Machine Learning

- supervised learning → regression / classification
- unsupervised learning → clustering / non clustering

Linear Regression

$$J = \frac{1}{2m}\sum_{i}^{m}(h_\theta(x^{(i)}) - y)^2$$

- 2 - variables $h_\theta(x) = \theta_0 + \theta_1 x$

- Gradient Descent

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial\theta_0}J(\theta)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

- Multivariables Linear Regression

$h_\theta(x) = \theta^T x$ ← parameter for variable

$[\ ]_{1\times n}\ [\ ]_{n\times 1}$ → $x_0 = 1$

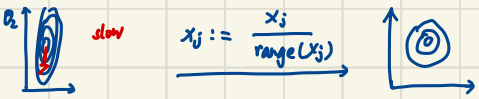
- Gradient Decrease

repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

}

- Feature Scaling +



- Mean normalization

$$x_j := \frac{x_j - \overline{x_j}}{\text{range}(x_j)}$$

- Polynomial Regression ← Multivariable Linear Regression

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$ → $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

$\begin{cases} x_1 = x \\ x_2 = x^2 \\ x_3 = x^3 \end{cases}$ → Feature Scaling

Normal Equation ?

{ n small : normal equation
{ n large : gradient descent (10⁴)

No Feature Scaling

$\theta = (X^T X)^{-1} X^T y$ → $\min_\theta J(\theta)$

design matrix

$$X = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots \\ 1 & x_1^2 & x_2^2 & \cdots \\ \vdots & \vdots & \vdots & \\ 1 & & & \end{bmatrix} \quad y = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$m \times (n+1)$ $\quad$ m-d vector

↑

m example $[x^{(i)}, y^{(i)}]$

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \end{bmatrix} \rightarrow X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}$$

$m \times n$

$X^T X$ non - invertible

- redundant features [linearly dependent]
- too many features [$m \leq n$]
  - delete
  - regularization

Vectorization

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$
$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

$$\underset{n\times 1}{[\theta]} := [\theta] - \frac{\alpha}{m} \begin{bmatrix} (1)\cdot x_0^{(1)} + (2)\, x_0^{(2)} + \cdots + (m)\cdot x_0^{(m)} \\ (1)\cdot x_1^{(1)} + (2)\, x_1^{(2)} + \cdots + (m)\cdot x_1^{(m)} \\ (1)\cdot x_2^{(1)} + (2)\, x_2^{(2)} + \cdots + (m)\cdot x_2^{(m)} \end{bmatrix} \rightarrow \begin{bmatrix} (1) \\ (2) \\ \vdots \\ (m) \end{bmatrix}' \cdot \begin{bmatrix} x_0^{(1)} \\ x_0^{(2)} \\ \vdots \\ x_0^{(m)} \end{bmatrix}$$
$$\rightarrow \begin{bmatrix} (1) \\ (2) \\ \vdots \\ (m) \end{bmatrix}' \cdot \begin{bmatrix} x_1^{(1)} \\ \vdots \\ x_1^{(m)} \end{bmatrix}$$
$$\rightarrow \begin{bmatrix} (1) \\ (2) \\ \vdots \\ (m) \end{bmatrix}' \cdot [\, x_1$$

$$:= [\theta] - \frac{\alpha}{m} \underset{n\times m}{\begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \cdots & x_0^{(m)} \\ x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(m)} \end{bmatrix}} \cdot \underset{m\times 1}{\begin{bmatrix} (1) \\ (2) \\ \vdots \\ (m) \end{bmatrix}}$$
($n\times 1$)

$$h_\theta(x^{(i)}) = \theta^T x^{(i)} \leftarrow \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$$

$$\begin{bmatrix} (1) \\ (2) \\ \vdots \\ (m) \end{bmatrix} = \begin{bmatrix} \theta^T x^{(1)} - y^{(1)} \\ \theta^T x^{(2)} - y^{(2)} \\ \vdots \\ \theta^T x^{(m)} - y^{(m)} \end{bmatrix} = X\ \theta\ - y$$
($m\times 1$) = ($m\times n$)($n\times 1$) − ($m\times 1$)

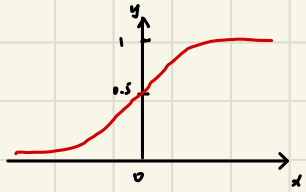
$$\theta := \theta - \frac{\alpha}{m} X^T (X\theta - y)$$

Classification

Sigmoid Function / Logistic Regression

Linear Regression

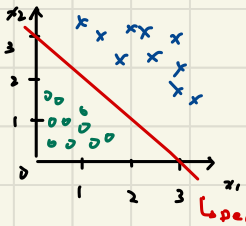
$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$



$$h_\theta(x) = P(y=1 \mid x, \theta)$$

Decision Boundary

- linear

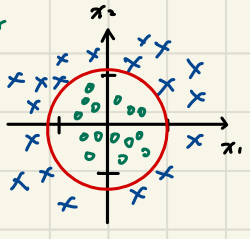


Decision Boundary

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ $\quad \theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$-3 + x_1 + x_2 \geq 0$

