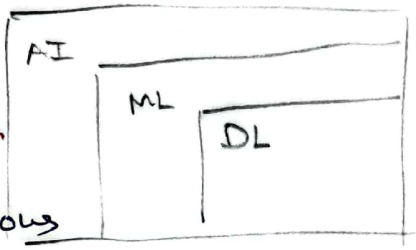
L1: What & Why?

* Machine learning is a subfield of computer science, that evolved from the study of pattern recognition & computational learning theory.
  - use Algos to iteratively learn from data.
  - Allows computers to discover patterns, without explicitly programming where to look.

Q. What IS machine learning?

Ability of computers to learn from data.




learn → - supervised learning
- unsupervised learning
- Reinforcement learning

data → from various sources. (with noise redundancy)

1) Supervised learning:-

We got training data; with labelled datums.

Goal is to predict labels on new datums.

2) Unsupervised:- data studied through clustering, mining associations, dimen. redu!

→ we know nothing about fruits. only their parameters.
- color
- size
- shape.

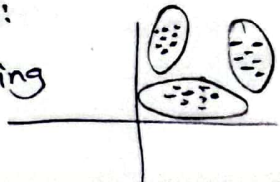
→ we got to group fruits of same kind.
→ no meaning of labels.

* both approaches have tradeoffs. ① → human annotated data
② → more computation.

→ Three canonical learning problems:-

1) Regression - Supervised:- when predicting a continuous value. not only linear. even quadratic.
2) Classification - Supervised:- predict color of fruit; translation of english.
3) Unsupervised learning:

i) clustering



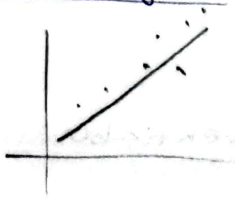
(ii) dim. redn

L2.1

* In supervised learning; the desired outputs are provided on training data. where-in Unsup. learning; data is analysed & studied through clustering, mining associations, dimensionality reduction etc. into different classes.

* 3 canonical learning problems:

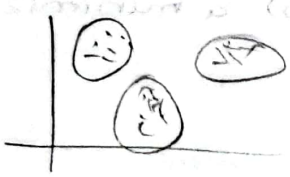
1. Supervised regression



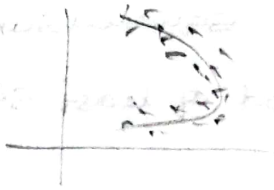
2) Supervised classification:

3. Unsupervised learning:-

• clustering



• dim. red$^{n}$



→ Supervised learning:-

Trainig data $\{(x^i, y^i) \in X * Y\}$

Function $F : X \longrightarrow Y$

Learning:- finding $\hat{f} \in F$

s.t. $\hat{f}(x^i) \approx y^i$ here we have concept of error.

Prediction:- $y = \hat{f}(x)$ → new data 'x'.

1) Noise in data:-

- data might have noise, random human errors.
- ∴ Most methods deal with expectations as they minimize effect of error.
- Better to find outliers & clean data in 1st step. "data clensing"

2) How to predict?

- Curve fitting! Linear or non-linear.
- Criteria to compare two curves for a dataset.

  errorfunction!

∴ we describe an Errorfunction $E(f, D)$ which returns real no.

Examples for some E:-

$\sum_D f(x_i) - y_i$ ✗

$L_1$ norm ↝ $\sum_D |f(x_i) - y_i|$ okay; not differentiable

$L_2^2$ norm ↩ $\sum_D (f(x_i) - y_i)^2$ okay. differentiable! cool.

Hence we choose this squared sum as $E(f, D)$ & minimize this.

"method of least squares".

## L2.2 :- Regression

Linear regression as a canonical example :-

↳ we got
- regularization → Ridge, Lasso, Bayesian
- non-parametric estimation
- non-linearity through Kernels.

* Regression is about learning to predict a set of output vars (dependent) as a function of set of input variables (independent).

Eg: $x_i$ → money spent on advertising

$y_i$ → sales increased.

then we model

$$y = \beta_0 + \beta_1 x.$$

& we find $\beta_0, \beta_1$ by least squares.

then $y^* = \beta_0 + \beta_1 x^*$ ✓.

### Basic Notation:-

* Dataset $D = \langle x_1, y_1 \rangle \langle x_2, y_2 \rangle \cdots \langle x_m, y_m \rangle$

$m$ = no. of training samples

$x$'s = input / training vars

$y$'s = output / target vars

sales as a non-linear function of investing!

* $\phi_i$'s are the attribute/basis functions. & let

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_p(x_1) \\ \phi_1(x_2) & & & \\ \vdots & & \ddots & \\ \phi_1(x_m) & & \cdots & \phi_p(x_m) \end{bmatrix}_{m \times p}$$

- radial basis
- polynomial basis $(1, x, x^2, \cdots)$
- fourier basis
- wavelet basis.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{m \times 1} \qquad \phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_p(x) \end{bmatrix}$$

∴ not linear in $x$.

* imp not linear. hmmm....

→ General regression problem:-

* Determine $f^*$ such that $f^*(x)$ is the best predictor of $y$ wrt $D$.

$$f^* = \underset{f \in F}{\operatorname{argmin}}\ E(f, D)$$

set of all possible functions over which error minimized. (linear or non)

→ Parameterized Regression problem:- not linear.

* Determine parameter $w$ for function $f(\phi(x), w)$ which minimizes $E(f(\phi(x), w), D)$.

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$$

$$w^* = \underset{w}{\operatorname{argmin}}\ E(f(\phi(x), w), D)$$

$$\phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_p(x) \end{bmatrix}$$

---

→ Types of regression:-

** classified based on function class and error function. neat!

* Linear regression ⇒ F is space of linear functions.

$$f(\phi(x), w) = w^T \phi(x) + b$$

bias. may or may not exist.

problem is

$$w^* = \underset{w}{\operatorname{argmin}}\ E(\phi(x), w, D)$$

we introduce $\tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}$, $\tilde{\phi} = \begin{bmatrix} \phi \\ 1 \end{bmatrix}$

* Ridge regression:- ⇝ $E(\cdot) = \sum_1^m (y_i - w^T \phi(x_i)) + \|w\|_2^2$

A shrinkage parameter is added to error function to reduce discrepancies due to variance.

Logistic regression:-

Models conditional probability of output variable, given independent vars. & used extensively in classification tasks.

$$f(\phi(x), w) = \log \underbrace{\frac{Pr(y|x)}{1 - Pr(y|x)}}_{\text{odds}} = b + w^T \phi(x)$$

$y$ is $\{0, 1\}$ but $\log \frac{Pr(y|x)}{1 - Pr(y|x)}$ is real valued.

more: lasso regression $E$ = sq. sum $+ \|w\|_1$
Stepwise regression
etc.

## Least Squares Solution:-

$$E(f,D) = \sum_{j=1}^{m} (f(x_j) - y_j)^2$$

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{j=1}^{m} \left( \left[ \sum_{i=1}^{p} w_i \phi_i(x_j) \right] - y_j \right)^2$$

→ case-i)

* if minimum value of squared loss is 0;

then $\forall u \quad \phi(x_u)^T \cdot w^* = y_u$ or equivalently $\Phi \cdot w^* = y$

$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix}$

$\Phi = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_p(x_1) \\ \vdots & & & \\ \phi_1(x_m) & & & \phi_p(x_m) \end{pmatrix}$

$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$

* $\Phi w^* = y$ has a solution, if $y$ is in the column space of $\Phi$

→ case-ii)

* if 0 loss were not attainable?

Geometric interpretation:

- let $y^*$ be a solution in column space of $\Phi$.
- then we need $y^*$ closest to $y$.
- Foot of perpendicular! yay!

$\therefore \quad (y - y^*)^T \cdot \Phi = 0$ since $y - y^*$ is $\perp^{ar}$ to every column vector.

$$(y^*)^T \cdot \Phi = y^T \cdot \Phi$$

$$(\Phi \cdot w^*)^T \Phi = y^T \Phi$$

$$(w^*)^T \cdot \Phi^T \cdot \Phi = y^T \cdot \Phi$$

$$\Phi^T \Phi w^* = \Phi^T y$$

$$\boxed{w^* = (\Phi^T \Phi)^{-1} \Phi^T y}$$

→ some notion of pseudo inverse.

** $\Phi^T \Phi$ is invertible iff $\Phi$ has full column rank.

PTO for proof:-

Proof:-

1. Say $\Phi$ doesn't have full column rank.

$\Rightarrow \exists x \neq 0$ s.t. $\Phi x = 0$

$\Rightarrow \Phi^T \Phi x = 0$

$\therefore \Phi^T \Phi$ is not invertible.

2. Say $\Phi^T \Phi$ is not invertible

$\Rightarrow \exists x \neq 0$ s.t. $\Phi^T \Phi x = 0$

$\Rightarrow x^T \Phi^T \Phi x = 0$

$\Rightarrow (\Phi x)^T \cdot (\Phi x) = 0$

$\Rightarrow \Phi x = 0$ with $x \neq 0$

$\Rightarrow \Phi$ doesn't have full column rank.

* Therefore

"least square solution $y^*$ is the orthogonal project of $y$ onto the column space of $\Phi$!"

