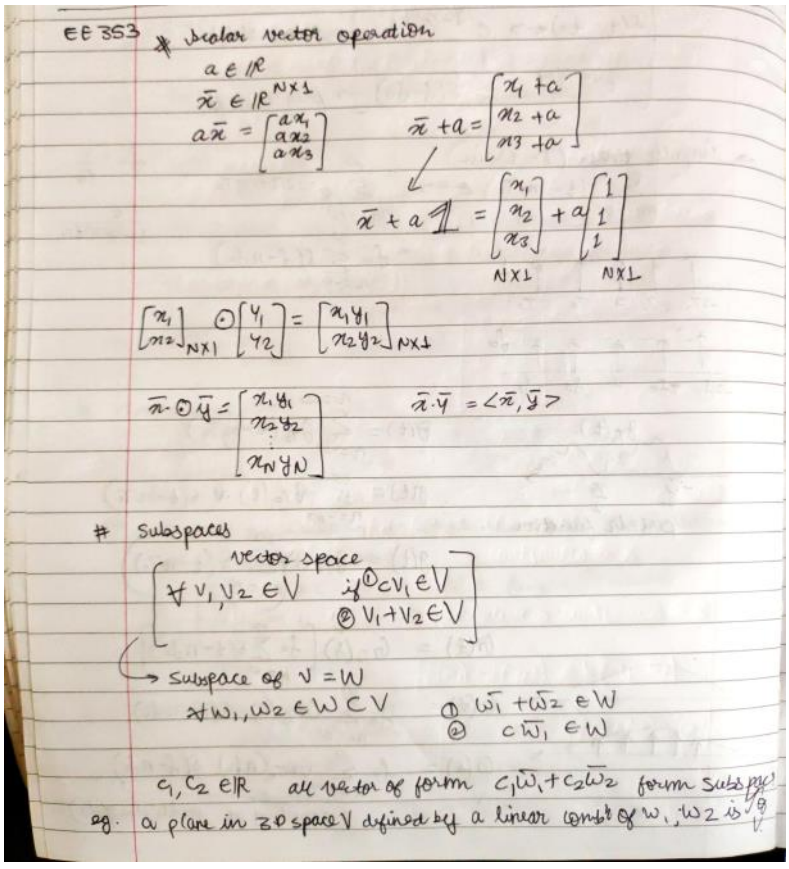
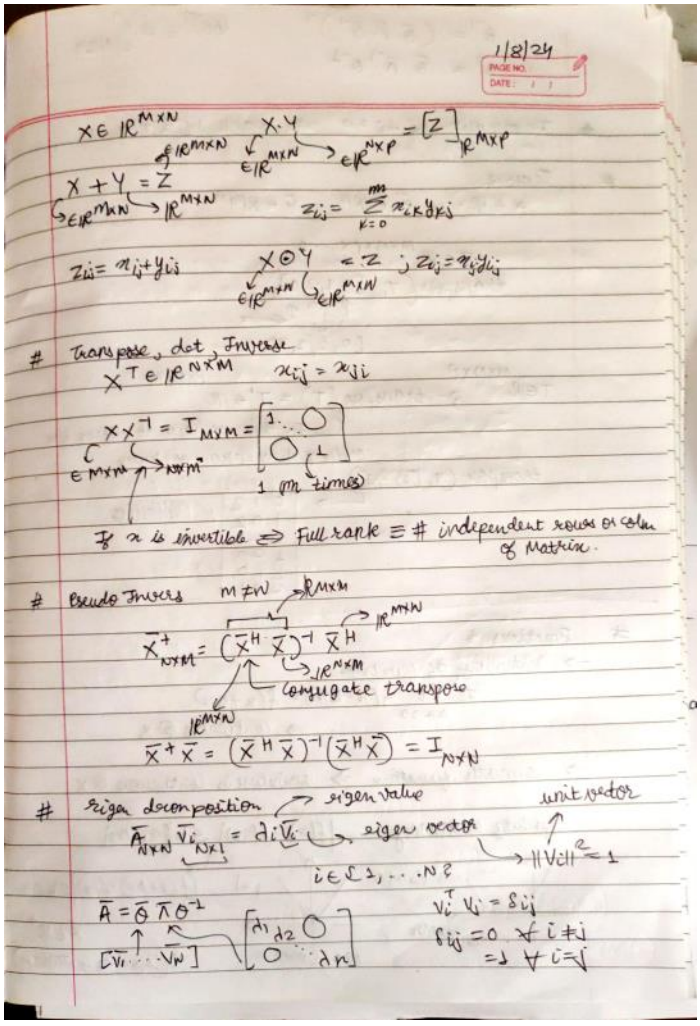
EE353_AIDS

