrow \lambda_1$ and λ_2 tend to be greatly different.

§ 5.5.3

Output should be stable against certain changes of input.

→ We want to tell the invariances to NN

(Otherwise much data are required for NN to learn the invariances by itself.)

- Multiplicate the data with the "invariant" changes.
- Add penalty to changes of output for transformed input: "tangent propagation".
- Preprocessing
- Invariant properties to be built into the structure of NN.

§ 5.5.4 Tangent propagation ... penalty for invariance.

Continuous transformation with L-parameter:

$$x \longmapsto s(x, \xi) \simeq x + \xi_l \tau_l(x) \quad ; \quad \tau_l(x) = \frac{\delta s}{\delta \xi_l}(x, 0)$$

($\simeq$: infinitesimal)

Considering $L = 1$ for simplicity, the output will be modified by

$$\left.\frac{\delta y}{\delta \xi}\right|_{\xi=0} = \sum_i \frac{\partial y}{\partial x_i} \left.\frac{\partial x_i}{\partial \xi}\right|_{\xi=0} = J \tau .$$

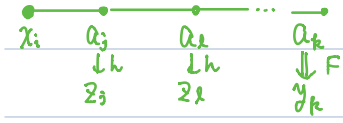
and we can penalize this:

$$E_{transf.} = \lambda \sum_{n=1}^{N} \Omega_n \quad , \text{ where } \Omega_n = \frac{1}{2} \left\| J(x_n)\, \tau(x_n) \right\|^2 .$$

Jacobian $J = \frac{\partial y}{\partial x}$ can be calculated by backprop:

$$J_{ki} = \frac{\partial y_k}{\partial x_i} = \sum_j \frac{\partial y_k}{\partial a_j} \frac{\partial a_j}{\partial x_i} = \sum_j w_{ji} \frac{\partial y_k}{\partial a_j}$$

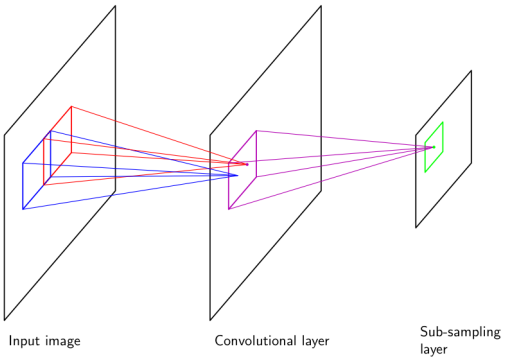
$$\frac{\partial y_k}{\partial a_j} = \sum_l \frac{\partial y_k}{\partial a_l} \frac{\partial a_l}{\partial a_j} = h'(a_j) \sum_l w_{lj} \frac{\partial y_k}{\partial a_l}$$



and $\tau(x_n)$ is calculated by

$$\tau(x) \simeq \frac{1}{2\varepsilon} \left[s(x, \varepsilon) - s(x, -\varepsilon) \right] .$$

§ 5.5.6 Convolutional Neural Network



Invariant properties to be built into the structure of NN.

→ We need to know the "symmetries" (e.g. translation invariance) beforehand to use this technique.

a convolutional layer = feature maps

a feature map has units :

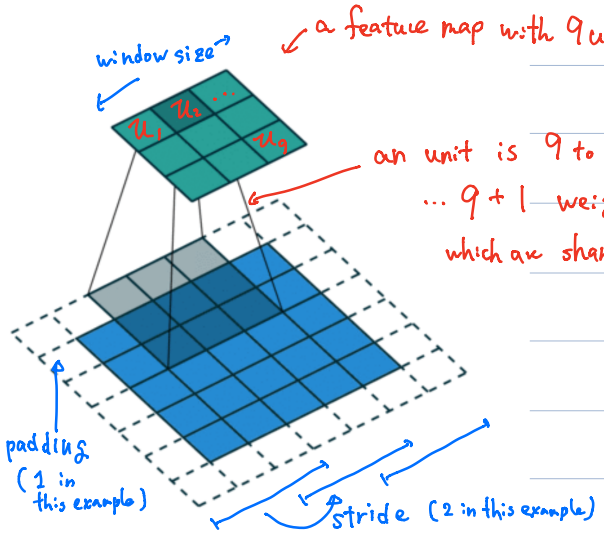


image from http://kaeken.hatenablog.com/entry/2016/11/13/135554

a subsampling layer (pooling layer) :

reduce the data size of each feature map (by MAX, AVG, etc.)