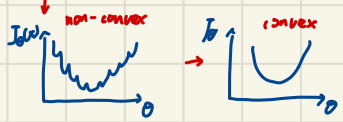
- non-linear

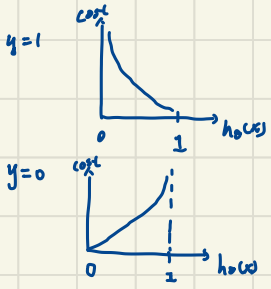


$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$ $\quad \theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

Cost function

if $J_\theta(x) = \frac{1}{m}\sum_i^m \underbrace{\frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2}_{\text{cost}(h_\theta(x^{(i)}),\, y^{(i)})}$





$$\text{cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & y=1 \\ -\log(1-h_\theta(x)) & y=0 \end{cases}$$

$$\text{cost}(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

$$\Downarrow$$

$$J(\theta) = -\frac{1}{m}\left[\sum_i^m y^{(i)}\log(h_\theta(x)) + (1-y)\log(1-h_\theta(x))\right]$$

Gradient Descent

repeat {

$$\theta_j := \theta_j - \alpha\frac{\partial J}{\partial \theta} = \theta_j - \alpha\left(-\frac{1}{m}\sum_i^m \frac{y^i}{h_\theta(x)}\cdot\frac{\partial h_\theta(x)}{\partial z}\cdot\frac{\partial z}{\partial \theta_j} + \frac{(1-y)}{1-h_\theta(x)}\cdot(-1)\cdot\frac{\partial h_\theta(x)}{\partial z}\cdot\frac{\partial z}{\partial \theta_j}\right)$$

}

$$= \theta_j - \alpha\left(-\frac{1}{m}\sum \frac{y}{h_\theta(x)}\cdot h_\theta(x)(1-h_\theta(x))\cdot x_j - \frac{(1-y)}{1+h_\theta(x)}\cdot h_\theta(x)\cdot(1-h_\theta(x))\cdot x_j\right)$$

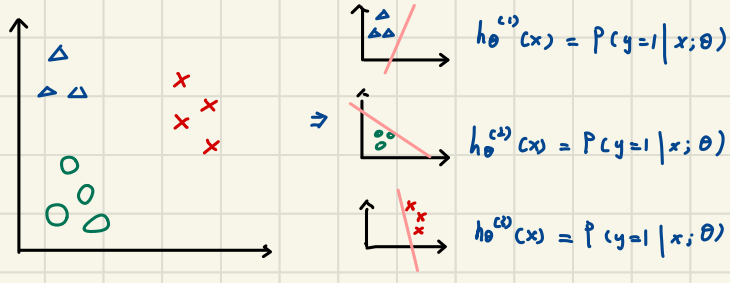
$$= \theta_j - \alpha\left(-\frac{1}{m}\sum\left[y - y h_\theta(x) - h_\theta(x) + y h_\theta(x)\right]x_j\right)$$

$$= \theta_j - \frac{\alpha}{m}\sum(h_\theta(x) - y)x_j$$

Vectorization

$$\theta := \theta - \frac{\alpha}{m}X^T(g(X\theta) - \vec{y})$$

Multiclass Classification



$$h_\theta^{(1)}(x) = P(y=1|x;\theta)$$

$$h_\theta^{(2)}(x) = P(y=1|x;\theta)$$

$$h_\theta^{(3)}(x) = P(y=1|x;\theta)$$

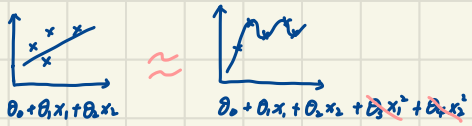
$$\max h_\theta^{(i)}(x)$$

Overfitting → Regularization

- perfectly fit example
- ✗ generalise to new data, ✗ predict well

Solution:

1. Reduce number of features (i)
   - manually
   - model selection algorithm
2. Regularization
   - keep all features, reduce values of θ → $\min_\theta \frac{1}{2m}\sum_i^m (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$ ($\theta_3 \to 0$, $\theta_4 \to 0$)
   - works well with lots of features, each contributes a little to predict y



Regularized Linear Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{m}\theta_j^2\right] \quad \times\theta_0$$

λ: regularization parameter

Gradient Descent $\times\theta_0$

$$\theta_j := \theta_j\left(1 - \frac{\partial\lambda}{m}\right) - \frac{\partial}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

↓vectorization

$$\theta := \left(1 - \frac{\partial\lambda}{m}\right)\theta - \frac{\partial}{m}X^T(\theta^T X - \vec{y})$$

Normal equation

$$\theta = (X^TX + \lambda\begin{bmatrix}0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1\end{bmatrix})^{-1}X^T\vec{y}$$

$(n+1)\times(n+1)$

Regularized Logistic Regression

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h_\theta(x^{(i)}) + (1-y)\cdot\log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Gradient Descent ($\times\theta_0$)

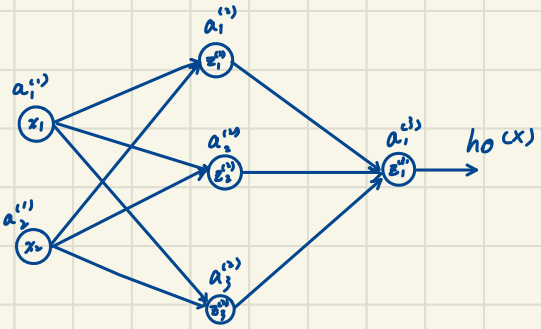
$$\theta_j := \theta_j - \frac{\partial}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} - \frac{\partial\lambda}{m}\theta_j$$

$$\theta_j := \theta_j\left(1 - \frac{\partial\lambda}{m}\right) - \frac{\partial}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