01 August 2024 23:37




1/8/24
PAGE NO.
DATE:
X ∈ ℝ^{M×N}
X + Y = Z
∈ℝ^{M×N} → ℝ^{M×N}
X · Y = Z
∈ℝ^{M×N} ∈ℝ^{N×P} → ℝ^{M×P}
z_ij = Σ_{k=0}^{m} x_ik y_kj
z_ij = x_ij + y_ij
X ⊙ Y = Z ; z_ij = x_ij y_ij
∈ℝ^{M×N} ∈ℝ^{M×N}
Transpose, dot, Inverse
X^T ∈ ℝ^{N×M}
x_ij = x_ji
X X^{-1} = I_{M×M}
∈ M×M → N×M
1 (m times)
If x is invertible ⇒ Full rank ≡ # independent rows or cols of Matrix.
Pseudo Inverse
M ≠ N
ℝ^{M×M}
X^+_{N×M} = (X̄^H X̄)^{-1} X̄^H
ℝ^{N×M}
ℝ^{M×N}
conjugate transpose
X̄^+ X̄ = (X̄^H X̄)^{-1}(X̄^H X̄) = I_{N×N}
eigen decomposition
eigen value
unit vector
A_{N×N} v_{i N×1} = λ_i v_i
eigen vector
||v_i||^2 = 1
i ∈ {1, ... N}
A = Θ Λ Θ^{-1}
[v_1 ... v_N]
λ_1 λ_2 0 ... 0 λ_N
v_i^T v_j = δ_ij
δ_ij = 0 ∀ i ≠ j
= 1 ∀ i = j

$A^{-1} = (\bar{\theta}\,\bar{\Lambda}\,\bar{\theta}^{-1})^{-1}$ $\quad (AB)^{-1} = B^{-1}A^{-1}$

$A^{-1} = \bar{\theta}\,\bar{\Lambda}^{-1}\,\theta^{-1}$

1/8/24

PAGE NO.:
DATE: / /

* Implication of $\lambda_i = 0 \Rightarrow$ rank deficiency

\# Tensors

$\alpha \in \mathbb{R}$; $\bar{x} \in \mathbb{R}^N$; $\bar{Y} \in \mathbb{R}^{M \times N}$

$T \in \mathbb{R}^{M \times N \times P \times \cdots \theta}$

transpose(T, [3,1,2,0])

[0, 2, 1, 3]

$T \in \mathbb{R}^{M \times N \times P}$; transpose(T) = $T' \in \mathbb{R}^{P \times N \times M}$

In this we have to define the order of dimension of array

transpose(T, [0, 2, 1])

0	1	2
1	0	2
0	2	1
1	2	0
2	1	0
2	0	1

← possibilities

\# Functions &

→ continuity of function:

If $\lim_{\Delta x \to 0} f(x - \Delta x) = f(x + \Delta x)$

$\Rightarrow$ continuous @ X

→ smooth function $\Rightarrow$ derivative is continuous @ X

Lipschitz Continuity: $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$

$|f(x + \Delta x) - f(x)| \leq K \Delta x$

$K \in \mathbb{R}^+$

$ax^2 + bx + c$ — smooth

$|x|$ — Not smooth

$\Rightarrow$ (Lipschitz Continuity)