In linear regression.

→ How about analytical derivation:-

- More generally, how to minimize a function
  - level curves & surfaces
  - Gradient vector
  - Directional derivative
  - Hyperplane
  - Tangential hyperplane
- Gradient descent algorithm.

## L3.1:- Linear regression, probabilistic interpretation & regularization:-

* owing to basis functions $\phi_j$, "linear regression" is linear in $w$ but not in $x$.
  - radial basis
  - fourier basis
  - wavelet basis
  - domain specific basis

  like $\phi_1(x) = 3x^2$ is perfectly alright.
  $\phi \rightarrow$ polynomial basis.

we found $w^* = \underset{w}{\text{argmin}}\ E(f(w,\phi), D)$.

$$w^* = (\Phi^T\Phi)^{-1}\Phi^T y$$

→ Probabilistic interpretation:- of ordinary least squares:-

* Gaussian error; MLestimation; addressing overfitting.

→ Linear model:-

* $y$ is a linear function of $\phi(x)$ subjected to a random noise variable $\varepsilon$ which we believe is 'mostly' bounded by threshold $\sigma$.

$$\underbrace{y}_{\text{expected output}} = \overbrace{w^T\phi(x)}^{\text{modeled output}} + \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0,\sigma^2)$$

$\therefore\ y \sim \mathcal{N}(w^T\phi(x), \sigma^2)$.

motivation:-

* $\mathcal{N}(\mu,\sigma^2)$ has maximum entropy among all real-valued distributions with specific variance $\sigma^2$.
* 3σ rule:-
  - 1σ away - 68%.
  - 2σ away - 95%.
  - 3σ away - 99.7%.

"we want a bounded; but as 'free' as possible distribution" for error.

* Now we need $w$; that most likely generated the data. $\hat{w}_{ML}$

$$Pr(D|w) = Pr(y|x,w) = \prod_{j=1}^{m} \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{(y_i - w^T\phi(x_i))^2}{2\sigma^2}\right)}$$

$$\hat{w}_{ML} = \underset{w}{\text{argmax}}\ Pr(D/w) = L(w|D)$$

$$\hat{w}_{ML} = \underset{w}{\text{argmax}}\ \underbrace{LL(w|D)}_{\text{loglikelihood}}$$

$$= \underset{w}{\text{argmin}} \sum_{j=1}^{m} (y_j - w^T\phi(x_j))^2$$

// same least squares solution.

→ Redundant $\phi$ and overfitting:-

Lets consider polynomial basis $\phi_i(x) = x^i$. with $\phi_1, \phi_2, \ldots \phi_t$.

* higher t can represent lower 't'.

But as $t \uparrow$, we try to fit every data point.

RMS error for train set decreases but test set increases.

"over fitting" → address via bayesian, or via regularisation. (add $\|w\|$ in error term) !!

* Empirically we observe, too many bends in curve ≡ high value of $w_i$'s.

∴ $\|w\|$ increases with increase in t.

* we want to address overfitting by restricting $\|w\|$.

add $\|w\|$ in error term. ← regularization! supress $\|w\|$.

---

1) By using Bayesian Linear Regression:-

- reasoning with some prior belief ($\|w\|$ is restricted in prior).

≡ use posterior over w as result.

we could find $\hat{w}_{MAP} = \underset{w}{\text{argmax}}\ Pr(w|D)$.

max aposterior probability

Intuitive prior:

each $w_i$ bounded by $\pm \frac{3}{\sqrt{\lambda}}$. (99%)

$Pr(w)$ s.t. $\|w\| \leq \frac{1}{\lambda}$. Gaussian again?

we'll have $w = \begin{bmatrix} w_1 \\ \vdots \\ w_t \end{bmatrix}$ where $w_i \sim N(0, \frac{1}{\lambda})$ Normal

$\lambda$ is called precision of the distribution.

* lets digress for bayesian estimation:-

parameter $\theta$

data $(\theta) \equiv D$

$$P(\theta|D) = \frac{P(D|\theta) \cdot P(\theta)}{\int_\theta P(D|\theta) \cdot P(\theta) d\theta}$$

(posterior ← P(θ|D); prior ← P(θ))

$\lambda$ is precision for $w_i$ in prior

$\sigma^2$ is error variance in data.

* conjugate prior means, if $P(x|\theta)$ has $d_1$ dist. & $P(\theta)$ has $d_2$ & $P(\theta|D)$ is also $d_2$ distribution; then $d_2$ is conjugate prior for $d_1$.

Eg: bernoulie r.v has β-distribution as conjugate prior

$$P(x|\theta) = \theta^x(1-\theta)^{1-x}$$

$$\therefore P(D|\theta) = \theta^{\Sigma x_i}(1-\theta)^{n-\Sigma x_i}$$

$$\text{Beta}(p;\alpha,\beta) \sim \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)} \quad \text{where } B(\alpha,\beta) = \int_{p=0}^{1} p^{\alpha-1}(1-p)^{\beta-1}\,dp$$

$$= \frac{\Gamma(\alpha)\cdot\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

$$\therefore P(\theta|D) = \frac{\theta^{\Sigma x_i}(1-\theta)^{n-\Sigma x_i}\cdot\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\text{normalizing const.}}$$

this is also Beta!

$$P(\theta|D) = \beta(\theta;\ \alpha+\Sigma x_i,\ \beta+n-\Sigma x_i)$$

Note:-

* for a Beta distribution $\beta(p;\alpha,\beta)$

$$E[p] = \frac{\alpha}{\alpha+\beta} \qquad \text{mode}(p) = \frac{\alpha-1}{\alpha+\beta-2}$$

* $\beta(p;1,1)$ is uniform distribution.

→ conjugate priors for gaussian dist:-

* say $P(x|\mu) \propto e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ ; $\sigma$ is a known value.

then conjugate prior is also a gaussian!

$$P(\mu) \propto e^{-(\mu-\mu_0)^2/2\sigma_0^2}$$

* now $P(\mu|D) \propto P(D|\mu)\cdot P(\mu)$

"product of gaussians is a gaussian."

say $P(\mu|D) \propto e^{-(\mu-\mu_m)^2/2\sigma_m^2}$

then we'll get

$$\frac{1}{\sigma_m^2} = m\cdot\frac{1}{\sigma^2} + \frac{1}{\sigma_0^2} \quad \text{(precisions are added)}$$

$$\frac{1}{\sigma_m^2}\cdot\mu_m = m\cdot\frac{\hat{\mu}_{MLE}}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}$$

MAP is $\mu_m$

where $\hat{\mu}_{MLE} = \text{mean}(x_1, x_2, \cdots x_n)$

* we'll see multivariat gaussian next.

* Multivarial Gaussian is

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \cdot e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

old friend

$x$: a vector of values.

$\Sigma$ is $\mathbb{R}^{n\times n}$ covariance matrix & positive-definite.

$\mu \in \mathbb{R}^n$.

- for our case

$$\mu_{MLE} = \frac{1}{m}\sum_{i=1}^{m}\phi(x_i)$$

$$\Sigma_{MLE} = \frac{1}{m}\sum_{i=1}^{m}(\phi(x_i)-\mu_{MLE})(\phi(x_i)-\mu_{MLE})^T$$

outer product.

} whatever!!

* Baysian multivar Gaussian:-

$$\mathcal{N}(x;\mu,\Sigma) \propto e^{\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]}$$

& $\mathcal{N}(\mu;\mu_0,\Sigma_0) \propto e^{\left[-\frac{1}{2}(\mu-\mu_0)^T\Sigma_0^{-1}(\mu-\mu_0)\right]}$

then $P(\mu|D)$ is also a multivar Gaussian with: $\mu_m, \Sigma_m$.

$$\Sigma_m^{-1} = m\cdot\Sigma^{-1} + \Sigma_0^{-1}$$

$$\Sigma_m^{-1}\mu_m = m\cdot\Sigma^{-1}\hat{\mu}_{mle} + \Sigma_0^{-1}\mu_0$$

multi saisian result.

correspondance of $\Sigma^{-1} \equiv \frac{1}{\sigma^2}$

→ coming back to our regression:

$\forall i\ w_i \sim \mathcal{N}(0, \frac{1}{\lambda})$ — precision.