↓vectorization

$$\theta := \left(1 - \frac{\partial\lambda}{m}\right)\theta - \frac{\partial}{m}X^T(g(\theta^T x) - \vec{y})$$

Neutral Network



Layer 1 Layer 2 Layer 3

$a^{(1)}$ $a^{(2)}$ $a^{(3)}$

$a^{(1)} = [a_0^{(1)}\ a_1^{(1)}\ a_2^{(1)}]$ $a^{(2)} = [a_0^{(2)} \cdots a_3^{(2)}]$ $a^{(3)} = [a_0^{(3)} \cdots a_1^{(3)}]$

≥ 1 ≥ 1

$\rightarrow a_1^{(2)} = g(\Theta_{1,0}^{(1)} a_0^{(1)} + \Theta_{1,1}^{(1)} a_1^{(1)} + \Theta_{1,2}^{(1)} a_2^{(1)})$

$\vdots$

$a_3^{(2)} = g(\Theta_{3,0}^{(1)} a_0^{(1)} + \Theta_{3,1}^{(1)} a_1^{(1)} + \Theta_{3,2}^{(1)} a_2^{(1)})$

↳ $\Theta^{(j)} \Rightarrow s_{j+1} \times (s_j + 1)$

$x^{(1)} = a^{(1)}$

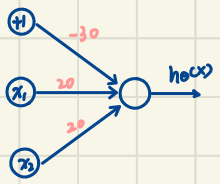
↳ $\Theta^{(1)} \cdot a^{(1)} = z^{(2)}$

↳ $g(z^{(1)}) = a^{(2)}$

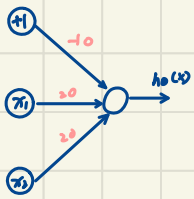
↳ $\Theta^{(2)} \cdot a^{(2)} = z^{(3)}$

↳ $g(z^{(3)}) = h_\Theta(x)$

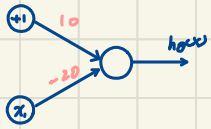
AND



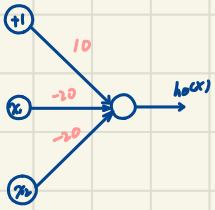
OR

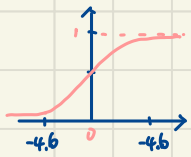


NOT



(NOT x_1) AND (NOT x_2)





Cost Function

$$J_\Theta = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - h_\Theta(x^{(i)}))_k\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2$$

m - batch size

k - number of classes in multi-classification

L - number of layers

s_l - number of units in Layer l

Backpropagation

$\Delta_{ji}^{(l)}$ use for all $\{x^{(m)}, y^{(m)}\}$

for $m : M$

1) $x^{(m)} \Rightarrow a^{(1)}$
2) for $l : L$
   - $z^{(l+1)} = \theta^{(l)} \cdot a^{(l)}$
   - $a^{(l+1)} = g(z^{(l+1)})$
   - $y^{(m)}$
3) $\delta^{(L)} = a^{(L)} - y^{(m)}$

   for $L \to 2$

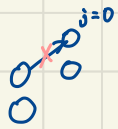
   $\delta^{(l)} = \left[\theta^{(l)T} \delta^{(l+1)}\right] .* g'(z^{(l)})$

   $g'(z^{(l)}) = a^{(l)} .* (1 - a^{(l)})$
4) $\Delta^{(l)} = \Delta^{(l)} + a^{(l)} \cdot \delta^{(l+1)}$

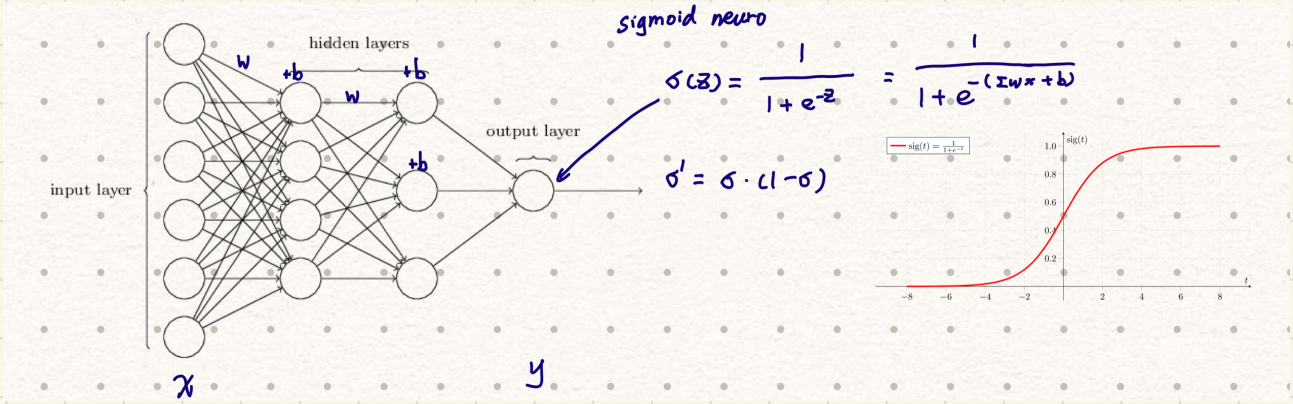
$$D_{ji}^{(l)} = \frac{1}{m}\left(\Delta_{ji}^{(l)} + \lambda \theta_{ji}^{(l)}\right) \quad : \text{if } j \neq 0$$