1/8/24

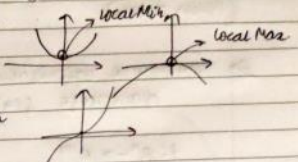
→ derivatives of function

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \left(\frac{f(x+\Delta x) - f(x)}{\Delta x} \right)$$

→ critical point of a function

$f'(x) = 0$

- Maxima
- Minima
- Inflection



$f'(x) = 0 \rightsquigarrow f''(x) > 0 \Rightarrow$ Minima

$f''(x) < 0 \Rightarrow$ Maxima

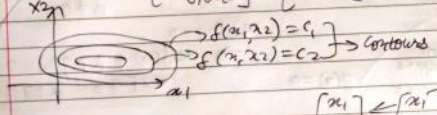
$f''(x) = 0 \Rightarrow$ Inflection

Multivariate function

$y = f(x_1, x_2)$

eg. $y = ax_1^2 + bx_2^2$

$$\nabla y = \begin{bmatrix} \partial y/\partial x_1 \\ \partial y/\partial x_2 \end{bmatrix} = \begin{bmatrix} 2ax_1 \\ 2ax_2 \end{bmatrix}$$



$$\nabla f = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \lambda \Delta y$$

gradient descent in 2D

@ critical $\nabla f = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

eg. $\nabla y = \begin{bmatrix} 2ax_1 \\ 2bx_2 \end{bmatrix} = 0$ for @ critical pt.

if $a > 0$; $b < 0$

$(0,0)$ is max for x_2

min for x_1

→ saddle point

2/8/2024

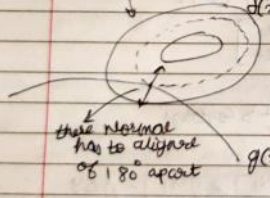
$$Hf = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 \\ \partial^2 f/\partial x_2 \partial x_1 & \partial^2 f/\partial x_2^2 \end{bmatrix} = \begin{bmatrix} 2a & 0 \\ 0 & 2b \end{bmatrix}$$

(Hessian Matrix)

constrained optimization using Lagrange multiplier

maximize $f(x)$; subject to $g(x) = 0$

$f: \mathbb{R}^n \rightarrow \mathbb{R}^m$	$f: \mathbb{R}^n \rightarrow \mathbb{R}$
$x \in \mathbb{R}^n \quad f(x) \in \mathbb{R}^m$	$x \in \mathbb{R}^n \quad f(x) \in \mathbb{R}$
$\left[J_{ij} = \frac{\partial f_i}{\partial x_j} \right]$	$(H_f)_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$
Jacobian Matrix	$H_f(f(x)) = J(\Delta f(x))^T$



$L(x) = f(x) + \lambda g(x)$

$\lambda \neq 0$

$\nabla L(x) = 0$

$\Rightarrow \nabla f(x) + \lambda \Delta g(x) = 0$

$\nabla f(x) = -\lambda \Delta g(x)$

sum + m/n − m = min * n + m%

sum − min * n = m

EE353 ⇒ Eigen vector are orthonormal for symmetric Matrix

Probability

≠ Random Variable

$P(X = x)$

≠ Discrete Probability ; pmf [Probability mass function]

≠ Cont^n Probability ; pdf [Probability density fn^n]

→ ① Bernoulli dist^n

$x \in \{0, 1\}$ $\quad E(x) = \sum_x x \cdot P(X=x) = \mu$

$\mu = P(X = 1)$

$E((x-\bar{x})^2) = \mu(1-\mu)$

$Bern(x|\mu) = \mu^x (1-\mu)^{1-x} = \begin{cases} \mu & ; x=1 \\ 1-\mu & ; x=0 \end{cases}$

② Binomial dist^n

↳ N times Binary

[Expected value / Entropy value]

$E(y) = N\mu$

$y \in \{0, 1, \ldots N\}$ (more spaces)

$Bin(y|\mu, N) = \binom{N}{y} \mu^y (1-\mu)^{N-y}$

↳ (out of N, y occurs)