$\therefore$ for $w$: $\mu_0 = 0$; $\Sigma_0 = \frac{1}{\lambda}I$

& $\hat{\mu}_{mle} = \frac{1}{m}\Sigma\phi(x_i)$

$y_i = w^T\phi(x_i) + \varepsilon$

$\varepsilon \sim \mathcal{N}(0, \sigma^2)$ — error term.

we'll get

for me; not clear how.

$$\Sigma_m^{-1} = \Sigma_0^{-1} + \frac{1}{\sigma^2}\Phi^T\Phi$$

$$\Sigma_m^{-1}\mu_m = \Sigma_0^{-1}\mu_0 + \frac{\Phi^T\cdot y}{\sigma^2}$$

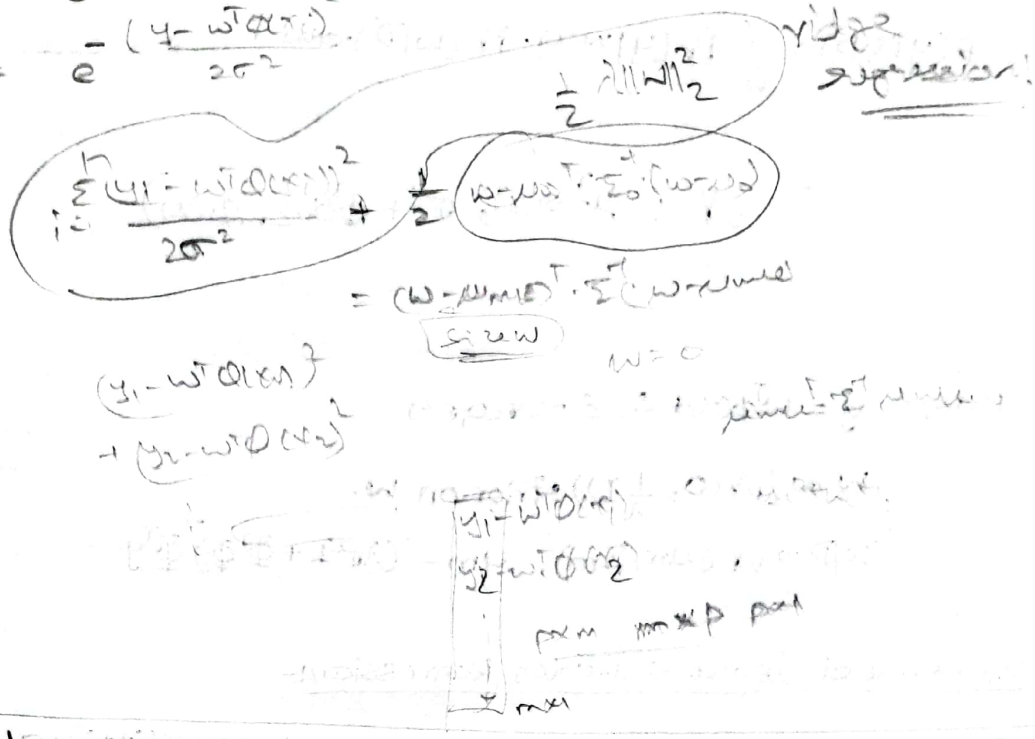
→

$$\Sigma_m^{-1} = \lambda I + \frac{\Phi^T\Phi}{\sigma^2}$$

$$\Sigma_m^{-1}\mu_m = 0 + \frac{\Phi^T y}{\sigma^2}$$

$y_i = w^T \phi(x_i) + \eta$.

$p(y_i|w) = e^{-\frac{(y - w^T\phi(x_i))^2}{2\sigma^2}}$




DT: 23.09.2021

* hence for multivar, bayesian linear regresson;

- $P(w|D)$ is a multivar gaussian $\mathcal{N}(\mu_m, \Sigma_m)$ such that

$$\Sigma_m^{-1} = \lambda I + \frac{1}{\sigma^2}\Phi^T\Phi$$

$$\mu_m = (\lambda\sigma^2 I + \Phi^T\Phi)^{-1}\Phi^T y$$

* Here MAP estimate = mode $[P(w|D)]$ = $(\lambda\sigma^2 I + \Phi^T\Phi)^{-1}\Phi^T y$ (MAP)

Bayesian estimate = mean $[P(w|D)]$ = $(\lambda\sigma^2 I + \Phi^T\Phi)^{-1}\Phi^T y$

Hence

| method | estimate | $P(x\|D)$ |
|---|---|---|
| MLE | $\hat{\theta}_{MLE} = \arg\max_\theta LL(D\|\theta)$ | $P(x\|\theta_{MLE})$ |
| Bayes | $\hat{\theta}_B = E_{P(\theta\|D)}[\theta]$ | $P(x\|\theta_B)$ |
| MAP | $\hat{\theta}_{map} = \arg\max_\theta p(\theta\|D)$ | $P(x\|\theta_{MAP})$ |
| Pure Bayesian | $\theta$ is a distribution, rather than single value. | a pdf for $x$'s image. $P(x\|D) = \int_{-\infty}^{\infty} P(x\|\theta)\cdot P(\theta\|D)\, d\theta$ wow Such a pain.... |

→ Pure Bayesian Regression Summarized:-

* $P(y|x,D) = \int_w \underbrace{Pr(y|w,x)}_{\mathcal{N}(w^T x, \sigma^2)} \cdot \underbrace{Pr(w|D)}_{\mathcal{N}(\mu_m, \Sigma_m)} \cdot dw$ w is a vector here.

$$= \mathcal{N}\left(\mu_m^T \phi(x),\ \sigma^2 + \phi^T(x)\Sigma_m \phi(x)\right)$$

pure bayesian result.
Here mode, mean is $\mu_m^T \phi(x)$

hmmm..... this we do! anyways

Here; $y = w^T\phi(x) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

$w \sim \left(\mathcal{N}(0, \frac{1}{\lambda} I)\right)$ • Prior on w.

$w|D \sim \mathcal{N}(\mu_m, \Sigma_m)$, $\mu_m = (\lambda\sigma^2 I + \Phi^T\Phi)^{-1}\Phi^T y$

---

→ Regularised least squares Regression:-

* Regularization is done to prevent overfitting & keep the model relevant for test samples.
  - In overfitting, w's coefficents become large. Hence include them in loss term.

* we got two forms — penalized, constrained } one form, can be converted to another.

a) Penalized linear reg.:-

* generally, we take

$$w_{reg} = \underset{w}{\operatorname{argmin}} \left[ \|\phi w - y\|_2^2 + \lambda \Omega(w) \right]$$

if $\Omega(w) = \|w\|_2^2$ : Ridge regression } coincides with Bayesian Lin. Reg. Gaussian Prior

if $\Omega(w) = \|w\|_1$ : Lasso-regression } coincides with laplacian prior.

if $\Omega(w) = \|w\|_0$ : Support based penalty

* Some $\Omega(w)$ correspond to priors that are inclosed form.
  Some give good working solutions.
  Some norms are easier to handle.

b) Constrained Linear Regression:-

* generally

$$w_{reg} = \underset{w}{\operatorname{argmin}} \|\Phi w - y\|_2^2 \text{ such that } \Omega(w) \leq \theta.$$

* claim: Any penalized formulation with a 'λ' can be equivalently converted into constrained formulation, with a 'θ'.

Proof requires tools of optimization / duality.

---

* complex models lead to overfitting & has high $\|w\|$ value.

Increasing the penalizing coefficient "λ" makes the model more linear. eigen values of $(\lambda I + \Phi^T\Phi)$ are indicative of curvature of model. more λ means less curvy models.

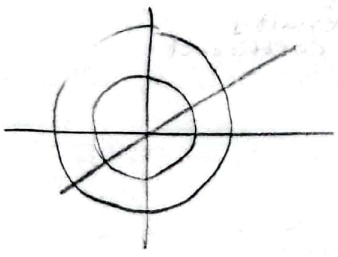
**Ridge**

* $w_{reg} = \underset{w}{\operatorname{argmin}} \|\Phi w - y\|_2^2 + \lambda \|w\|_2^2 = (\lambda I + \Phi^T\Phi)^{-1}\Phi^T y$

closed form!
obtain by differentiating.

*

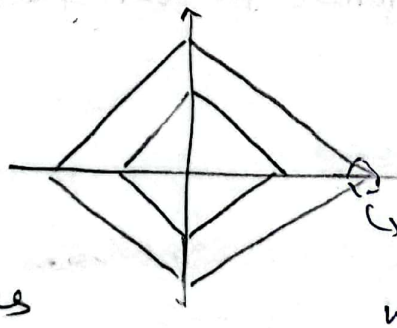
$\|w\|_2^2$ ridge



level curves

we won't get sparse $w_i$ s.

$\|w\|_1^1$ lasso



↳ corners are more prominent. (coefficients of w are 0) we can get sparse $w_i$ s.

Hence, we can find out irrelevant features.! (whose $w_i = = 0$).

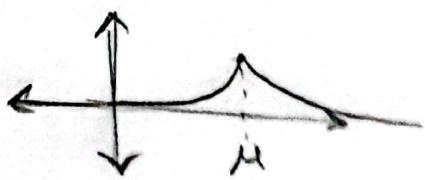
**Lasso**

* $w_{reg} = \underset{w}{\operatorname{argmin}} \|w\Phi - y\|_2^2 + \lambda \|w\|_1^1$ {closed form solution not there}

→ Here laplacian prior on $w_i$'s is used.

$$\text{Laplace}(w|\mu,b) = \frac{1}{2b} \cdot \exp\left(-\frac{|w-\mu|}{b}\right)$$

to get Lasso.



