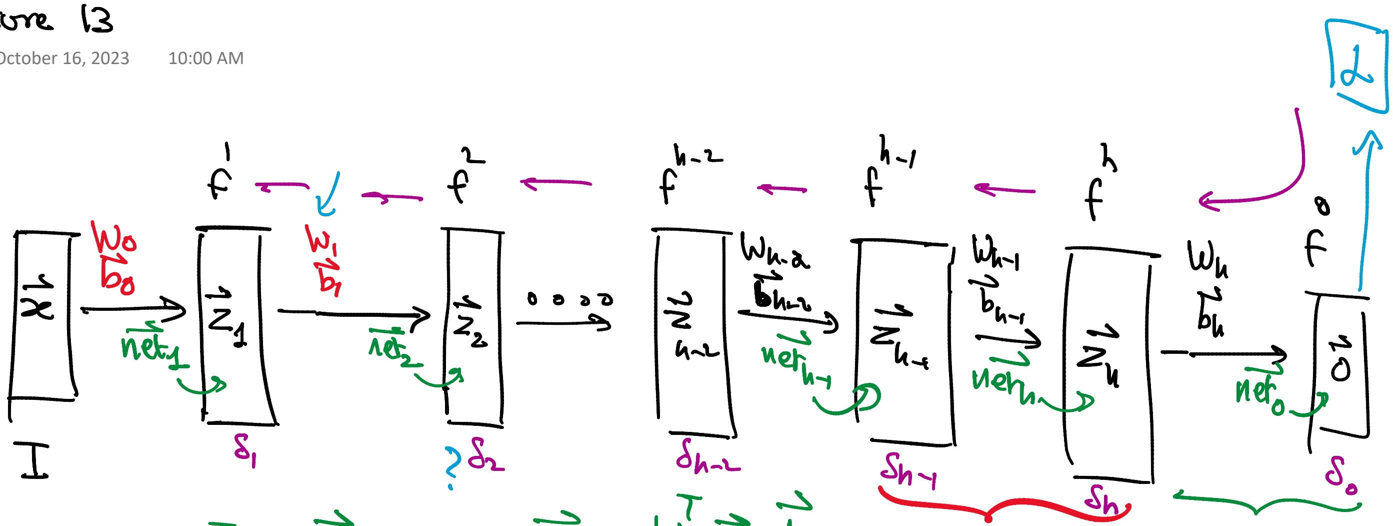
Lecture 13

Monday, October 16, 2023 10:00 AM



$$\vec{net_1} = \underline{W_0^T \vec{x} + \vec{b_0}}$$

$$\vec{z_1} = f^1(\vec{net_1})$$

$$\vec{net_2} = \underline{W_1^T \vec{z_1} + \vec{b_1}}$$

$$\vec{z_2} = f^2(\vec{net_2})$$

$$\vec{net_o} = \underline{W_h^T \vec{z_h} + \vec{b_h}}$$

$$\vec{o} = f^o(\vec{net_o})$$

$$\vec{net_h} = W_{h-1}^T \vec{z}_{h-1} + \vec{b}_{h-1}$$

$$\vec{z_h} = f^h(\vec{net_h})$$

1) feed-forward

$\vec{x} \longrightarrow \vec{o}$

given input   compute the output

2) Loss / Error

3) backpropagation

Compute the gradients of all W_i, $\vec{b_i}$

∇_{W_i}, $\nabla_{\vec{b_i}}$ $\forall\ i = 0 \ldots h$

$$\nabla W_1 = \frac{\partial \mathcal{L}}{\partial W_1}$$

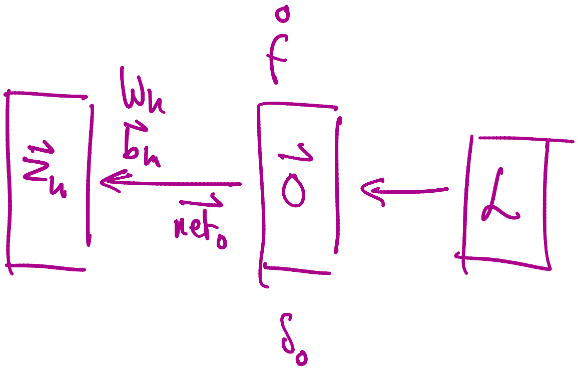
$$\nabla_{W_i} = \frac{\partial \mathcal{L}}{\partial W_i} \qquad \nabla_{b_i} = \frac{\partial \mathcal{L}}{\partial b_i}$$