$\delta(y)$

5/8/24

$E(\delta(y)) = \sum_{y=0}^{N} Bin(y|\mu, N) y$

$E(f(y)) = \sum_y P(y) \cdot f(y)$

$Entropy(y) = E[-\log P(y)]$

⇒ for Bernoulli dist^n. $Entropy[y] = -\mu \log \mu - (1-\mu)\log(1-\mu)$

$\frac{d}{d\mu} = -\log \mu$

$-\log \mu - 1 + \log(1-\mu) - 1 = 0$

$\log\left(\frac{1-\mu}{\mu}\right) = 2$

$\frac{1-\mu}{\mu} = e^2 \quad \left(\mu = \frac{1}{1+e^2}\right)$

Continuous Rand var

$P(x = x) = 0$ (PDF)

$P(a < x \le b) = \int_a^b p(x)\,dx$ $\quad P(x \le a) = \int_{-\infty}^{a} p(x)\,dx$ (CDF, PDF)

$\int_{-\infty}^{\infty} p(x)\,dx = 1$ $\quad p(x) \ge 0$

Gaussian/Normal dist^n

$N(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

$E(x) = \int_{-\infty}^{\infty} x\, p(x)\,dx = \mu$

$E[(x-\mu)^2] = \sigma^2$

Uniform $U(x; a, b) = \begin{cases} 1/b-a & a \le x \le b \\ 0 & \text{oth} \end{cases}$

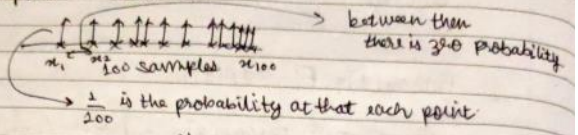
$\left[\frac{1}{b-a}\int_a^b x\,dx = \frac{1}{2}(b-a)\right]$ expected value

$E(f(x)) = \frac{1}{b-a}\int_a^b f(x)\,dx$

$H(x) = -\int_a^b \frac{1}{b-a} \log\left(\frac{1}{b-a}\right) dx = \log(b-a)$ entropy

variance = $E[(x - E(x))^2]$

Empirical Distribution



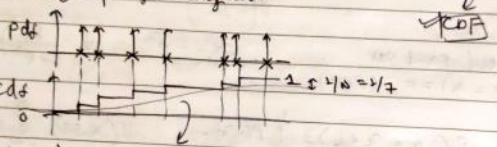
between them there is 2-0 probability

$\frac{1}{200}$ is the probability at that each point

↑ → discrete data $\sum_{n=1}^{N} \frac{1}{N} \delta(x - x_n)$ pdf

[Not generalizing] discrete random var.

overfitting Vs generalization



CDF

Approximate empirical CDF (Smoother Approximation of CDF)

$\int_{-\infty}^{x} dn$; $\frac{d}{dx}$ differentiation

Smoother Approx. of PDF.

The IID Assumption (Independent Identity distro.)

→ All samples are drawn from same sample (distro.)

$x_i, x_j \sim p_x$

Independent $p(x_i, x_j) = p(x_i) \cdot p(x_j)$

Identity $p_{x_i}(X = x_i) = p_{x_j}(X = x_j)$

$x_i \sim p_x$

$x_j \sim p_y$

Comparision of two dist.