$$D_{ji}^{(l)} = \frac{1}{m} \Delta_{ji}^{(l)}$$

$j = 0 \Rightarrow \theta_{0i}$



NNN structure



$y_i \rightarrow x_i$

$$x_j^J = \sum_k^k w_{jk}^J \sigma\left(x_k^{J-1}\right) + b_j^J$$

$$X^J = W^J \sigma\left(X^{J-1}\right) + B^J$$

★ $$C = \frac{1}{2n} \sum [y(x) - a]^2$$

Logic

Data get ← training / validation / test

↓

Network intialize ← layers build; $w, b \sim N(0, 1)$

learning initialize : learning rate, epochs, lambda

↓

loop (epochs)

↓

shuffle training data

↓

split batch $\frac{n}{m}$

↓

training 1 vector in batch → get $\frac{\partial C}{\partial b} = \delta^L$, $\frac{\partial C}{\partial w} = a^{L-1}\delta^L$

forward: $x \rightarrow \sigma \rightarrow a$

backpropagation: $\delta^L \rightarrow \delta^{L-1} \rightarrow \delta^l$

$$w' = w - \frac{\gamma}{m} \sum \frac{\partial C}{\partial w}$$

$$b' = b - \frac{\gamma}{m} \sum \frac{\partial C}{\partial b}$$

↓

evaluation accuracy code

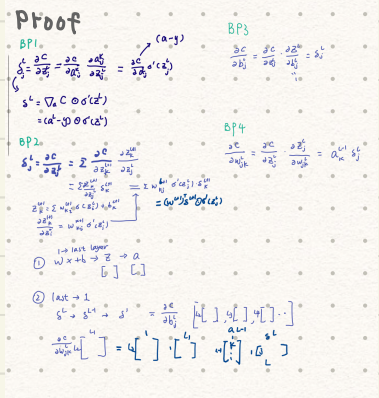
Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad \text{(BP2)}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad \text{(BP3)}$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad \text{(BP4)}$$



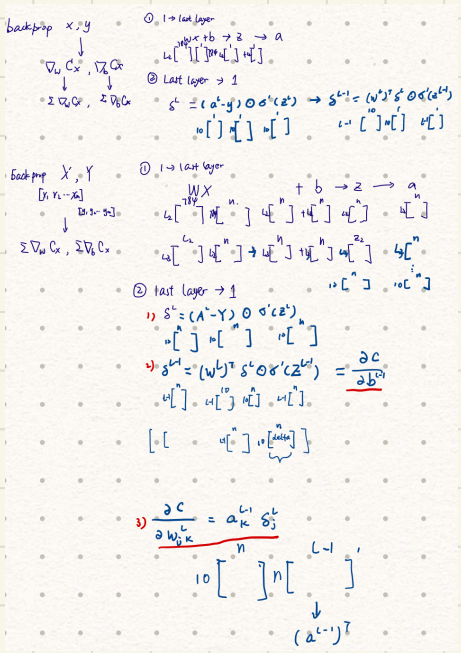
Data structure

Class Network

- n of layers
- layers [, ,]
- w $[w^{[1]}, w^{[2]}, w^{[3]} \cdots]$
- b $[b^{[1]}, b^{[2]}, b^{[3]} \cdots]$
- activations $[a^{[m]}, a^{[m]} \cdots a^{[m]}]$
- zs $[z^{[m]}, z^{[m]}, \cdots z^{[m]}]$
- nabla_b $[\nabla b^{[m]}, \nabla b^{[m]}, \cdots \nabla b^{[m]}]$
- nabla_w $[\nabla w^{[L_1]}, \nabla w^{[L_2]}, \cdots \nabla w^{[L_3]}]$

P1

Batch vector → Batch matrix



Neuron Function

Softmax

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}$$

$$j \neq k: \quad \frac{\partial a_j^L}{\partial z_k^L} = -\frac{e^{z_j^L}}{(\sum_k e^{z_k^L})^2} \cdot e^{z_k^L} < 0$$

$$j = k \quad \frac{\partial a_j^L}{\partial z_j^L} = e^{z_j^L}(\sum_k e^{z_k^L})^{-1} - e^{z_j^L}(\sum_k e^{z_k^L})^{-2} \cdot e^{z_j^L}$$

$$= e^{z_j^L}(\sum_k e^{z_k^L})^{-1}\left(1 - \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}\right) > 0$$

P3 inverting the softmax layer $z_j^L = f(a_j^L)$

$$a_j^L = e^{z_j^L} / \sum_k e^{z_k^L}$$

$$\ln a_j^L = z_j^L - \ln C$$

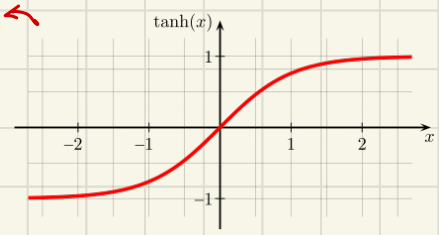
(C: independent of j)