weight gradient   bias gradient

Computational graph approach

net gradient for each layer

$$\nabla_{net_i} = \boxed{\vec{\delta}_i} = \frac{\partial \mathcal{L}}{\partial net_i}$$



$$\vec{\delta}_o = \frac{\partial \mathcal{L}}{\partial \vec{net}_o}$$

$$= \frac{\partial \vec{o}}{\partial \vec{net}_o} \cdot \frac{\partial \mathcal{L}}{\partial \vec{o}}$$

$$= \boxed{\underbrace{\frac{\partial \overset{o}{f}(\vec{net}_o)}{\partial \vec{net}_o}}_{\text{derivative of the activation function } f^o} \cdot \underbrace{\frac{\partial \mathcal{L}}{\partial \vec{o}}}}$$

$$\underbrace{\vec{1} \cdot (\vec{o} - \vec{y})}$$

$$\boxed{\vec{\delta}_o = \vec{o} - \vec{y}}$$

$$\vec{o} = \overset{o}{f}(\vec{net}_o)$$

$$\vec{net}_o = W_h^T \vec{z}_h + \vec{b}_h$$

e.g.

squared error $\quad \mathcal{L} = \frac{1}{2} \|\vec{y} - \vec{o}\|^2$

$$\frac{\partial L}{\partial \vec{o}} = \frac{1}{2}\left(2 \cdot (\vec{y} - \vec{o})\right) \cdot -1$$

$$= \vec{o} - \vec{y}$$

predicted value $\quad$ true value

$\overset{o}{f} \equiv$ linear

$$\partial \overset{o}{f}(net_o) = \partial net_o = \vec{1}$$

$$\frac{\partial \hat{f}(net_o)}{\partial net_o} = \frac{\partial net_o}{\partial net_o} = \vec{1}$$

e.g. $\hat{f} =$ softmax

$L =$ CE loss (cross entropy)

$$L = -\sum_{j=1}^{k} y_j \log(o_j)$$

$\vec{y} \equiv$ one-hot vector

$\vec{o} \equiv$ vector of probabilities for k classes

$$\vec{y} = (0\,0\,..\,1\,0\,0\,0) = (y_1, y_2 \,..\, y_k)^T$$

$$\vec{o} = (o_1, o_2, .. \, o_k)^T$$

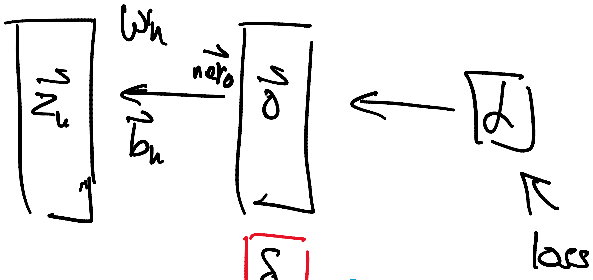
$$\frac{\partial L}{\partial \vec{o}} = \begin{bmatrix} \frac{\partial L}{\partial o_1} \\ \frac{\partial L}{\partial o_2} \\ \vdots \\ \frac{\partial L}{\partial o_k} \end{bmatrix} = \begin{bmatrix} -y_1/o_1 \\ -y_2/o_2 \\ \vdots \\ -y_k/o_k \end{bmatrix}$$

$$\frac{\partial L}{\partial o_1} = -\frac{y_1}{o_1}$$

$$\frac{\partial \, \text{softmax}(\vec{net_o})}{\partial net_o} = \begin{bmatrix} \text{matrix} \end{bmatrix}$$

$$\text{softmax}(net_i) = \frac{e^{net_i}}{\sum_{j=1}^{k} e^{net_j}}$$

$$\underbrace{\frac{\partial f(net_o)}{\partial net_o} \cdot \frac{\partial L}{\partial \vec{o}}}_{\vec{o} - \vec{y}}$$



$$\boxed{\delta_o}$$

$$\nabla W_h = \frac{\partial \mathcal{L}}{\partial W_h} = \underset{=}{\frac{\partial net_o}{\partial W_h}} \cdot \underbrace{\frac{\partial f'(net_o)}{\partial net_o} \cdot \frac{\partial \mathcal{L}}{\partial \vec{o}}}_{\vec{\delta}_o}$$