$X = (x_1, x_2, \dots x_N)$ ← likelihood of whole data

$p(X) = p(x_1, x_2, \dots x_N)$

$= q(x_1) \cdot q(x_2) \cdots q(x_N)$

$p(X) = \prod_{i=1}^{N} q(x_i)$

If parameter is θ

$\arg\max_\theta \prod_{i=1}^{N} p_\theta(x_i)$

$L(X) = \log p(X) = \sum_{i=1}^{N} \log q(x_i)$

$\arg\max_\theta \sum_{i=1}^{N} \log(p_\theta(x_i))$

$\arg\max_\theta L_\theta(X)$

$\Rightarrow \prod_{i=1}^{N} p_\alpha(x_i) > \prod_{i=1}^{N} p_\beta(x_i)$

$p_\alpha(x)$ is Better Explanation of Data than $p_\beta(x)$

ML Estimate of μ, σ

$p(X) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

↓ take log, $\frac{d}{dx} = 0$

$\Rightarrow \mu_{ML} = \frac{1}{N} \sum_{i=1}^{N} x_i$

ML Estimate a, b

$p(X) = \prod_{i=1}^{N} \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & 0 \end{cases}$ $\Rightarrow$ $a \le \min(x_i)$, $b \ge \max(x_i)$

Independent:

- **Mathematical Definition**: Two random variables X_1 and X_2 are independent if the joint probability of them occurring together equals the product of their individual probabilities:

$$P(X_1 = x_1, X_2 = x_2) = P(X_1 = x_1) \cdot P(X_2 = x_2)$$

For more than two variables, this extends to:

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i)$$

This means the occurrence of one event doesn't affect the probability of the others.

Identically Distributed:

- **Mathematical Definition**: Random variables $X_1, X_2, \ldots, X_n$ are identically distributed if they all have the same probability distribution function (PDF) or probability mass function (PMF):

$$F_{X_1}(x) = F_{X_2}(x) = \cdots = F_{X_n}(x) = F_X(x)$$

where $F_X(x)$ is the cumulative distribution function (CDF) common to all X_i. This implies that each random variable has the same mean, variance, and other distributional characteristics.

IID in Practice:

- When we say a set of random variables $\{X_1, X_2, \ldots, X_n\}$ is IID, it means:
  1. **Independence**: X_i and X_j for any $i \neq j$ satisfy:

$$P(X_i \leq x_i \text{ and } X_j \leq x_j) = P(X_i \leq x_i) \cdot P(X_j \leq x_j)$$

  2. **Identical Distribution**: Each X_i shares the same distribution function $F_X(x)$.

Why It Matters:

- **Central Limit Theorem (CLT)**: The concept of IID is central to the CLT, which states that the sum (or average) of a large number of IID random variables will be approximately normally distributed, regardless of the original distribution.

$$\text{If } X_1, X_2, \ldots, X_n \text{ are IID with mean } \mu \text{ and variance } \sigma^2, \text{ then as } n \to \infty:$$

$$\frac{1}{n}\sum_{i=1}^{n} X_i \sim \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right)$$

- **Law of Large Numbers (LLN)**: For IID random variables, the sample mean converges to the population mean as the sample size increases.

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} X_i = \mu$$

Mathematical Explanation

Suppose you have a dataset consisting of n independent and identically distributed (IID) observations $X_1, X_2, \ldots, X_n$ from some probability distribution with probability density function (PDF) $f(X;\theta)$, where θ represents the parameters of the distribution.

Likelihood Function

The likelihood function for the entire dataset is the joint probability (for discrete data) or joint density (for continuous data) of all observations. For IID data, because each X_i is independent and identically distributed, the likelihood $L(\theta)$ can be expressed as:

$$L(\theta) = P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n \mid \theta)$$

Since the data points are independent, this joint probability (or density) factorizes into the product of the individual probabilities (or densities):

$$L(\theta) = \prod_{i=1}^{n} P(X_i = x_i \mid \theta) = \prod_{i=1}^{n} f(x_i;\theta)$$

Key Insight

The factorization of the likelihood function is what allows us to simplify many complex problems in statistics and machine learning. When data points are IID, the complexity of calculating the joint likelihood of the entire dataset reduces to calculating the product of individual likelihoods.

This factorization is crucial in:

- **Maximum Likelihood Estimation (MLE)**: Where you find the parameter θ that maximizes $L(\theta)$.
- **Bayesian Inference**: Where you update the prior distribution based on the likelihood of observed data.