§5.5.7 Soft weight sharing

… regularization to motivate similar weight

(≡ prior)

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

$\Rightarrow \pi_j$ is constrained: $\sum \pi_j = 1$.

$$\Rightarrow \tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \Omega(\mathbf{w}) \quad \text{where}$$

$$\Omega(\mathbf{w}) = -\sum_i \ln \left(\sum_j \pi_j \, \mathcal{N}(w_i \mid \mu_j, \sigma_j^2) \right)$$

→ error minimization w.r.t. $\{w_i, \pi_j, \mu_j, \sigma_j\}$

Defining $\gamma_j(x) = \dfrac{\pi_j \, \mathcal{N}(x \mid \mu_j, \sigma_j^2)}{\sum_k \pi_k \, \mathcal{N}(x \mid \mu_k, \sigma_k^2)}$ (posterior — prior),

$$\frac{\partial \tilde{E}}{\partial w_i} = \frac{\partial E}{\partial w_i} + \sum_{j=1}^{M} \frac{w_i - \mu_j}{\sigma_j^2} \gamma_j(w_i) \qquad \cdots w \to \mu.$$

$$\frac{\partial \tilde{E}}{\partial \mu_j} = -\sum_i \frac{w_i - \mu_j}{\sigma_j^2} \gamma_j(w_i) \qquad \cdots \mu \to w.$$

$$\frac{\partial \tilde{E}}{\partial \sigma_j} = \sum_i \left(\frac{1}{\sigma_j} - \frac{(w_i - \mu_j)^2}{\sigma_j^3} \gamma_j(w_i) \right)$$

[practically $\eta_j = \log \sigma_j^2$ is used to guarantee $\sigma_j^2 > 0$.]

I doubt the sentence "It also has the effect of discouraging pathological solutions in which one or more of the σ_j goes to zero" because this is just a variable replacement. Such an effect might be observed in numerical calculation, which is just because of real-number precision.

Noting that $\sum \pi_j = 1$ with $0 \le \pi_j \le 1$, $\frac{\partial}{\partial \pi}$ should be calculated with introducing $\pi_j =: \dfrac{\exp a_j}{\sum \exp a_k}$. $\left(\Leftrightarrow a_j = \log \pi_j + C\right)$

$$\frac{\partial \tilde{E}}{\partial a_j} = \frac{\partial \pi_k}{\partial a_j} \frac{\partial \Omega}{\partial \pi_k}$$

$$= \left(\delta_{jk} - \pi_j \right) \pi_k \cdot \left(-\sum_i \frac{\gamma_k}{\pi_k} \right)$$

$$= \sum_i \left(\pi_j - \gamma_j(w_i) \right). \qquad \cdots \pi \to \gamma.$$

Direct calculation gives equivalent result:

$$-\frac{\partial \Omega}{\partial \pi_j} = \frac{N_j - N_M}{\sum \pi_k N_k} = \frac{\gamma_j}{\pi_j} - \frac{\gamma_M}{\pi_M} \propto \gamma_j (1 - \pi_1 - \cdots - \pi_{M-1}) - \gamma_M \pi_j$$

$$\downarrow \sum_{j=1}^{M}$$

$$\gamma_M - \pi_M$$

§ 5.6 Mixture density network

Non-Gaussian $p(t|x)$ often appears, e.g., in "inverse problem"

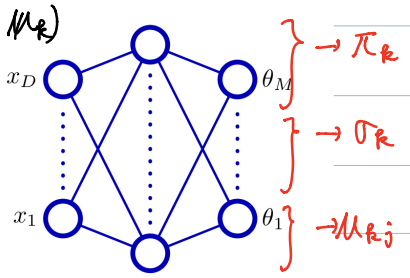
→ use "mixture model" for $p(t|x)$. ... multimodal distribution

$$\sum_i \pi_i(x) \underbrace{p_i(t|x)}_{\text{coefficients also depend on } x.}$$