$\vec{o}$

$$= \frac{\partial W_h^T \vec{z}_h + b_h}{\partial W_h} \cdot \vec{\delta}_o$$

$$\nabla W_h = \underbrace{\left(\underset{m_h \cdot 1}{\underline{\vec{z}_h}} \cdot \underset{1 \cdot k}{\vec{\delta}_o^{\,T}} \right)}_{m_h \cdot k}$$

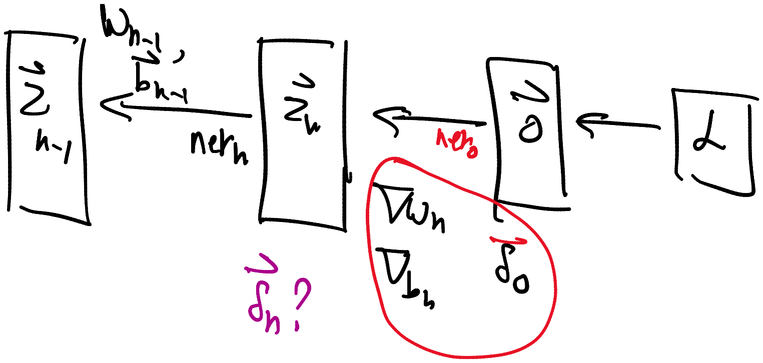
$$W_h \in \mathbb{R}^{m_h \times k}$$

$m_h =$ size of $\vec{z}_h$

$k =$ # of output neurons

$$\nabla b_h = \frac{\partial (W_h^T z_h) + b_h}{\partial b_h} \cdot \vec{\delta}_o$$

$$= \vec{1} \cdot \vec{\delta}_o = \vec{\delta}_o$$



$$\vec\delta_h = \frac{\partial L}{\partial net_h} = \boxed{\frac{\partial \vec z_h}{\partial net_h}} \; \frac{\partial net_o}{\partial \vec z_h} \; \boxed{\underbrace{\frac{\partial L}{\partial net_o}}_{\vec\delta_o}}$$

$$\underbrace{\frac{\partial f^h(net_h)}{\partial net_h}}_{\text{derivative of activation for hidden layer}}$$

$$\frac{\partial W_h^T \vec z_h + \vec b_h}{\partial \vec z_h} = W_h$$

$$\boxed{= \frac{\partial f^h}{\partial net_h} \odot \left(W_h \cdot \delta_o \right)}$$

element-wise product

Backprop:

1) Compute $\vec\delta_o = \frac{\partial f^o}{\partial net_o} \cdot \frac{\partial L}{\partial \vec o}$ $\left(\vec o - \vec y\right)$

$L =$ squared error
$f^o =$ linear

Compute net gradients

2) for $l = h, h-1, \ldots, 1$

$f = $ linear

$\mathcal{L} = $ CE
$f^o = $ softmax

$\mathcal{L} = $ BCE
$f^o = $ sigmoid

2) for $l = h, h-1, \dots, 1$

$$\tilde{\delta}_l = \frac{\partial f^l}{\partial net_l} \odot \left(W_l \cdot \tilde{\delta}_{l+1} \right)$$

3) Compute weight & bias gradient

$$\nabla_{W_l} = \underbrace{z^l \cdot (\delta^{l+1})^T}_{\text{outerproduct}} \leftarrow \text{matrix}$$

$$\nabla_{b_l} = \tilde{\delta}^{l+1}$$

4) gradient descent

$$W_l^{(t+1)} = W_l^{(t)} - \eta \cdot \nabla W_l$$

$$b_l^{(t+1)} = b_l^{(t)} - \eta \cdot \nabla_{b_l}$$

NN training

```
for e = 1 -- max-epochs          ← pass over the data
over D
    pick a minibatch, extreme case $B=1$
        1) forward
        2) backprop
```