→ Lasso: ISTA:- iterative soft thresholding Algorithm:-

* Gaussian → quick closed form, less sparse

Lasso Reg. → slow iterative, more sparse

* lets see the algorithm:-

$$w_{lasso} = \underset{w}{\operatorname{argmin}} \left[ E_{LS}(w) + \lambda |w|_1 \right]$$

while drop in $E_{lasso}(w^t)$ across $t=k, k+1$ is significant:

1. LS iterate: $w_{LS}^{K+1} = w_{Lasso}^{K} - \eta \nabla E_{LS}(w_{Lasso}^{K})$

2. Proximal Step:

- for every $i \in [p]$

$$[w_{Lasso}^{K+1}]_i = \begin{cases} [w_{LS}^{K+1}]_i - \lambda\eta & \text{if } w_{LS,i}^{K+1} > \lambda\eta \\ [w_{LS}^{K+1}]_i + \lambda\eta & \text{if } [w_{LS}^{K+1}]_i < -\lambda\eta \\ 0 & \text{else} \end{cases}$$ } soft thresholding

* This soft thresholding yeilds greater sparsity as $\lambda \uparrow$.
(penalty coefficient)

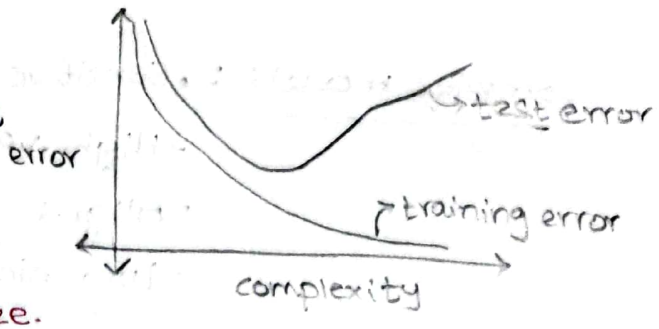
By now; we discussed

- OLS. ordinary least squares.
- MLE interpretation of OLS
- Bayesian linear regression & its closed form
- Ridge linear reg. & its closed form
- Lasso linear regression & its ISTA approach.

# L6: Understanding Bias, Variance:-

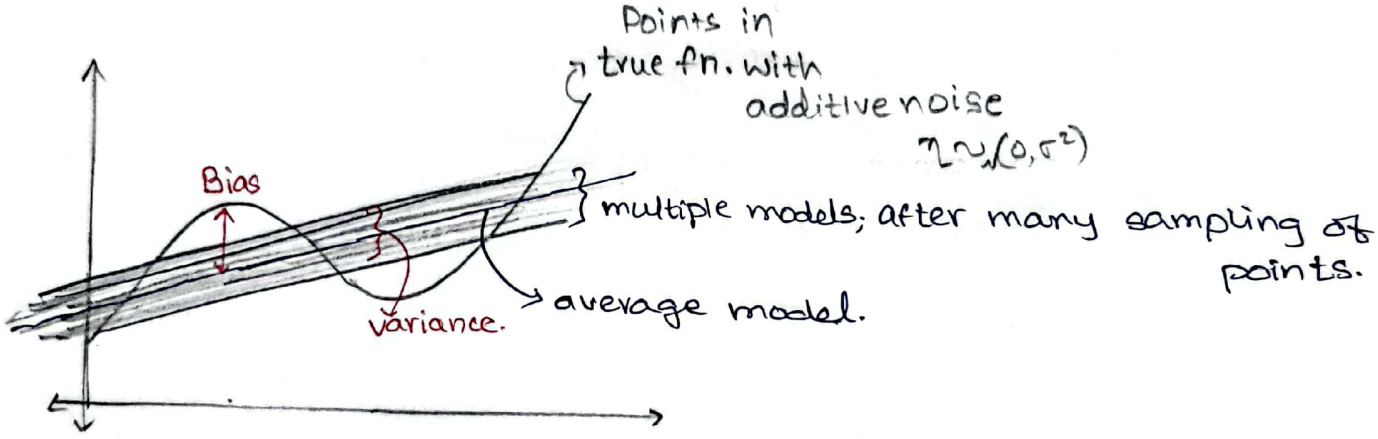
* underfitting too!
* overfitting is a problem!

when model complexity increases, overfitting happens.




* The three sources of error are (→ doesn't depend on sample size.)
  - Bias } b/w avg model & real values
  - Variance } b/w avg model & every model
  - Noise } in real values.

*




* Say true function is $g(x)$. So, $y = g(x) + \epsilon$ in sampling.
  $\epsilon \sim \mathcal{N}(0,\sigma^2)$
  - let $f_D(x) = w^T x$ be our model.
  - given a point $\hat{x}$, we got $\hat{y} = g(\hat{x}) + \epsilon$ in measurement & $f_D(\hat{x})$ via model.

$\therefore Err(\hat{x}) = E_{D,\epsilon}[(\hat{y} - f_D(\hat{x}))^2]$

average over D & noise $\epsilon$.

we manage to show;

$$* Err(\hat{x}) = \underbrace{E[(f_D(\hat{x}) - \hat{f}_D(\hat{x}))^2]}_{\text{variance}} + \underbrace{(g(\hat{x}) - \hat{f}_D(\hat{x}))^2}_{\text{Bias}^2} + \underbrace{\sigma^2}_{\text{Noise}}$$

$E((\hat{y})^2 + (f_D(\hat{x}))^2 - 2\hat{y} \cdot f_D(\hat{x}))$

$= E[(f_D(\hat{x}) - \hat{f}_D(\hat{x}))^2 + (E(f_D(\hat{x})))^2$

$+ E(\epsilon^2) + E((g(\hat{x}))^2)$

$E(-2\hat{g}(\hat{x}) f_D(\hat{x}))$

$- E(2\epsilon f_D(\hat{x}))$

= variance + Bias

↓

proof doable.

* Here we note:

Complex models : • Sensitive to change in data
-High variance
• Aligns with true function
-Low bias

Simple models : • Low variance
• Can't Align. Has straight line Shape
- High bias

*

low $\lambda$ ⟶ high $\lambda$
Complex Simple

⟶
variance decreases
Bias increases.

L7,8 - Linear models for classification. Perceptrons

not for predicting a continuous value!
for classifying!

* we are given data $x_1 x_2 \cdots x_m$.

& possible class choices $C_1 C_2 \cdots C_k$.

we need to model a mapping/classifier

$$f: x \rightarrow \{C_1, C_2, C_3, \cdots C_k\}$$

that predicts class labels for each data.

→ we could try

1) $y = i$ for $C_i$ & apply usual linear reg.

No! we are imposing unnecessary ordering among classes.

2) $y = \begin{cases} [1\,0\,0]^T \text{ for } C_1 \\ [0\,1\,0]^T \text{ for } C_2 \\ [0\,0\,1]^T \text{ for } C_3 \end{cases}$ & apply usual linear regression.

All are equidistant; no ordering imposed here...

NO! calls for more machinary.

* instead of single mapping, we'll try for a series of mappings

$$x \xrightarrow[f(\cdot)]{} y \xrightarrow[g(\cdot)]{} z \longrightarrow \cdots$$

(y: some latent space.)

* this can decrease bias. as complexity increases.

→ Perceptron classifier

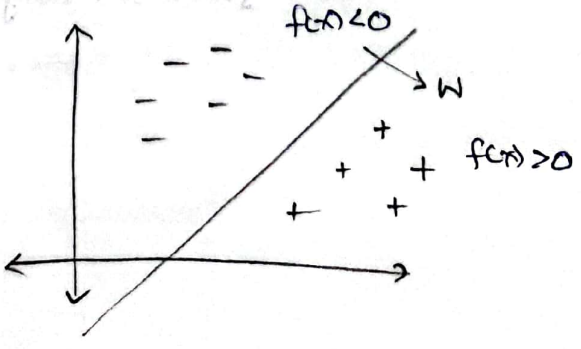
* Lets consider a binary classification problem:

$f(x) \in \{+1, -1\}$ this ordering doesn't mean much.

our $f(x) = w^T \phi(x) + b$

if $f(x) < 0 \rightarrow y = -1$ prediction

if $f(x) > 0 \rightarrow y = +1$ prediction



* How do we find w?

we iteratively update it

$w^{(k+1)} = w^{(k)} + \underline{\quad ? \quad}$ update rule

$= 0$ when point is rightly classified.

$= \eta y \phi(x)$ when otherwise.

* for a rightly classified point $y \cdot w^T \phi(x) > 0$.

unsigned distance.

if $y \cdot w_{(k)}^T \phi(x) < 0$;

then

$w_{(k+1)} = w_{(k)} + \eta y \phi(x)$ update rule of perceptron.

bcoz, now

$y w_{(k+1)}^T \phi(x) = y w_{(k)}^T \phi(x) + \eta y^2 \phi^T(x) \cdot \phi(x)$

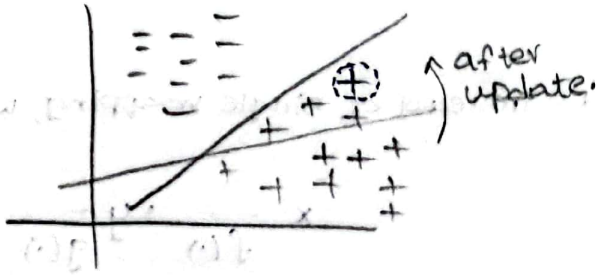
always +ve

∴ we trying to make unsigned distance positive.

for every data point $x_i$

* This perceptron doesnot find the best seperating hyperplane. It finds any.

- SVMs address this problem of providing a breathing space.




* Perceptron can model a linearly seperable function.