EXAMPLE: Heteroscedastic model

$$p(t|x) = \sum_{k=1}^{K} \pi_k(x)\, \mathcal{N}(t \mid \mu_k(x), \sigma_k^2(x)\, \mathbb{I})$$

practically we use a^π and a^σ (and $a^\mu_k \equiv \mu_k$)



$$\pi_k =: \frac{\exp a_k^\pi}{\sum_l \exp a_l^\pi}$$

$$\sigma_k^2 =: \exp a_k^\sigma$$

so that $\begin{cases} \sum \pi_k = 1, \ 0 \le \pi_k \le 1 \\ \sigma_k^2 > 0 \end{cases}$ are satisfied.

Straightforwardly, w.r.t. $N_{nk} := \mathcal{N}(t_n|\mu_k(x_n), \sigma_k^2(x_n)\,\mathbb{I})$,

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k(\mathbf{x}_n, \mathbf{w}) \mathcal{N}\left(\mathbf{t}_n | \boldsymbol{\mu}_k(\mathbf{x}_n, \mathbf{w}), \sigma_k^2(\mathbf{x}_n, \mathbf{w})\, \mathbb{I}\right) \right\}$$

$$\frac{\partial E_n}{\partial a_k^\pi} = \pi_k - \gamma_{nk}, \quad \frac{\partial E_n}{\partial a_{kl}^\mu} = \gamma_{nk} \cdot \frac{\mu_{kl} - t_{nl}}{\sigma_k^2}, \quad \frac{\partial E_n}{\partial a_k^\sigma} = \gamma_{nk}\left(L - \frac{\|t_n - \mu_k\|^2}{\sigma_k^2}\right)$$

where $L = \dim t$, $\gamma_{nk} \equiv \gamma_k(t_n|x_n) := \dfrac{\pi_k N_{nk}}{\sum_{l=1}^{K} \pi_l N_{nl}}$.

→ we can determine not only

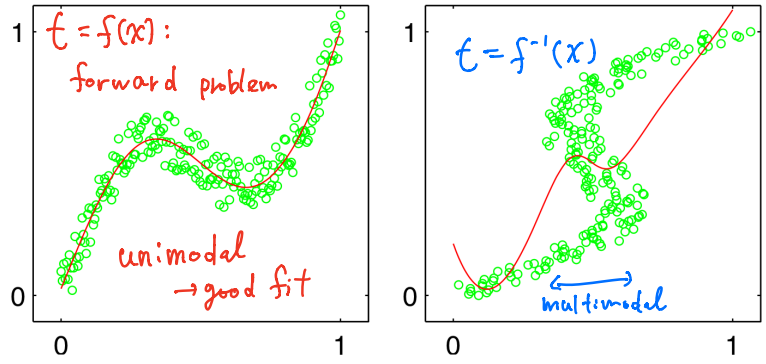
but also the uncertainty

$$\mathbb{E}[\mathbf{t}|\mathbf{x}] = \int \mathbf{t} p(\mathbf{t}|\mathbf{x})\, d\mathbf{t} = \sum_{k=1}^{K} \pi_k(\mathbf{x}) \boldsymbol{\mu}_k(\mathbf{x})$$