$\mathcal{L}$

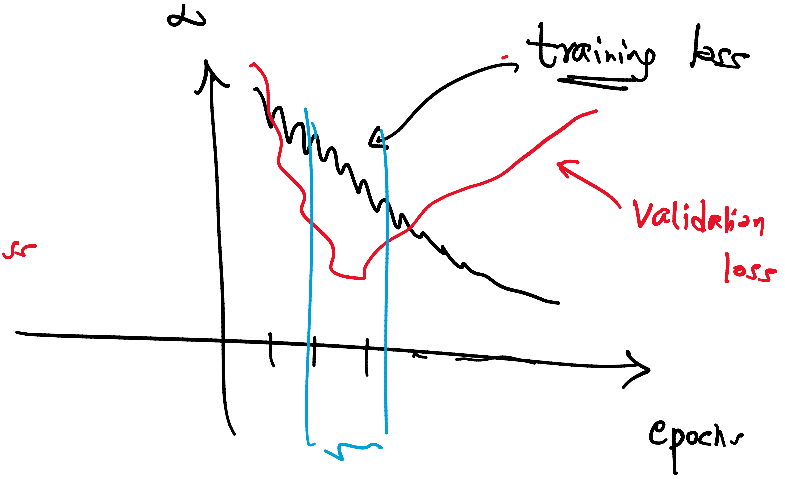
$\uparrow$

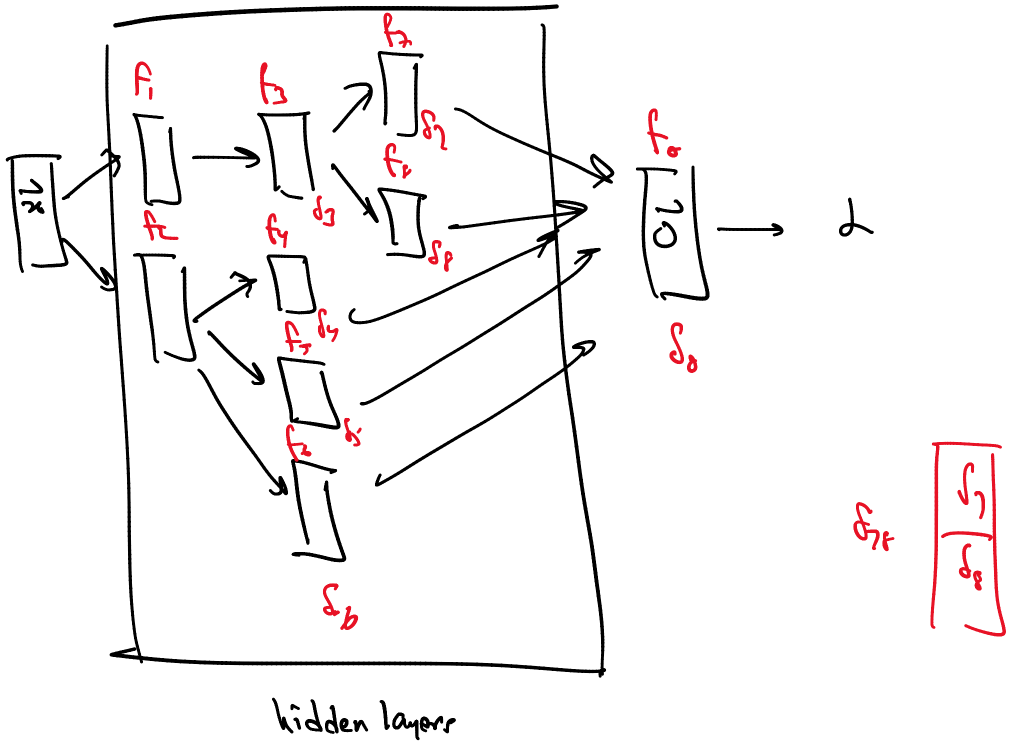
training loss

L

When to stop?