* if $\exists w^*$ s.t. points are all satisfying $y w^{*T} x > 0$ then

wlog we can assume

a) $\|w^*\|_2 = 1$ or $\|x\|_2 < 1$ } use a scaling transformation

b) $y x^T w^* > \text{margin}$.

sure! our set is finite...

→ convergence of perceptron:-

* lets say the given data is linearly seperable.

  • Hence $u$ be the unit weight vector.

  • Also take that $\|x\|_2 < 1$ for all datums.

  • let $\gamma = \min_{x \in D} |u^T x|$ is the min. distance of $x \in D$ from '$u$' line.

* lets watch $|w_{(k)}^T u|$ & $\|w_{(k)}\|_2^2$:

a) $w_{(k+1)}^T = w_{(k)}^T + (y(x))^T$

take $\eta = 1$ & $y = 1$

$\|w_{k+1}\|^2 = \|w_k\|^2 + \|x\|^2 + 2w_k^T x$.

-ve; hence we are updating

$\therefore \|w_{k+1}\|^2 - \|w_k\|^2 < 1$

b) $u^T w_{(k+1)} = u^T w_{(k)} + \underbrace{u^T x}_{>0}$

$\therefore u^T w_{k+1} - u^T w_k > \gamma$

* say after $k$ updates, having $w_0 = 0$

$\|w_k\|^2 < k$

$u^T w_k > k\gamma$

& we know that $\|w_k\|^2 \geq |u^T \cdot w_k|^2$

$\therefore k > k^2\gamma^2$

$k < \frac{1}{\gamma^2}$ Hence, we've shown proof & rate of convergence.

if we had $\|x\|_2 < t$

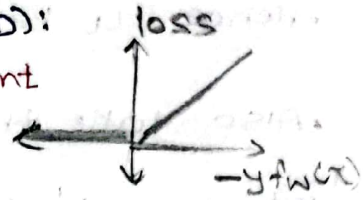
then $k < \frac{t^2}{\gamma^2}$ (midsem question)

$\gamma$ definition still same.

→ What are we minimising? Hinge loss:-

* $\boxed{L_p(f_w(x), y) = \max(0, -y f_w(x))}$ for each $x_i$

lets apply Stochastic gradient descent (SGD): update for each datapoint




$w \leftarrow w - \nabla L_p(f_w(x), y)$

if $y f_w(x) > 0$; $\nabla L_p(\cdot,\cdot) = 0$

if $y f_w(x) < 0$; $\nabla L_p(f_w(x), y) = -yx$.

$\therefore$ $w \leftarrow w + yx$ (same as our update rule)

* Always the final perceptron isn't a good idea. Overfitting

- Voted perceptron:
  votes = no. of samples correctly classified
  wted avg!
- Average perceptron:
  Average different values of w at the end of algorithm.

* Batch stochastic gradient descent:-

$$w \leftarrow w - \sum_{(x,y) \in B} \nabla_w L_p(f_w(x), y)$$

B → Batch

// if we do batch wise, each batch entry should use same old 'w' to calculate $L_p(f_w(x), y)$

if B = complete dataset, we have regular gradient descent.

* SGD → each point $(x_i, y_i)$ should have different updated 'w'.

gradient descent → All points gradient is computed using single, old 'w' value & 'w' is updated.

Next iteration uses this updated 'w'.

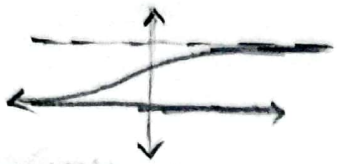
# Lecture 9 - Logistic Regression:-

* till now, $f_w(x) = \text{sign}(w^T\phi(x))$ signum activation.

Now, $f_w(x) = \sigma(w^T\phi(x))$ for activation function $\sigma$

$\sigma$ is sigmoid function ⟶ Logistic regression.

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



* Interpret $\sigma_w(x)$ as $P(y=1|x)$ which our model predicts.

* $L(D,w) = \prod_{i=1}^{m} P(y=y_i | x_i, w)$

**Aim:** find a 'w' such that our model's predictions are most accurate.
OR MLE of w.

$$\therefore \hat{w} = \underset{w}{\text{argmax}} \prod_{i=1}^{m} (\sigma(w^T\phi(x)))^{y_i} (1-\sigma(w^T\phi(x)))^{1-y_i}$$

$$\hat{w} = + \underset{w}{\text{argmin}} -\log(L(D,w))$$

$$\hat{w} = \underset{w}{\text{argmin}} \{ \underbrace{-y_i}_{\text{data to code}} \underbrace{\log(\sigma(w^T\phi(x)))}_{\text{coding length.}} - (1-y_i) \underbrace{\log(1-\sigma(w^T\phi(x)))}_{\text{coding length}} \}$$

we are minimizing coding required for the data.

* This is the cross entropy loss function:-

$$E(w) = \sum -\frac{1}{M}\left[ y_i \log(\sigma_w(x_i)) + (1-y_i)\log(1-\sigma_w(x_i)) \right]$$

↓ After simplification:-

$$y_i \log\left(\frac{e^{wx}}{1+e^{wx}}\right) + (1-y_i)\log\left(\frac{1}{1+e^{wx}}\right)$$

$$\downarrow$$

$$ywx - 1\log(1+e^{wx})$$

$$E(w) = -\left[ \frac{1}{m} \sum_{i=1}^{m} \left( \underbrace{y_i w^T\phi(x_i)}_{\text{unsigned loss of normal perceptron}} - \log(1+e^{w^T\phi(x_i)}) \right) \right]$$

* NO closed form for w. do gradient descent:-

$$\nabla(-\eta E(w)) = \eta \left[ \frac{1}{m} \sum_{i=1}^{m} (y_i - \sigma_w(x_i)) \phi(x_i) \right]$$

we get this, rather than hinge loss, since sigmoid is smooth

$(y_i - \sigma_w(x_i))$ old term

if $y_i = 1$ & $\sigma_w(x_i) \to 1$ then no update! good!

$$\therefore \quad w \leftarrow w + \frac{\eta}{m} \left[ \sum_{i=1}^{m} (y_i \phi(x_i) - \sigma_w(x_i) \cdot \phi(x_i)) \right]$$

meaningful!

Stochastic version:-

$$w \leftarrow w + \eta y_i \phi(x_i) - \eta \sigma_w(x_i) \phi(x_i)$$

Here, sigmoid is smooth. So good expression.

Signum is step function.

→ Regularized LR:-

* $$E(w) = -\frac{1}{m} \sum_{i=1}^{m} \left[ y_i \log(\sigma_w(x_i)) + (1-y_i) \log(1-\sigma_w(x_i)) \right] + \frac{\lambda}{2m} \|w\|_2^2$$

regularized w.

Probabilistic reason:- A bayesian posterior explanation.

Gaussian prior $w_j \sim \mathcal{N}(0, \frac{1}{\lambda})$

→ Multiclass LR:-

* K classes; each class has different weight vector

$w_1, w_2, w_3 \dots w_{K-1}$ (one class don't need)

&

$$P(Y=C \mid \phi(x)) = \frac{e^{-w_C^T \phi(x)}}{1 + e^{-w_1^T \phi(x)} + e^{-w_2^T \phi(x)} + \dots e^{-w_{K-1}^T \phi(x)}} \quad C = 1, 2, \dots K-1$$

&

$$P(Y=K \mid \phi(x)) = \frac{1}{1 + \sum_{C=1}^{K-1} e^{-w_C^T \phi(x)}}$$

# Lecture-10 Kernels:-

* We observe that, we get a lot of $\phi(x_i)^T\phi(x_j)$ everywhere. So, we introduce

$$K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$$

And, by talking in terms of $K(\cdot,\cdot)$ we can use implicitly large $\phi(x)$ spaces.
even infinite.
Kernels are powerful.

→ Kernel perceptron:-

* $f_w(x) = \text{sign}\left(\sum_i \alpha_i y_i K(x, x_i) + b\right)$

Never explicitly compute $\phi(x)$ now. Its very costly

- Initialize : $\alpha_i = 0 \quad \forall i \in [m]$
- update :

  if $f(x) \neq y_i$
  then
  $\alpha_i \leftarrow \alpha_i + 1$ ~~~ this is the normal $w \leftarrow w + y_i \phi(x)$
  But kernalised.

* Linear Kernel : $K(x,y) = x^Ty$ Exponential: $K(x,y) = e^{x^Ty}$ (Kinda polynomial)

Polynomial : $K(x,y) = (1+x^Ty)^d$

RBF Kernel : $K(x,y) = e^{-\frac{1}{2}\left(\frac{\|x-y\|^2}{\sigma^2}\right)}$ } Implicit $\phi(x)$ is $\infty$.

$K_y(x)$ is Smoothing Kernel.

Eg: What is the implicit $\phi(x)$:-

Say $K(x,y) = (1+x^Ty)^2$ & $x, y \in \mathbb{R}^2$

We can get non-linear classifiers !!!

$$= (1 + x_1y_1 + x_2y_2)^2$$

$$= 1 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1y_1 + 2x_2y_2 + 2x_1x_2y_1y_2.$$

Now; $\phi(x)^T\phi(y) = \nearrow$

$$\therefore \phi(x) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1x_2 \end{bmatrix}$$

So simple $K(x,y)$ 2D Space
But so huge $\phi(x)$ !! 6D Space
So $K(x,y)$ has obvious advantage.

even in CS215??