Likelihood function

Likelihood Function

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ and a statistical model parameterized by θ, the likelihood function $L(\theta \mid X)$ represents the probability of observing the data X given the parameter θ. For a set of independent observations from a distribution with parameter θ, the likelihood function is:

$$L(\theta \mid X) = \prod_{i=1}^{n} f(x_i \mid \theta)$$

where $f(x_i \mid \theta)$ is the probability density function (PDF) or probability mass function (PMF) of the distribution.

Maximizing the Likelihood Function

- **Objective**: We seek to find the value of θ that maximizes $L(\theta \mid X)$. This value of θ is called the **Maximum Likelihood Estimate (MLE)**.
- **Mathematical Expression**:

$$\hat{\theta} = \arg\max_{\theta} L(\theta \mid X)$$

  Here, $\hat{\theta}$ is the estimate of the parameter θ that maximizes the likelihood function.

Summary

- $\arg\max_\theta L(X)$ denotes the parameter value θ that maximizes the likelihood function given the data X.
- Finding this value involves differentiating the likelihood function (or its logarithm), setting the result to zero, and solving for θ.
- The result is known as the Maximum Likelihood Estimate (MLE) of the parameter θ.

Sufficient statistics are a key concept in statistical theory that help to simplify the process of parameter estimation. Here's a detailed explanation:

Definition of Sufficient Statistic

A statistic $T(X)$ is said to be **sufficient** for a parameter θ if it captures all the information about θ contained in the sample data X. In other words, once you know the value of the sufficient statistic, the data provides no additional information about the parameter θ.

Formally, a statistic $T(X)$ is sufficient for θ if the conditional probability distribution of the sample data X given $T(X)$ does not depend on θ.

Formal Definition: Factorization Theorem

A common way to establish that a statistic $T(X)$ is sufficient for θ is by using the **Factorization Theorem**. According to this theorem, a statistic $T(X)$ is sufficient for θ if the joint probability density function (PDF) or probability mass function (PMF) of the data X can be factorized into two parts:

$$f(x \mid \theta) = g(T(x) \mid \theta) \cdot h(x)$$

where:

- $g(T(x) \mid \theta)$ is a function of the statistic $T(x)$ and the parameter θ,
- $h(x)$ is a function of the sample data x but does not depend on θ.

where $f_2(x_i \mid \theta_2)$ is the PDF or PMF for Family 2 with parameter θ_2.

2. **Comparing Likelihoods Directly**

   Comparing the likelihoods $L_{\theta_1}(X)$ and $L_{\theta_2}(X)$ directly can be done to assess how well each family fits the data, but this comparison can be challenging if the distributions are not nested or if they have different parameter spaces.

   - **Likelihood Ratio**: If the two families are not nested or have different parameterizations, the likelihood ratio test may not be directly applicable. However, you can still compare the likelihood values of the best-fitting models from each family:

     $$\text{Likelihood Ratio} = \frac{L_{\hat{\theta}_1}(X)}{L_{\hat{\theta}_2}(X)}$$

     where $\hat{\theta}_1$ and $\hat{\theta}_2$ are the MLEs of θ_1 and θ_2, respectively.

   - **Normalized Comparison**: Normalize the likelihoods to compare their relative fit:

     $$\text{Relative Likelihood} = \frac{L_{\hat{\theta}_1}(X)}{L_{\hat{\theta}_1}(X) + L_{\hat{\theta}_2}(X)}$$

     This gives you a sense of how each model fits the data relative to the other.

5. **Practical Considerations**

   - **Model Complexity**: Consider the complexity of each model and whether the improvement in fit justifies the additional complexity.
   - **Domain Knowledge**: Use domain-specific knowledge to guide the choice of distribution family.

Bayesian Estimation

Bayesian Estimation

1. Bayesian Estimate: In Bayesian estimation, you incorporate prior beliefs about the parameters into the estimation process. The parameter θ is treated as a random variable with a prior distribution $p(\theta)$. The goal is to find the posterior distribution of θ given the observed data.