Tanh

$a = \tanh(wx+b)$

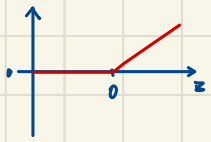
$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \rightarrow \sigma(z) = \frac{1+\tanh(\frac{z}{2})}{2}$$

normalize



Rectified Linear

$a = \max(0, wx+b)$



Cost Function 2 Cross Entropy

$$C = \frac{1}{2n}\sum (y-a)^2$$

↓

$$C = -\frac{1}{n}\sum [y \ln a + (1-y)\ln(1-a)]$$

① $C > 0$

② $$\frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x \left[\frac{y}{\sigma(z)}\cdot\sigma'\cdot x_j - \frac{1-y}{1-\sigma}\cdot\sigma'\cdot x_j\right]$$

$$= -\frac{1}{n}\sum \frac{y-\sigma}{\sigma(1-\sigma)}\sigma' x_j$$

↓ $\sigma' = \sigma(1-\sigma)$

$$= \frac{1}{n}\sum (\sigma - y)x_j \rightarrow \frac{\partial C}{\partial w_{jk}^L} = \frac{1}{n}\sum_x a_k^{L-1}(a_j^L - y_j)$$

③ $$\frac{\partial C}{\partial b} = \frac{1}{n}\sum(\sigma - y) \rightarrow \frac{\partial C}{\partial b_j^L} = \frac{1}{n}\sum_x (a_j^L - y_j)$$

P2

choose cost function to eliminate x_j

goal: $\frac{\partial C}{\partial w} = x_j(a-y)$

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial a}\cdot\frac{\partial a}{\partial w} = \frac{\partial C}{\partial a}\cdot\sigma'\cdot x_j$$

$$\frac{\partial C}{\partial a} = \frac{(a-y)}{\sigma' x_j}$$

↓ $\sigma' = \sigma(z)(1-\sigma(z)) = a(1-a)$

$$\frac{\partial C}{\partial a} = \frac{(a-y)}{a(1-a)x_j}$$

$$C = \frac{1}{x_j}[y\ln a + (1-y)\ln(1-a)] + \text{Constant}$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial a}\cdot\frac{\partial a}{\partial b} = \frac{(a-y)}{a(1-a)x_j}\cdot\sigma'$$

$$= \frac{1}{x_j}(a-y)$$

Cost function

$$C = -\ln a_y^L$$

$$\frac{\partial C}{\partial b_j^L} = \frac{\partial C}{\partial a_j^L}\cdot\frac{\partial a_j^L}{\partial z_j^L}\cdot\frac{\partial z_j^L}{\partial b_j} \overset{j=k}{=} \frac{-1}{a_j^L}\cdot\frac{e^{z_j^L}}{\sum_k e^{z_k^L}}\left(1 - \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}\right) = a_j^L - 1$$

$$\overset{j\neq k}{=} -\frac{1}{a_j^L}\cdot\frac{(e^{z_j^L})^2}{(\sum_k e^{z_k^L})^2} = a_j^L$$

$$\Rightarrow \frac{\partial C}{\partial b_j^L} = a_j^L - y$$

$$\frac{\partial C}{\partial w_{jk}^L} = \frac{\partial C}{\partial a_j^L}\cdot\frac{\partial a_j^L}{\partial z_j^L}\cdot\frac{\partial z_j^L}{\partial w_{jk}^L} \overset{j=k}{=} (a_j^L - 1)\cdot x_k^{L-1}$$

$$\overset{j\neq k}{=} a_j^L \cdot x_k^{L-1}$$

$x_k^{L-1} = a_k^{L-1}$

$$\Rightarrow \frac{\partial C}{\partial w_{jk}^L} = (a_j^L - y)\cdot a_k^{L-1}$$

Backpropagation

$$\delta_j^L \equiv \frac{\partial C}{\partial z_j^L}$$

$$= \frac{\partial C}{\partial a_j^L}\cdot\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial b_j^L} = a_j^L - y_j$$

Regularization

L2

$C = C_0 + \frac{\lambda}{2n}\sum_w w^2$

C_0: original cost function

$$\frac{\partial C}{\partial w^l_{jk}} = \frac{\partial C_0}{\partial w^l_{jk}} + \frac{\lambda}{n} w^l_{jk}$$

$$\frac{\partial C}{\partial b^l_j} = \frac{\partial C_0}{\partial b^l_j}$$

learning →

$$b^l_j = b^l_j - \eta \frac{\partial C_0}{\partial b^l_j}$$

$$w^l_{jk} = w^l_{jk} - \eta \frac{\partial C_0}{\partial w^l_{jk}} - \frac{\eta\lambda}{n} w^l_{jk}$$

$$= \left(1 - \frac{\eta\lambda}{n}\right) w^l_{jk} - \eta \frac{\partial C_0}{\partial w^l_{jk}}$$

weight decay

mini-batch →

$$w = \left(1 - \frac{\eta\lambda}{n}\right) w - \frac{\eta}{m}\sum_x \frac{\partial C_x}{\partial w}$$

$$b = b - \frac{\eta}{m}\sum_x \frac{\partial C_x}{\partial b}$$

m: training set, $\sum_x$: mini-batch size

L1

$C = C_0 + \frac{\lambda}{n}\sum_w |w|$

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \frac{\lambda}{n}\,\mathrm{sgn}(w)$$