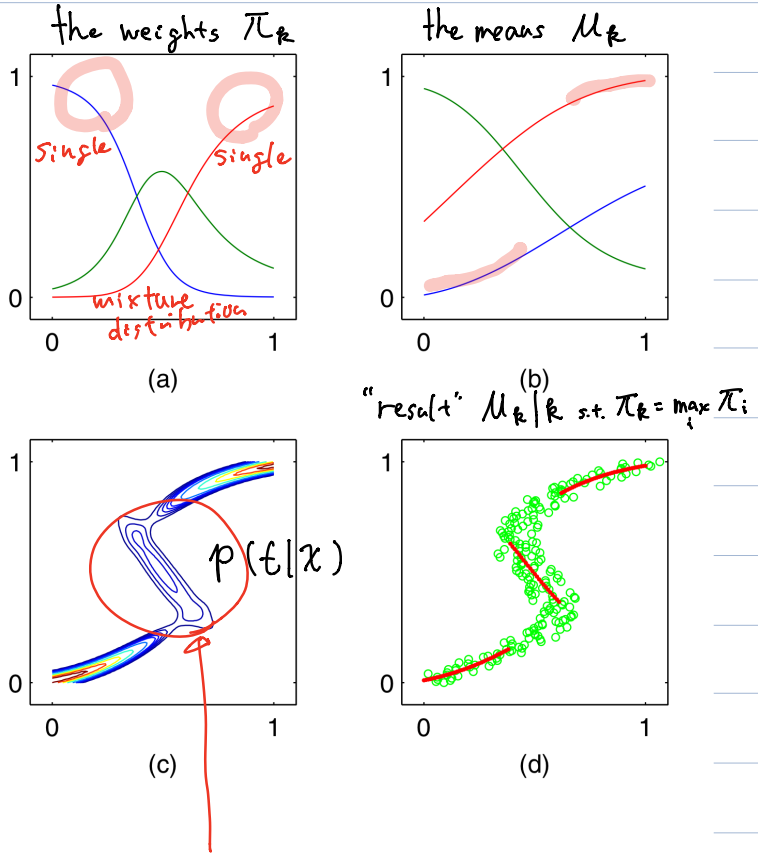
$$\begin{aligned} s^2(\mathbf{x}) &= \mathbb{E}\left[\|\mathbf{t} - \mathbb{E}[\mathbf{t}|\mathbf{x}]\|^2 |\mathbf{x}\right] \\ &= \sum_{k=1}^{K} \pi_k(\mathbf{x}) \left\{ \sigma_k^2(\mathbf{x}) + \left\| \boldsymbol{\mu}_k(\mathbf{x}) - \sum_{l=1}^{K} \pi_l(\mathbf{x}) \boldsymbol{\mu}_l(\mathbf{x}) \right\|^2 \right\} \end{aligned}$$

An example : fitting $f(x) = x + 0.3\sin 2\pi x$

- Simple NN



- Mixture density network



mean $\mathbb{E}[t|x] = \sum \pi_k \mu_k$ is not good for this purpose.

... We should rather use, for example,
the mean μ_k of the most probable distribution.

§5.7 Bayesian NN.

… no longer analitical due to non-linearity

{ "variational influence" … §10.
  Laplace approximation

$p(z) \approx \sqrt{\frac{A}{2\pi}} \exp\left(-\frac{A}{2}(z-z_0)^2\right)$,
$f'(z_0)=0 \quad f''(z_0)<0 \quad A = -\frac{d^2}{dz^2}\ln f(z_0)$.

Given a dataset $\mathcal{D} = \{(x_1, t_1), \cdots, (x_N, t_N)\}$.

posterior of w ∝ prior of w · likelihood

$$p(w|\mathcal{D}, \alpha, \beta) \propto p(w|\alpha)\, p(t|X, w, \beta)$$

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

we shall suppose $\beta^{-1} \equiv \sigma^2$

$$\prod_n \mathcal{N}(t_n | y(x_n, w), \beta^{-1})$$

$y(x_n, w)$: NN: non-linear

Laplace approx. (with α and β fixed for simplicity)

$$\ln p(w|\mathcal{D}) = -\frac{\alpha}{2} w^T w - \frac{\beta}{2}\sum_n \left(y(x_n, w) - t_n\right)^2 + \text{const.}$$

$$\Rightarrow A = -\nabla\nabla \ln p(w|\mathcal{D}) = \alpha I + \beta H; \qquad H = \nabla\nabla \frac{1}{2}(y-t)^2.$$

(back propagation)

$$p(w|\mathcal{D}) \approx \mathcal{N}(w|w_{MAP}, A^{-1})$$
$$=: q(w|\mathcal{D})$$