2. Mathematical Formulation: Using Bayes' Theorem, the posterior distribution $p(\theta \mid X)$ is:

$$p(\theta \mid X) = \frac{p(X \mid \theta) \cdot p(\theta)}{p(X)}$$

where:

- $p(X \mid \theta)$ is the likelihood function.
- $p(\theta)$ is the prior distribution of θ.
- $p(X)$ is the marginal likelihood or evidence, which ensures the posterior is a valid probability distribution.

3. Bayesian Point Estimates: Bayesian estimation often involves calculating the expected value of the parameter with respect to the posterior distribution. Common Bayesian point estimates include:

- **Posterior Mean**: The expected value of θ given the data:

  $$\hat{\theta}_{\text{Bayes}} = \mathbb{E}[\theta \mid X] = \int_{\Theta} \theta \cdot p(\theta \mid X)\, d\theta$$

- **Posterior Median**: The value of θ such that the posterior distribution has 50% probability on either side.
- **Posterior Mode**: The value of θ that maximizes the posterior distribution $p(\theta \mid X)$.

A confidence interval provides a range of values within which we expect the true population parameter (such as the mean) to lie with a certain level of confidence. Here's how to construct and interpret a confidence interval for the mean, given a sample and sample mean:

Steps to Construct a Confidence Interval for the Mean

1. **Calculate the Sample Mean ($\bar{x}$)**
   - This is the average of the sample data. It's a point estimate of the population mean.
2. **Determine the Sample Standard Deviation (s)**
   - This measures the variability of the sample data. If you have the sample standard deviation, use it. Otherwise, if you have the sample variance (s^2), take the square root to get s.
3. **Choose the Confidence Level ($1 - \alpha$)**
   - Common confidence levels are 95% or 99%. This confidence level determines the probability that the confidence interval contains the true population mean.
4. **Find the Critical Value**
   - **For a known population standard deviation**: Use the Z-distribution (standard normal distribution). The critical value is denoted as $Z_{\alpha/2}$.
   - **For an unknown population standard deviation**: Use the t-distribution. The critical value is denoted as $t_{\alpha/2,\text{df}}$, where df (degree ↓ freedom) is $n - 1$, and n is the sample size.

5. **Compute the Margin of Error (ME)**
   - **For large samples (or known population standard deviation):**

$$\text{ME} = Z_{\alpha/2} \times \frac{s}{\sqrt{n}}$$

   - **For small samples (unknown population standard deviation):**

$$\text{ME} = t_{\alpha/2,\text{df}} \times \frac{s}{\sqrt{n}}$$

6. **Construct the Confidence Interval**
   - The confidence interval is given by:

$$\bar{x} \pm \text{ME}$$

   - So the interval is:

$$[\bar{x} - \text{ME}, \bar{x} + \text{ME}]$$

- **Determine the p-value:**
  Use the calculated test statistic and the degrees of freedom to find the p-value from the t-distribution. This p-value tells you the probability of observing a test statistic as extreme as, or more extreme than, the one calculated, assuming the null hypothesis is true.
- **Compare the p-value to the Significance Level (α\alphaα):**
  - **Significance Level (α\alphaα)**: Commonly set at 0.05 (5%).
  - **Decision Rule**:
    - If p≤αp \leq \alphap≤α, reject the null hypothesis H0H_0H0. This indicates that there is a significant difference between the means of the two groups.
    - If p>αp > \alphap>α, do not reject H0H_0H0. This indicates that there is not enough evidence to conclude that the means are different

LINEAR REGRESSION

- Linear regression is a powerful and commonly used model due to its simplicity and flexibility. Its utility includes:
  - **Simplest to code**: The implementation of linear regression is straightforward in most programming languages. Libraries such as Python's `scikit-learn` offer built-in functions to perform linear regression in just a few lines of code.
  - **Simplest to understand mathematically**: Linear regression is based on simple concepts like finding the line of best fit to minimize the difference between predicted and actual values (using the least squares method). This makes it an excellent model for teaching and explaining fundamental statistical concepts.
  - **Nonlinear feature extraction can be used with linear solutions**: Despite its linear nature, linear regression can still capture nonlinear relationships in the data by using polynomial or interaction terms as additional features. This allows for increased model flexibility without losing the simplicity of a linear model.