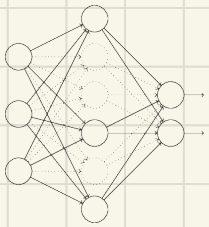
learning →

$$w^l_{jk} = w^l_{jk} - \frac{\eta\lambda}{n}\,\mathrm{sgn}(w^l_{jk}) - \eta \frac{\partial C_0}{\partial w^l_{jk}}$$

mini-batch →

$$w = w - \frac{\eta\lambda}{n}\,\mathrm{sgn}(w) - \frac{\eta}{m}\sum_x \frac{\partial C_x}{\partial w}$$

Dropout



① delete half of neurons

② forward $x \to a$

backpropagate $\nabla C \leftarrow a$

③ $w = w - \eta \nabla C$

$b = b - \eta \nabla C$

④ restore dropout neurons

new batch ←

Weight initialization

$z = \sum w_j x_j + b$

$w_j \sim N(0, 1)$

$b \sim N(0, 1)$

$n = 1000$ input → $\frac{1}{2} : 1$, $\frac{1}{2} : 0$

$\sum w_j x_j \sim N(0, (500 \cdot 1)^2)$ not the same w_j

~~$+b \sim N(0, (500 \cdot 1)^2 + 1^2)$~~

$+b \sim N(0, 501 \cdot 1^2)$

$z \sim N(0, (\sqrt{501})^2)$

$w_j \sim N(0, (\frac{1}{\sqrt{n}})^2)$

$b \sim N(0, 1)$

$n = 1000$

half 1

half 0

$\sum w_j x_j \sim N(0, 500 \cdot \frac{1}{1000})$

$+b \sim N(0, \frac{1}{2} + 1)$

$z \sim N(0, (\sqrt{\frac{3}{2}})^2)$

Compare L2

$w \sim N(0, 1)$

$w = (1 - \frac{\eta\lambda}{n}) w - \frac{\eta}{m} \sum_x \frac{\partial C_x}{\partial w}$ (mini-batch size)

$b = b - \frac{\eta}{m} \sum_x \frac{\partial C_x}{\partial b}$

weight decay : $\frac{\eta\lambda}{n} w + \frac{\eta}{m} \sum_x \frac{\partial C_x}{\partial w}$

P4

- λ too small, weight decay → first epoch learning
- $\eta\lambda \ll n$, decay → $e^{-\frac{\eta\lambda}{m}}$
- λ too large, $w \approx \frac{1}{\sqrt{n_w}}$ → no decay

$\frac{\partial C_x}{\partial w}$ random, no use

epoch $m_1 \to m_2$

近似 $\frac{n}{m} : \lambda$

$(1 - \frac{\eta\lambda}{n})^{\frac{n}{m}}$

$n \gg \eta\lambda$

$\lim_{n \to \infty} (1 + \frac{1}{n})^n = e$

$\to \lim_{n \to \infty} (1 + (-\frac{\eta\lambda}{n}))^{(-\frac{n}{\eta\lambda}) \cdot (-\frac{\eta\lambda}{m})}$

$= e^{(-\frac{\eta\lambda}{m})}$

i	i	
0	0	
0	0	0
0	0	0
0	0	0

$i \cdot j + j \cdot k + \cdots = n_w$

$x \; a^1 \; a^2$

$x \to \sigma(wx + b) \to \sigma(w[\sigma(w \sum_i x + b^1)] + b^2)$

$\frac{\partial C}{\partial w} = 0 \to \frac{\lambda}{2n} \sum w^2 + C_x = C$ (C constant, C_x constant)

a^{L-1} → $\frac{\partial C}{\partial b} = 0$ → b = constant

$C_x = \frac{1}{2n} \sum (y - a^L)^2$ (a^L constant)

C_x = constant

?

Early out

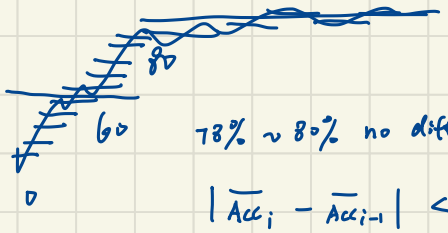
no - improvement - in - n epochs

PS

1) epochs → w b → validation accuracy

simply check if $i > i-n$

2)

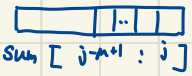


78% ~ 80% no differences

$|\overline{Acc}_i - \overline{Acc}_{i-1}| < \Delta_{Acc}$ stop at j

$\overline{Acc} = \frac{\sum_{M_{Acc}} Acc}{M_{Acc}}$

if nepoch > M_{Acc} start

for each epoch j $\sum Acc =$  Sum [j-m+1 : j]

$\frac{\quad}{M_{Acc}}$

if len c

too specefic → rough!