→ The Gram matrix:- Kernel matrix

$$K = \begin{bmatrix} K(x_1,x_1) & \cdots & K(x_1,x_m) \\ \vdots & & \vdots \\ K(x_m,x_1) & \cdots & K(x_m,x_m) \end{bmatrix}$$

if $K(x_i,x_j)$ is valid and $= \phi(x_i)^T \cdot \phi(x_j)$

then

K is Symmetric matrix

pos. semidefinite matrix.

$b^T K b \geq 0$ we can show...

Hence eigenvalues $\geq 0$. ∴.

* Since K is positive semi definite, we can write (SVD?)

$$K = U\Sigma U^T = (U\Sigma^{1/2})(U\Sigma^{1/2})^T$$
$$= RR^T$$

- here rows of U are linearly independent. U is an orthogonal matrix.
- $\Sigma$ is a positive diagonal matrix.

This rows of R can be seen as feature vectors.

$\phi(x_1)$
$\phi(x_2)$
⋮
$\phi(x_m)$

→ Mercer's theorem:-

def^n: $K(x_1,x_2)$ is a mercer kernel, if

$$\int_{x_1}\int_{x_2} K(x_1,x_2)\,g(x_1)\,g(x_2)\,dx_1 dx_2 \geq 0$$ for all square integrable functions $g(x)$.

i.e. $\int_x (g(x))^2 dx < \infty$.

thm: "For any mercer kernel $K(x_1,x_2)$,

$$\exists\, \phi(x) : \mathbb{R}^n \longmapsto H, \text{ s.t. } K(x_1,x_2) = \phi(x_1)^T \phi(x_2)"$$

H: hilbert space

- H is hilbert space, infinite dimension version of eculedian space.
- Formally, H is a innerproduct space with associated norm, where every cauchy sequence is convergent.

* prove $x_1^T x_2$ is a mercer kernel :-

$$\int\int_{x_1 x_2} x_1^T x_2 \, g(x_1) \cdot g(x_2) \, dx_1 dx_2$$

$$= \left(\int_{x_1} x_1 g(x_1)\right)^T \left(\int_{x_2} x_2 g(x_2)\right)$$

relabel $x_2$ as $x_1$ :

$$= \left(\int_{x_1} x_1 g(x_1) dx_1\right)^T \cdot \left(\int_{x_1} x_1 g(x_1) dx_1\right)$$

$$\geq 0$$

$$\int_{x_1}\int_{x_2} (x_{1a}x_{2a} + x_{1b}x_{2b}) g(x_1) g(x_2)$$

$$= \sum \int\int_{x_1 x_2} x_{1a} g(x_1) \cdot x_{2a} g(x_2)$$

$$= \sum \left(\int_{x_1} x_{1a} g(x_1) dx_1\right) \left(\int_{x_2} x_{1a} g(x_2) dx_2\right)$$

$$= \sum \left(\int_{x_1} x_{1a} g(x_1) dx_1\right)^2$$

$$\geq 0 \checkmark$$

→ Kernels closure properties:-

1) Addition:-

* sum of two kernels is also kernel

* $\phi_{new}$ is concat $(\phi_1, \phi_2)$. ← this is proof.

2) Product:-

* product of two kernels also kernel.

* $\phi_{new}$ is cross-multiplication of $\phi_1, \phi_2$ ....

Good!

# Lecture-11: Various Kernelisations

→ Kernalized LR:-

* cross entropy loss function was :-

$$E(w) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y_i w^T\phi(x_i) - \log(1+e^{w^T\phi(x_i)})\right)\right] + \frac{\lambda}{2m}\|w\|_2^2$$

Now, kernalized version is:-

$$E_D(\alpha) = -\left[\sum_{i=1}^{m}\left(\sum_{j=1}^{m} y_i K(x_i,x_j)\alpha_j - \frac{\lambda}{2}\alpha_i\alpha_j K(x_i,x_j)\right) - \log\left(1+\exp\left[-\sum_{j=1}^{m}\alpha_j K(x_i,x_j)\right]\right)\right]$$

easy.... //

& decision function:

$$\sigma_w(x) = \frac{1}{1+\exp\left[-\sum_{j=1}^{m}\alpha_j K(x_i,x_j)\right]}$$ // fine ✓✓

→ Kernalized ridge regression:-

↳ wow!

* Recall ridge regression:-

The bayes and map estimate for linear regression $y = w^T\phi(x) + \epsilon$ with gaussian prior $w \sim N(0, \frac{1}{\lambda}I)$ coincide with ridge regression $\epsilon \sim N(0,\sigma^2)$

$$w_{ridge} = \arg\min_w \left(\|\Phi w - y\|_2^2 + \lambda\sigma^2\|w\|_2^2\right)$$

with closed form

$$w_{ridge} = (\Phi^T\Phi + \lambda\sigma^2 I)^{-1}\Phi^T y$$

So, now the function f(x) will be...

$$f(x) = w^T\phi(x) = \phi(x)^T w = \phi(x)^T(\Phi^T\Phi + \lambda I)^{-1}\Phi^T y$$

Now.... look for kernalization...

$\Phi^T\Phi$ is NOT K.

$\Phi\Phi^T$ is kernal matrix K.

Need to use matrix identity...

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = P B^T (B P B^T + R)^{-1}$$ hehe... oops

Hence, now $R = I$, $P = \frac{I}{\lambda}$, $B = \phi$

$$(\lambda I + \Phi^T \Phi)^{-1} \Phi^T = \frac{1}{\lambda} \Phi^T \left( \frac{\Phi \Phi^T}{\lambda} + I \right)^{-1}$$

$\therefore$ $w = \Phi^T (\Phi \Phi^T + \lambda I)^{-1} y = \sum_{i=1}^{m} \alpha_i \phi(x_i)$, $\alpha_i = ((\underbrace{\Phi \Phi^T}_{K} + \lambda I)^{-1} y)_i$

ridge reg.

So; $f_w(x) = \sum_{i=1}^{m} \alpha_i \underline{\phi^T(x) \cdot \phi(x_i)}$

Kernel!

$\therefore$ $\boxed{f_w(x) = \sum_{i=1}^{m} \alpha_i K(x, x_i)}$ where $\alpha_i = ((K + \lambda I)^{-1} y)_i$

Hence; given $(x_i, y_i)$ we can directly use kernels and calculate a new $f(x)$

there's no 'w' here now.
only $\alpha_i$'s.

complicated ones... with infinitely large implicit $\phi(x)$ !

# Lecture-12: Unsupervised learning

## PCA and dimensionality reduction

principle component Analysis

* dimensionality reduction is a feature extraction technique where we want to represent input data with fewer dimensions.

Occams Razor: Make everything simple.

* consider $x \in \mathbb{R}^d$.

project it to a k-dim vector z → here, the new axes are eigen vectors of old space.

Awesome!

$z = U^T x$ where $U_{d \times k}$ is projection matrix.

& $x = Uz$

we want

$U^T U = I$. } like, columns are orthonormal eigen vectors.

$UU^T$ can be anything.

- whats are PCA's objectives?

Perspective-1:- Minimized sum of squared distances of $\tilde{x}$ from x.

$\tilde{x}$ ↓ constructed.

$\tilde{x} = Uz$

$x$ is given

$z = U^T x$.

∴ we need $U^* = \underset{U}{\text{argmin}} \sum_x \|x - UU^T x\|^2$ such that $U^T U = I$

Perspective-2:- Maximize variance of projected data.

Assume $E(X) = 0$. (since variance won't change)

$z = U^T x$

$E[\|z\|^2] = \frac{1}{n} \sum_{i=1}^{n} \|U^T x_i\|_2^2$

∴ we need $U^* = \underset{U, U^T U = I}{\text{arg max}} \frac{1}{n} \sum_{i=1}^{n} \|U^T x_i\|^2$

Both are equivalent — oh lol!

since $x = \underbrace{UU^T x}_{A} + \underbrace{(I - UU^T)x}_{B}$ $\quad AB = 0$

$\therefore \|x\|^2 = \underbrace{\|UU^T x\|^2}_{E(\|U^T x\|^2)} + \|x - UU^T x\|^2$ } reconstruction error. to minimize

to maximize.

* we are going to work with "maximizing variance." lets do one component

$$\max_{\|u\|=1} E[(u^T x)^2]$$

$$= \max_{\|u\|=1} \frac{1}{n} \|u^T X\|^2 \qquad X = \begin{bmatrix} \vdots & \vdots & \\ x_1 & x_2 & \cdots \\ \vdots & \vdots & \end{bmatrix}$$

$$= \max_{\|u\|=1} \frac{1}{n} u^T (xx^T) u$$

$$= \max_{\|u\|=1} u^T S u$$

this means $u$ is an eigenvector.

$S = \frac{1}{n}(XX^T)$ covariance matrix. $\because E(x) = 0$.

$u \equiv$ eigen vector with largest eigen value.