look @ Validation loss

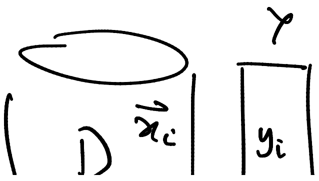




Naive Bayes Classifier

↳ Bayes classifier

↳ probabilistic classifier



$y_i \in \{ c_1 \quad c_2 \ \ldots \ c_k \}$

$$D \quad \vec{x}_i \;\Big|\; y_i \qquad\qquad \underbrace{y_i \in \{C_1 \;\; C_2 \; \ldots \; C_k\}}_{k\text{-way classification}}$$

Posterior probability

$$\begin{tikzcd} & P(C_1|\vec{x}) \\ P(C_i|\vec{x}) \arrow[ur] \arrow[r] \arrow[ddr] & P(C_2|\vec{x}) \\ & \vdots \\ & P(C_k|\vec{x}) \end{tikzcd}$$

<u>Optimal</u> decision rule:

$$C_* = \underset{i=1}{\overset{k}{\arg\max}} \left\{ \underset{\text{Unknown}}{\underline{\underline{P(C_i|\vec{x})}}} \right\}$$

a) Logistic regression

b) <u>Bayes</u> classifier

$$\underset{\text{Posterior}}{P(C_i|\vec{x})} = \boxed{\frac{\overset{\text{likelihood}}{P(\vec{x}|C_i)} \cdot \overset{\text{Prior} \checkmark}{P(C_i)}}{\underset{\text{evidence}}{P(\vec{x})}}} = \frac{P(\vec{x}|C_i)\cdot P(C_i)}{\sum_{j=1}^{k} P(\vec{x}|C_j)\cdot P(C_j)}$$

$$D \quad (\vec{x}_i, y_i) \rightarrow \boxed{\begin{array}{l} D_1 = \{\vec{x}_i \mid y_i = c_1\} \\ D_2 = \{\vec{x}_i \mid y_i = c_2\} \\ \vdots \\ D_k = \{\vec{x}_i \mid y_i = c_k\} \end{array}} \qquad \begin{array}{ll} n_1 = |D_1| & \hat{P}(c_1) = n_1/n \\ n_2 = |D_2| & \hat{P}(c_2) = n_2/n \\ \vdots & \\ n_k = |D_k| & \hat{P}(c_k) = n_k/n \end{array}$$

$$P(c_i) = \frac{n_i}{n} \leftarrow \frac{\#\text{ in class } c_i}{\#\text{ of total points}}$$

$$P(\vec{x} \mid c_i) = \begin{cases} \rightarrow \text{a) } \underline{\text{parametric}} \ (\text{model assumptions}) \\ \rightarrow \text{b) non-parametric } (\text{no prior assumption}) \end{cases}$$

<u>Parametric</u> : assume class c_i is normally distributed

$$\underline{\underline{P(\vec{x} \mid c_i)}} \equiv N(\vec{x} \mid \underbrace{\overset{\uparrow}{\vec{\mu}_i}, \overset{\uparrow}{\Sigma_i}}_{\text{for } c_i})$$

mean   cov

Unknown

$\vec{x}_i \in \mathbb{R}^d$

$$\vec{\mu}_i = \text{mean}(D_i) \qquad d$$

$$\Sigma_i = \text{Cov}(D_i) \qquad d \times d$$

$$\begin{tikzcd} & D_1 & n_1 \\ D \arrow[ur] \arrow[r] \arrow[dr] & D_2 & n_2 \\ n & \vdots & \end{tikzcd}$$

$\Sigma_i = \mathrm{cov}(D_i) \quad d \times d$

$n \searrow \vdots \quad D_k \quad n_k$

1) Class imbalance

$n_i << n$

2) dimensionality $\underline{\underline{d}}$

e.g. $d = 1000$

$\Sigma_i \approx 10^6$ parameters

$\boxed{O(d^2) \text{ vs } n}$ vs

$$P(c_i|x) \cong \frac{N(\vec{x} \mid \mu_i, \Sigma_i) \cdot \frac{n_i}{n}}{P(\vec{x})}$$

$$c^* = \operatorname{argmax} \left\{ \frac{N(\vec{x} \mid \mu_i \Sigma_i) \, n_i}{\boxed{P(\vec{x}) \cdot n}} \right\}$$

$$\hat{y} = \underset{\substack{\uparrow \\ \text{predicted class}}}{c^*} = \operatorname{argmax} \left\{ N(\vec{x} \mid \underline{\mu_i}, \underset{\substack{\uparrow \\ d \times d}}{\underline{\Sigma_i}}) \cdot \underline{n_i} \right\}$$