Get Hyper parameter

1. Broad strategy ① small training data & validation data (0 hidden neurons) ⟶ fast feedback

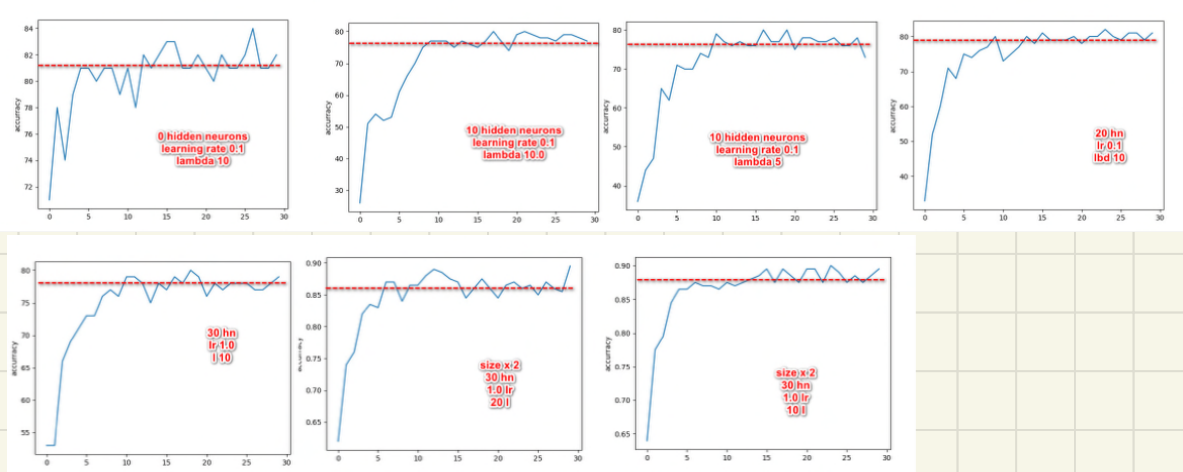
② λ ↓ same rate

③ learning rate η ↓ or ↑ to best

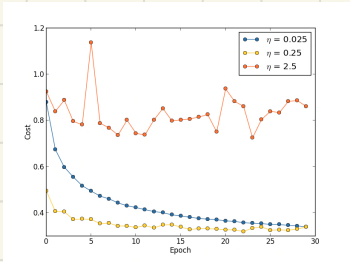
④ λ ↓ or ↑ to best

⑤ + hidden neurons

⑥ ③ ∨ ⑤ → best



2. learning rate η : same epochs
mini-batch-size
λ
} compare



① choose estimate η

② if Cost ↓

η ↑ ×10 , ×100

if Cost ↑ , NW

η ↓ ×10, ×100

larger η , C ↓

③ η ↓ , not to slow

3. regularization parameters λ

① λ=0 → η

② intial λ=1

③ λ+=10 / λ−=10

5. auto select

Random search for hyper-parameter optimization
grid search

Practical Bayesian Optimization of Machine Learning Algorithms

4. mini batch size × small specific
× big × weights change

Hessian Technique

goal: choose $\Delta w \to \min\left(C(w+\Delta w)\right)$

$$C(w+\Delta w) = C(w) + \sum \frac{\partial C}{\partial w_j}\Delta w_j + \frac{1}{2}\sum \frac{\partial^2 C}{\partial w_j \partial w_k}\Delta w_j \Delta w_k + \cdots$$

$$= C(w) + \nabla C\,\Delta w + \frac{1}{2}\Delta w^T H \Delta w + \cdots$$

converge > standard gradient descent

$$\approx C(w) + \nabla C\,\Delta w + \frac{1}{2}\Delta w^T H \Delta w$$

($C(w)$: constant)

$$\frac{\partial}{\partial \Delta w} = \nabla C^T + \frac{1}{2}(H + H^T)\Delta w = 0$$

$$\nabla C + \frac{1}{2}\Delta w^T (H^T + H) = 0$$

($\frac{\partial^2 C}{\partial j \partial k} \to H^T = H$)

$$\nabla C + \frac{1}{2}\Delta w^T 2H = 0$$

$$\Delta w = -\frac{\nabla C}{H}$$

Logic:

① w

② $w' = w - H^{-1}\nabla C$

③ $w'' = w' - H'^{-1}\nabla C'$

. . . .

Momentum-based gradient descent

velocity

$v' = \mu v - \gamma \nabla C$

$w' = w + v'$

① $\mu = 0 \to w' = w - \gamma \nabla C$

② $\mu = 1 \to$ 1) $\Delta w = v - \gamma \nabla C$ > ①

2) $\Delta w = v - \gamma \nabla C - \gamma \nabla C$ < Δw_1 (underbrace: $< v$)

. . .

slow

③ $0 < \mu < 1 \to$ 1) $\Delta w = \mu v - \gamma \nabla C$ $0 <$ <③

2) $\Delta w = \mu(\mu v - \gamma \nabla C) - \gamma \nabla C$ (underbrace: $< v$) ①< << ② gentle slow

... $\Delta w \to$ ①

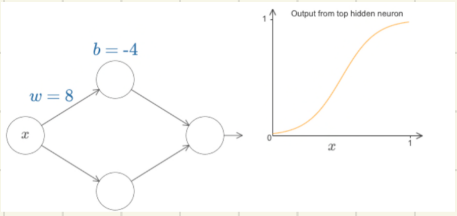
④ $\mu < 0 \to$ 1) $\Delta w = \mu v - \gamma \nabla C$ $\Delta w <$ ① too slow