$\therefore$ the $d \times k$ projection matrix $U = \begin{bmatrix} \vdots & \vdots & \vdots & \\ e_1 & e_2 & e_3 & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}_{d \times k}$

eigenvalues in decreasing order.

PCA Algorithm !

---

→ Kernel PCA Algorithm:-

* consider SVD of $X$. $X = ADB^T$ ($n \times n$, $n \times k$, $k \times k$)

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x_2 & \cdots & x_m \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

so $nS = XX^T = AD^2A^T$

$X^T X = BD^2B^T$

$U = AD$ is the matrix of all principle components

* Now, after taking the kernel matrix, $K = X^T X = BD^2B^T$.

the projections $XU$ of data $X$ onto components $U$ can be computed from eigen decomposition of $K$.

Pending

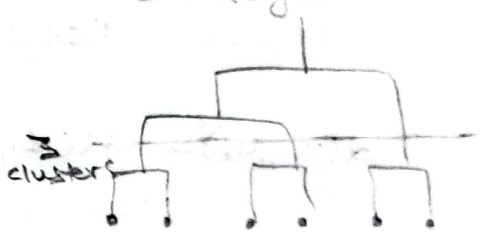
L12.

# Lecture-13: clustering

## Unsupervised learning.

* Two paradigms — Bottom-up agglomerative clustering
  — Top-down partitioning "move to nearest"

dendogram




$O(n^2 \log n)$ with $n^2$ space.

→ K-means clustering:-

Important step!

1) Initialize $\mu_i^{(0)}$ to random values.

2) $P_r^{(t+1)} \in \underset{P}{\text{argmin}} \sum_{j=1}^{m} \sum_{l=1}^{k} P_{l,j} \| \phi(x_j) - \mu_l^{(t)} \|^2$

**solution:**

let $i$ be $\underset{l}{\text{argmin}} \| \phi(x_j) - \mu_l^{(t)} \|$

then $P_{i,j}^{(t+1)} = 1$ and

$P_{l,j}^{(t+1)} = 0 \quad \forall l \neq i$

3) $\mu^{(t+1)} = \underset{\mu}{\text{argmin}} \sum_{j=1}^{m} \sum_{l=1}^{k} P_{l,j}^{(t+1)} \| \phi(x_j) - \mu_l \|^2$

**solution:**

$$\mu_l^{(t+1)} = \frac{\sum_{j=1}^{m} P_{l,j}^{(t+1)} \cdot \phi(x_j)}{\sum_{j=1}^{m} P_{l,j}^{(t+1)}}$$

4) if any $P_{l,j}$ is changed, go to step 2.

* K-means will converge, since we have finite labelling for points & we always move to a less error value.

→ Kernel Kmeans:-

* Basic idea; replace $\mu_l$ by $\phi(\mu_l)$, which is calculated by $P_{l,j}$.

* we expand $d(x_i, \mu_l)^2 = \|\phi(x_i) - \mu_l\|^2$ as

$$d(x_i, \mu_l)^2 = \|\phi(x_i)\|^2 + \|\mu_l\|^2 - 2\|\phi(x_i) \cdot \mu_l\|$$

$$d(x_i, \mu_l)^2 = K(x_i, x_i) + K(\mu_l, \mu_l) - 2K(x_i, \mu_l)$$

and now step-3 won't be there.

in step-2; we calculate

1) $K(x_i, \mu_l)$ using

$$\frac{\sum P_{l,j} K(x_i, x_j)}{\sum P_{l,j}}$$

thanks Mukesh!

2) $K(\mu_l, \mu_l)$ using

$$\frac{\sum P_{l,j_1} \cdot P_{l,j_2} K(x_{j_1}, x_{j_2})}{\sum P_{l,j_1} \cdot P_{l,j_2}}$$

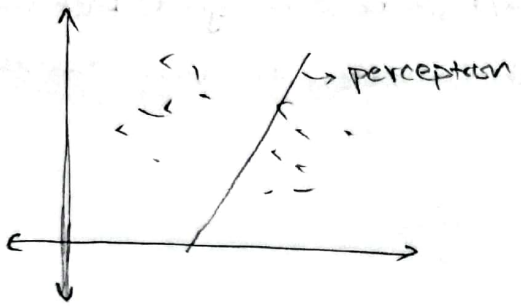
* Note that $\phi$ doesn't have to be computed/stored for each data instance $x_i$ or for the cluster means $\mu_l$.

* remember that $\phi(\mu_l)$ is mean of data's in feature space, which might not have a pre-image in the data space.

# Lecture-14 : SVMs

support vector machines.

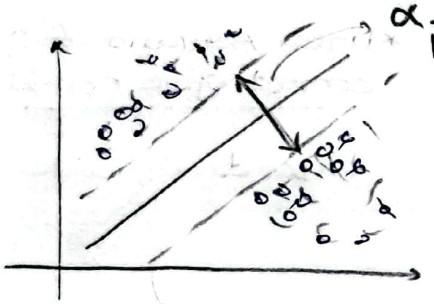
* The perceptron discussed previously doesn't find the Best hyperplane. to address this, we use SVMs.





→ support vector classification:-

1) seperable case:-



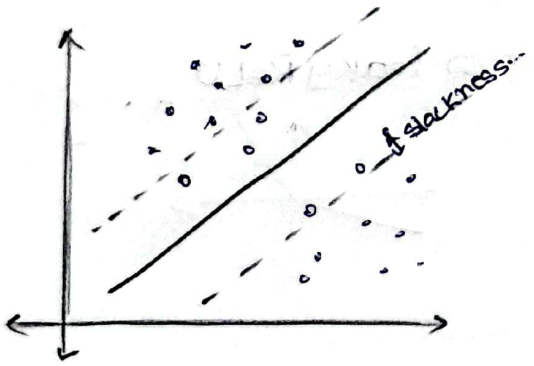
$\propto \frac{1}{\|w\|}$

$y(w^T\phi(x)+b) \geq +1$

margin!

$\propto \frac{1}{\|w\|}$

2) Non-seperable case:-




when examples are not seperable, we need to consider slackness

$\xi_i \geq 0$

$y(w^T\phi(x)+b) \geq 1-\xi_i$

Objective:-

* 1) maximize the margin $= \frac{2}{\|w\|_2}$

2) Minimize the slackness $= \sum \xi_i$

this min, will maximize margin val.

∴ final objective: $(w^*, b^*) = \underset{w,b,\xi_i}{\operatorname{argmin}} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i$

such that $y_i(\phi(x_i)^T w + b) \geq 1-\xi_i$ and $\xi_i \geq 0$

✓ good!

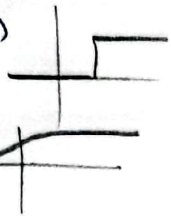
# Lecture 15 – Intro to neural networks:-

* **Neural networks:- universal approximation properties**

  • with a single hidden layer of sufficient size and reasonable choice of non-linearity: sigmoid/hyperbolic tangent/ RBF unit; one can represent any smooth function to any desired accuracy.

• Activation functions

1) step : $g(s) = 1$ if $s \in [0, \infty)$
$= 0$ else



2) sigmoid: $g(s) = \frac{1}{1+e^{-s}}$



rectified linear unit

3) ReLU: (most popular) $g(s) = \max(0, s)$

4) ~~Softmax~~: $g(s) = \ln(1+e^s)$ Softplus

diff. for ReLU *

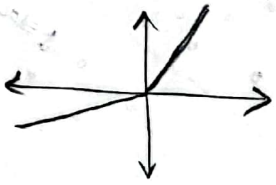
5) tanh: $g(s) = \frac{2}{1+e^{-2x}} - 1$

*neural networks is cascade of perceptrons giving non-linearity.

*Linear Activation fun", cannot give non-linearity

cascade of linear is linear.

6) leaky ReLU $f_w(s)$



* **Softmax**

• multi class logistic reg. with K weight vectors

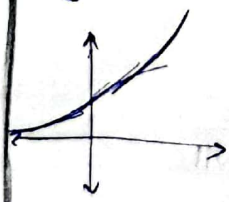
• $P(y=c|x) = \frac{e^{-w_c^T \phi(x)}}{\sum_{i=1}^{K} e^{-w_i^T \phi(x)}}$

• one vector is redundant

**Softplus**

Activation function...

$g(s) = \ln(1+e^s)$



smooth version to ReLU

→ VC-dimension:- Vapnik-Chervonenkis

* measure for seperability of a classifier.

* = cardinality of the largest set of points that can be shattered.

a function $f_w$ is said to shatter a set of points $(x_1, x_2, \ldots x_n)$ if, for all $2^n$ assignments of labels to those points, there exists a 'w' such that $f_w$ makes no errors.

* Hence; if there exist atleast one set of size 'n', then $V.C._{dim} \geq n$.