<u>Naive Bayes</u>

$P(\vec{x}|c_i) \equiv$ joint prob $P((x_1 \, x_2 \, .. \, x_d)^T | c_i)$

$$P(\vec{x}|c_i) \equiv \text{joint prob } \underline{P\left((x_1\ x_2 \ldots x_d)^T \middle| c_i\right)}$$

$$\Downarrow$$

assume all attributes are independent

$$= P(x_1|c_i) \times P(x_2|c_i) \times \cdots \times P(x_d|c_i)$$

$$P(\vec{x}|c_i) = \prod_{j=1}^{k} P(x_j|c_i)$$

$$\Downarrow$$

approx via univariate normal

$$= \prod_{j=1}^{k} N_j(\underline{x_j} \mid \underset{\text{mean}}{\underline{\mu_{ij}}}, \underset{\text{var}}{\underline{\sigma_{ij}^2}})$$

normal per attribute per class

j = attribute

i = class

diagonal cov!

$$\Sigma_i = \begin{bmatrix} \underline{\sigma_{i1}^2} & & 0 \\ & \underline{\sigma_{i2}^2} & \\ & & \ddots & \\ 0 & & & \underline{\sigma_{id}^2} \end{bmatrix}$$

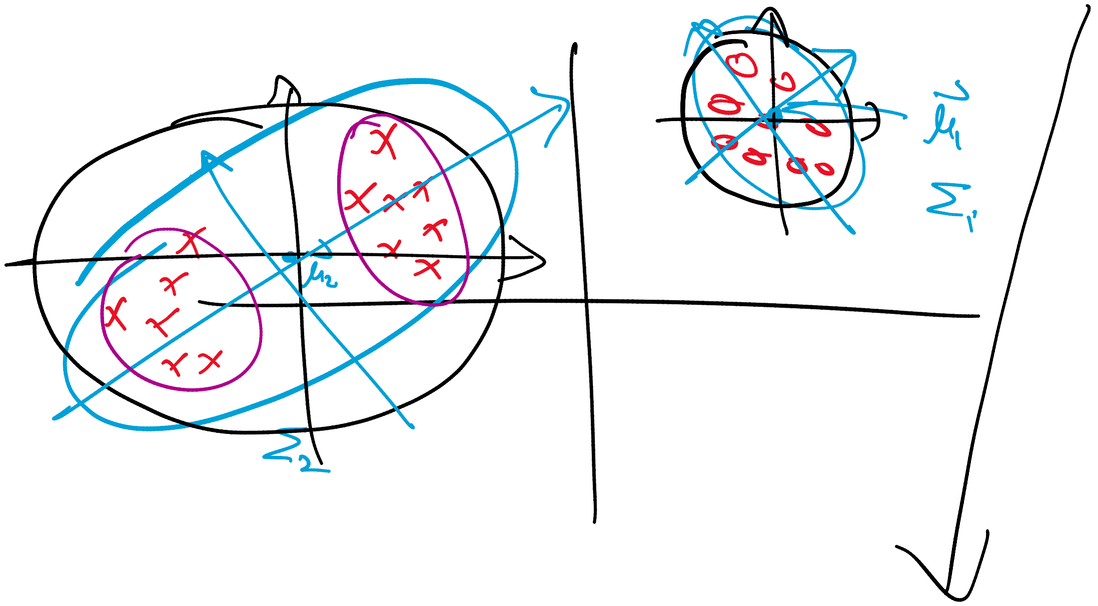
d value to be estimated

μ_{ij} = mean

σ_{ij}^2 = variance

$$P(c_i|\vec{x}) = \frac{\boxed{P(\vec{x}|c_i)} \cdot \overset{\checkmark}{P(c_i)}}{P(\vec{x})}$$

likelihood a) $p(\vec{x}|c_i) = \underline{N(\vec{x}|\mu_i, \Sigma_i)}$



Bayes approach

($\text{full } \Sigma_i$)

Naive Bayes

(diagonal Σ_i)

axis aligned

mixture of Gaussians