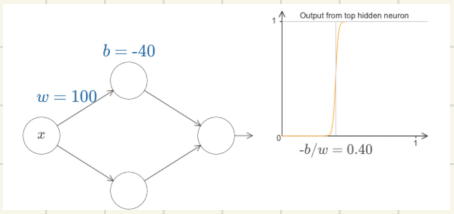
2) $\Delta w = \mu(\mu v - \gamma \nabla C) - \gamma \nabla C$ $\Delta w <$ ①

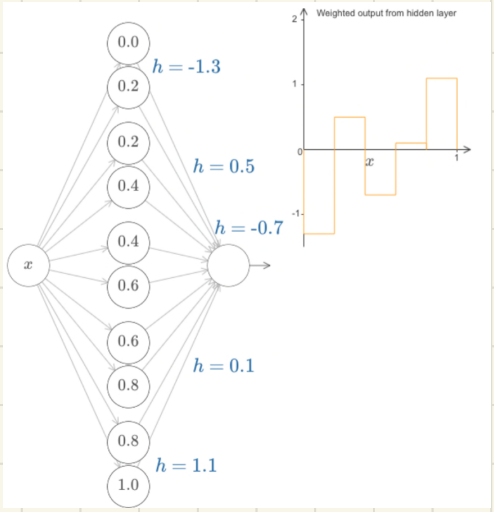
⑤ $\mu \geq 1 \to$ 1) $\Delta w = \mu v - \gamma \nabla C$ >> ② too fast

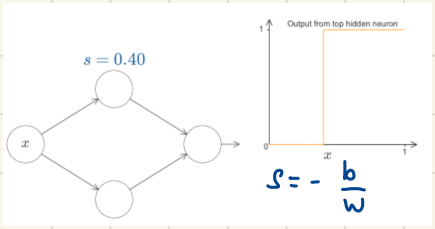
2) $\Delta w = \mu(\mu v - \gamma \nabla C) - \gamma \nabla C$ >>> ②

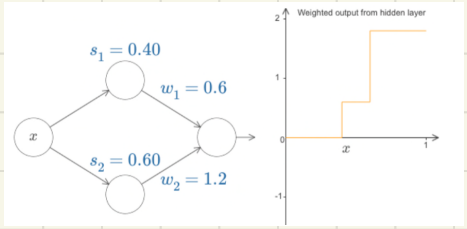
Visual Learning

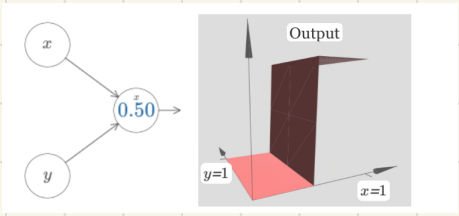


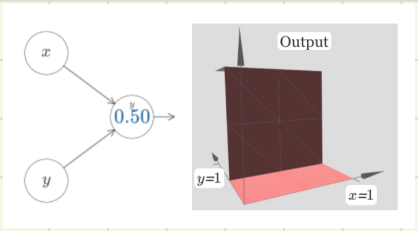


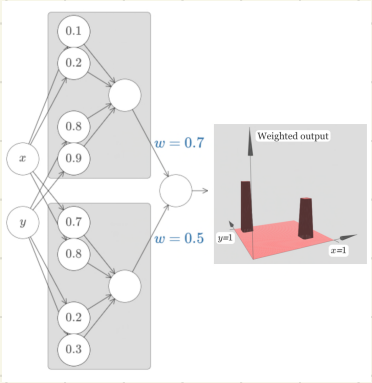


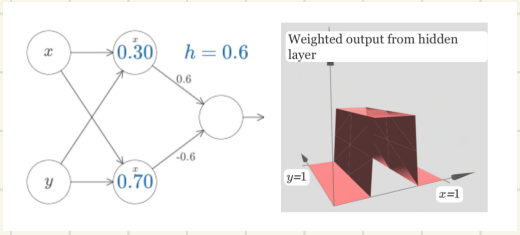


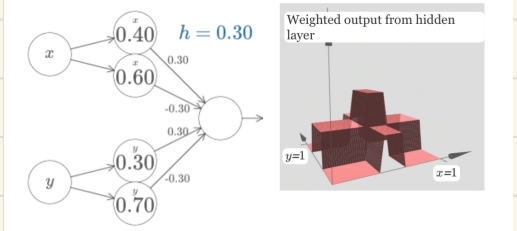






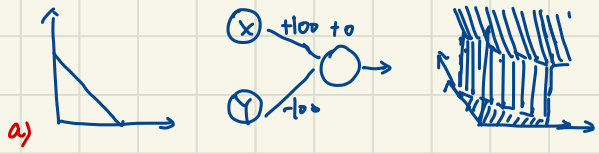




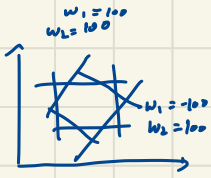


P6

a)



b)



P7

a) $w \to \infty$

$x \to 0^+$

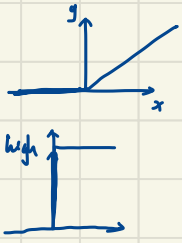
$\Sigma wx+b \to \infty$

$a \to \infty$

$x \to 0^-$

$\Sigma wx+b \to -\infty$

$a \to 0$



b) $S(z) = z$

$w \to \infty$

$x \to 0^+$

$a \to \infty$

$x \to 0^-$

$a \to -\infty$