Predictive distribution is given by

$$p(t|x, \mathcal{D}) \approx \int dw\; p(t|x, w, \beta)\, q(w|\mathcal{D})$$

$$= \int dw\; \underline{\mathcal{N}(t|y(x,w), \beta^{-1})}\, \mathcal{N}(w|w_{MAP}, A^{-1})$$

again non-linear

assuming $w - w_{MAP} \equiv \Delta w \ll \beta^{-1/2} = \sigma$

$$\approx \mathcal{N}(t|y(x, w_{MAP}) + g^T(w - w_{MAP}), \beta)$$

$$\approx \mathcal{N}(t|y(x, w_{MAP}), \sigma^2(x)) \qquad g := \nabla_w y|_{w=w_{MAP}}$$

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

§5.7.2 Hyperparameters α and β can also be optimized :

predictive dist. / intrinsic noise / parameters from hyperparameter / hyperparameter estimation

$$p(t|x,\mathcal{D}) = \int dw\, d\alpha\, d\beta\; p(t|w,\beta)\, p(w|\mathcal{D},\alpha,\beta)\, p(\alpha,\beta|\mathcal{D})$$

$$\ln p(\mathcal{D}|\alpha,\beta) = \ln\left[\int dw\; \underbrace{p(\mathcal{D}|w,\alpha,\beta)}_{\prod N(t_n|y(x_n,w),\beta^{-1})}\, p(w|\alpha,\beta)\right]$$

→ assume $N(w|0,\alpha^{-1}\mathrm{I})$

Laplace

$$\approx \ln\left[\int dw\; f(w_{MAP}) \exp\left(-\frac{1}{2}(w-w_{MAP})^T A (w-w_{MAP})\right)\right]$$

$$f(w) = p(\mathcal{D}|w,\alpha,\beta)\, p(w|\alpha,\beta) \quad \cdots \text{numerator of } p(w|\mathcal{D},\alpha,\beta)!$$

→ A and w_{MAP} are in the previous page.

$$p(w|\mathcal{D},h) = \frac{p(\mathcal{D}|w,h)\,p(w|h)}{p(\mathcal{D}|h)} : \text{Bayes' theorem}$$

$$p(\mathcal{D}|h) = \int dw\; p(\mathcal{D}|w,h)\,p(w|h) : \text{marginalization}$$

$$= \ln f(w_{MAP}) - \frac{1}{2}\ln|A| + \frac{W}{2}\log 2\pi$$

$$= \underbrace{-\sum_n \frac{\beta}{2}\left(y(x_n,w_{MAP}) - t_n\right)^2 + \frac{\alpha}{2}\|w_{MAP}\|^2}_{\equiv E(w_{MAP})} + \frac{N}{2}\log\frac{\beta}{2\pi} + \frac{W}{2}\log 2\pi\alpha - \frac{1}{2}\ln|A|$$

(W ← dim w)

In evidence framework, we make point estimates for α and β by maximizing $\ln p(\mathcal{D}|\alpha,\beta)$.

As §3.5.1, with eigensystem $\{\lambda_i, u_i\}$ of βH,

$A = -\nabla\nabla \ln p(w|\mathcal{D}) = \alpha \mathrm{I} + \beta H$; $H = \nabla\nabla \frac{1}{2}(y-t)^2$.

$$\alpha = \frac{\gamma}{\|w_{MAP}\|^2},\quad \beta^{-1} = \frac{1}{N-\gamma}\sum_n \left(y(x_n,w_{MAP}) - t_n\right)^2,\quad \gamma = \sum_{i=1}^{W}\frac{\lambda_i}{\alpha+\lambda_i}.$$

and iteratively $\{w_{MAP}, \alpha, \beta\}$ is optimized.
Note that multiple (inequivalent) solutions may exist.

I do not understand (and thus doubt) the note that we have ignored the α-dependence of λs. We can consider p as a function of {w_MAP, α, β}, which is valid since we anyway find the minimum iteratively. Then the Hessian has no dependence on α and β.