Examples:

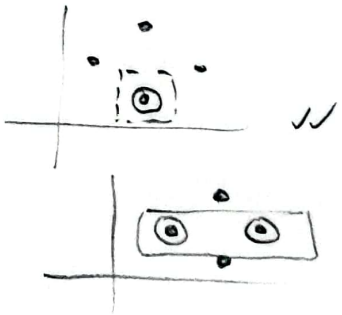
1) For $f_w$ as threshold classifier, VC = 1 in $\mathbb{R}$

2) for $f_w$ as interval classifier, VC = 2 in $\mathbb{R}$

3) For $f_w$ as perceptron in $\mathbb{R}^2$, VC = 3 (Linear)

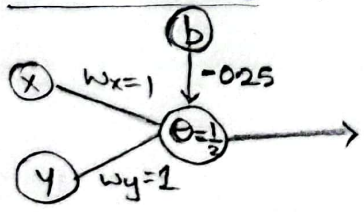
linear perceptron in $\mathbb{R}^n$, VC = n+1

VC = 4 for parallel rect. classifier

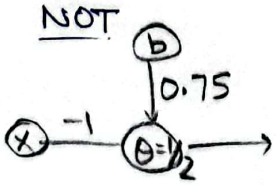


---

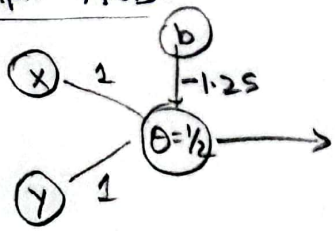
Simple OR using step




NOT




Simple AND

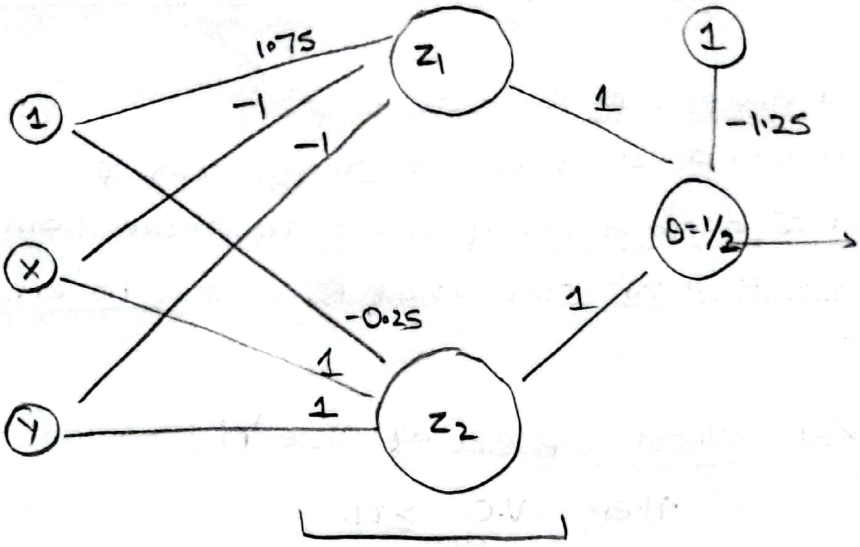



Now; whole logic gates are ready.

Any logic circuit can be built.

Now; we build XOR as

$(x \oplus y) = (x \vee y) \wedge (\bar{x} \vee \bar{y})$

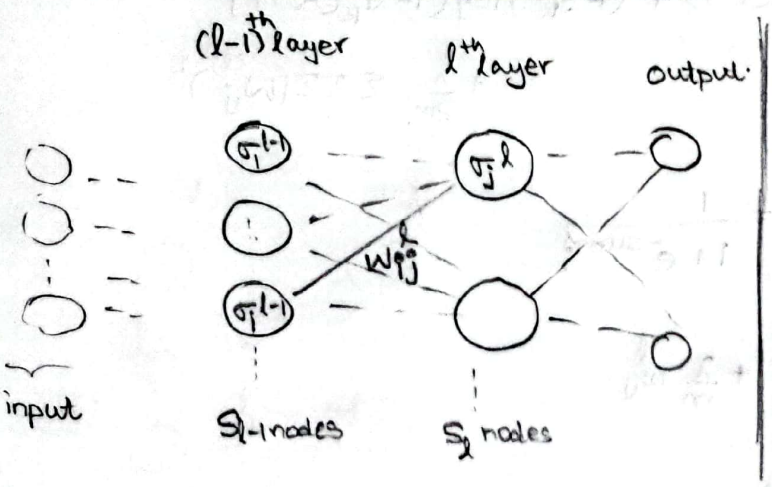


# Lecture 16 - Training NNs, Backpropagation Algo

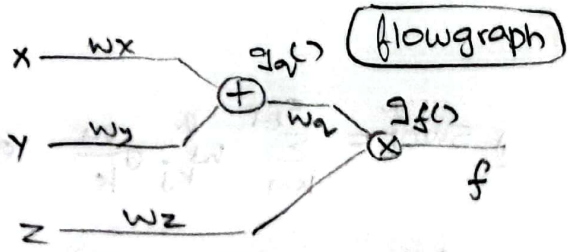
* picking a network architecture:
  - No. of input units : Dimension of features $\phi(x_i)$
  - No. of output units : No. of classes
  - Reasonable default: 1 hidden layer or
    >1 hidden layer having same no. of nodes
  - No. of hidden units in each hidden layer is constant factor ($\times 3$ or $\times 4$) of input units.

* we want to apply gradient descent to get optimal weights. (on each edge).

→ Backpropagation for training NNs:-




$f(x,y,z) = g_f(w_v g_v(w_x x + w_y y) + w_z z)$




now, $\frac{df}{dw_x} = \frac{df}{dg_f} \cdot \frac{dg_f}{dg_v} \cdot \frac{dg_v}{dw_x}$

back prop....

1) Randomly initialize weights $w_{ij}^{l}$ for $l = 1,2,\dots L$; $i = 1,2,\dots S_{l-1}$; $j = 1,2,\dots S_l$

2) Implement forward propagation to get $f_w(x)$ for any $x \in D$

3) Execute backward propagation:

   a) by computing partial derivatives

   $\frac{\partial E(w)}{\partial w_{ij}^{l}}$ using chain rule

level curves are:-

   b) and use gradient descent to try to minimize (non-convex) $E(w)$

$$w_{ij}^{l} = w_{ij}^{l} - \eta \frac{\partial}{\partial w_{ij}^{l}} E(w)$$

4) verify $E(w)$ has indeed reduced, else resort to random perturbation of w's to address non-convexity of $E(w)$.

memoize the previous partial derivatives!

* The objective to be minimised is

$$E(W) = \underbrace{\frac{-1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\sigma_k^{L}(x^{(i)})\right) + (1-y_k^{(i)})\log\left(1-\sigma_k^{L}(x^{(i)})\right)}_{\text{wrote for every } K \text{ classes.}} + \frac{\lambda}{2m}\sum_{l=1}^{L}\sum_{i=1}^{S_{l+1}}\sum_{j=1}^{S_l}\left(w_{ij}^{(l)}\right)^2$$

{ Cross entropy wrt output layer. true!

* chain rule & memoisation is bcoz of this part

all weights in hidden layers too regularised

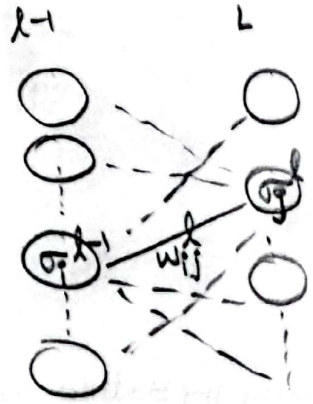
* lets consider sigmoidal activation $g(s) = \frac{1}{1+e^{-s}}$.

$$\frac{dg(s)}{ds} = g(s)(1-g(s))$$

Now, for single point...

$$E(W) = -\sum_{k=1}^{K} y_k^{(i)} \log\left(\sigma_k^{L}(x^{(i)})\right) + (1-y_k^{(i)})\log\left(1-\sigma_k^{L}(x^{(i)})\right) + \frac{\lambda}{2m}\Sigma\Sigma\Sigma\left(w_{ij}^{(l)}\right)^2$$

1) $sum_j^l = \sum_{k=1}^{S_{l-1}} w_{kj}^{l} \cdot \sigma_k^{l-1}$ & $\sigma_j^l = \frac{1}{1+e^{-sum_j^l}}$




2) $\frac{\partial E}{\partial w_{ij}^{(l)}} = \frac{\partial E}{\partial \sigma_j^l} \cdot \underbrace{\frac{\partial \sigma_j^l}{\partial sum_j^l}}_{\sigma_j^l(1-\sigma_j^l)} \cdot \underbrace{\frac{\partial sum_j^l}{\partial w_{ij}^{(l)}}}_{\sigma_i^{l-1}} + \frac{\lambda}{m} w_{ij}^{l}$

our gradient descent value

$\frac{\partial E}{\partial \sigma_j^l}$ = ?

3) ★ $\frac{\partial E}{\partial \sigma_j^l} = \sum_{p=1}^{S_{l+1}} \frac{\partial E}{\partial \sigma_p^{l+1}} \cdot \frac{\partial \sigma_p^{l+1}}{\partial sum_p^{l+1}} \cdot w_{jp}^{l+1}$

use memoization here!

$\frac{\partial sum_p^{l+1}}{\partial \sigma_j^l}$

$\sigma_p^{l+1}(1-\sigma_p^{l+1})$

at outer layer

$$\frac{\partial E}{\partial \sigma_j^L} = -\frac{y_j}{\sigma_j^L} - \frac{1-y_j}{1-\sigma_j^L}$$