→ $A = -\nabla\nabla \ln p(w|\mathcal{D}) = \alpha \mathrm{I} + \beta H$; $H = \nabla\nabla \frac{1}{2}(y-t)^2$

We need $p(\mathcal{D})$ to compare different models.

$$p(\mathcal{D},\alpha,\beta) = p(\mathcal{D}|\alpha,\beta)\,p(\alpha,\beta) \approx p(\mathcal{D}|\hat{\alpha},\hat{\beta})$$ (or marginalization: MacKay 1992c, Bishop 1995a)

and $p(\mathcal{D}) \approx p(\mathcal{D},\alpha,\beta) \times 2^M M!$

but note that $|A|$ is tough to calculate numerically.

§9.2.3 For classification. $\mathcal{X} \mapsto t = \{0, 1\}$

$\equiv p(\mathcal{D}|w)$

$$p(w|\mathcal{D}, \alpha) \propto p(w|\alpha)\, p(t|X, w)$$

$$\to \mathcal{N}(w|0, \alpha^{-1}\mathbb{I}) \prod_n y_n^{t_n}(1-y_n)^{1-t_n}$$

$$\to -\ln p(w|\mathcal{D}, \alpha) = E(w) + \text{const.}$$

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

$$\nabla E(w) = \sum_n \frac{y_n - t_n}{y_n(1-y_n)} \nabla y_n + \alpha w \implies w_{MAP}.$$

$$\nabla\nabla E(w) = \alpha \mathbb{I} + \sum_n \left[\frac{y_n^2 - 2t y_n + t}{y_n^2(1-y_n)^2} (\nabla y)(\nabla y) + \frac{y_n - t_n}{y_n(1-y_n)} \nabla\nabla y \right]$$

Hessian H

Model discrimination

$$\ln p(\mathcal{D}|\alpha) = \ln \left[\int dw\; p(\mathcal{D}|w)\, p(w|\alpha) \right]$$

Laplace

$$\approx \ln f(w_{MAP}) - \frac{1}{2}\ln|A| + \frac{W}{2}\ln 2\pi$$

$$= \sum_n \left[t_n \ln y_n + (1-t_n)\ln(1-y_n) \right]_{w_{MAP}} - \frac{W}{2}\ln\frac{2\pi}{\alpha} - \frac{\alpha}{2}\|w_{MAP}\|^2$$

$\equiv -E(w_{MAP})$

$$= -E(w_{MAP}) - \frac{1}{2}\ln|A| + \frac{W}{2}\ln\alpha \qquad -\frac{1}{2}\ln|A| + \frac{W}{2}\ln 2\pi$$

$$\implies \alpha = \frac{\gamma}{\|w_{MAP}\|^2}, \quad \gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\alpha + \lambda_i} \;;\; \lambda_i \text{ are eigenvalues of } H. \text{ (and iterate)}$$

Predictive distribution is given by

$$p(t|x, \mathcal{D}) \approx \int dw\; p(t|x, w)\, \mathcal{N}(w|w_{MAP}, A^{-1}) \qquad \left(\text{or} \approx p(t|x, w_{MAP}) \right)$$

($p(t|x,w) \leftarrow y^t(1-y)^{1-t}$)

Expansion should be done w.r.t. $a = \sigma^{-1}(y)$, not y itself.

$$a(x, w) \simeq a_{MAP}(x) + b^T(w - w_{MAP}) \quad \text{with } b = \nabla a(x, w_{MAP}) \text{ and}$$

$$p(t|x, \mathcal{D}) \approx \int dw\; p(t|x, w)\, \mathcal{N}(w|w_{MAP}, A^{-1}) \quad : g(a) = \sigma(a)^t(1-\sigma(a))^{1-t}$$

$$= \int dw\, da\; \delta(a - a_{MAP} - b^T(w - w_{MAP}))\, g(a)\, \mathcal{N}(w|w_{MAP}, A^{-1})$$

$$= \int da\; g(a)\, \mathcal{N}(a|a_{MAP}, b^T A^{-1} b)$$

$$\therefore p(t=1|x, \mathcal{D}) \approx \sigma(\kappa a_{MAP}) \quad \text{where } \kappa = \left(1 + \frac{\pi}{8} b^T A^{-1} b\right)^{-1/2}.$$