- $$t = Xw + e$$

  Where:

  - X is the matrix of input features,
  - w is the vector of weights (parameters),
  - $e \sim \mathcal{N}(0, \sigma^2)$, i.e., the noise is normally distributed with mean 0 and variance σ^2.

  Alternatively, using precision $\beta = \frac{1}{\sigma^2}$, we can rewrite the noise distribution as $e \sim \mathcal{N}(0, \frac{1}{\beta})$.

  Thus, the conditional probability distribution $p(t|X, w, \beta)$ is the **likelihood of observing** t, assuming the model parameters and input data are known.

  ### Gaussian Likelihood:

  Since the noise is Gaussian, the targets t are also Gaussian around the predicted values Xw:

  $$p(t|X, w, \beta) = \mathcal{N}(t|Xw, \frac{1}{\beta})$$

- Likelihood in Linear Regression

Gaussian Likelihood:

Since the noise is Gaussian, the targets t are also Gaussian around the predicted values Xw:

$$p(t|X,w,\beta) = \mathcal{N}(t|Xw, \frac{1}{\beta})$$

This means that t follows a normal distribution with:

- Mean Xw (the predictions from the model),
- Variance $\frac{1}{\beta}$, or precision β.

In more detail, the likelihood function is:

$$p(t|X,w,\beta) = \prod_{i=1}^{m} \mathcal{N}(t_i|X_i w, \frac{1}{\beta})$$

Where each target t_i is drawn from a normal distribution centered at $X_i w$ (the predicted value for input X_i) and with variance $\frac{1}{\beta}$.

Expanded Form:

The probability density function for a Gaussian distribution $\mathcal{N}(t_i|X_i w, \frac{1}{\beta})$ is:

$$p(t_i|X_i,w,\beta) = \frac{\sqrt{\beta}}{\sqrt{2\pi}} \exp\left(-\frac{\beta}{2}(t_i - X_i w)^2\right)$$

For the entire dataset, the likelihood is the product of these individual probabilities:

$$p(t|X,w,\beta) = \left(\frac{\sqrt{\beta}}{\sqrt{2\pi}}\right)^m \exp\left(-\frac{\beta}{2}\sum_{i=1}^{m}(t_i - X_i w)^2\right)$$

Where m is the number of data points.

Log-Likelihood:

The log-likelihood is often used to simplify optimization. Taking the logarithm of $p(t|X,w,\beta)$ gives:

$$\log p(t|X,w,\beta) = \frac{m}{2}\log\beta - \frac{m}{2}\log 2\pi - \frac{\beta}{2}\sum_{i=1}^{m}(t_i - X_i w)^2$$

This log-likelihood is maximized to find the optimal values of w and β.

To **maximize the likelihood** in the context of a linear regression model with Gaussian noise, we can use the **maximum likelihood estimation (MLE)** approach. This involves finding the values of the parameters w (the weights) that maximize the likelihood function $p(t|X,w,\beta)$.

2. **Log-Likelihood**:

To simplify the maximization, we take the **logarithm** of the likelihood function, which gives the **log-likelihood**:

$$\log p(t|X, w, \beta) = \frac{m}{2} \log \beta - \frac{m}{2} \log 2\pi - \frac{\beta}{2} \sum_{i=1}^{m} (t_i - X_i w)^2$$

- Since the first two terms do not depend on w, we focus on maximizing the last term:

$$\log p(t|X, w) = -\frac{\beta}{2} \sum_{i=1}^{m} (t_i - X_i w)^2$$

This is equivalent to minimizing the **residual sum of squares (RSS)**:

$$\text{RSS}(w) = \sum_{i=1}^{m} (t_i - X_i w)^2$$

3. **Set Gradients with Respect to w to Zero**:

To maximize the log-likelihood (or equivalently, minimize the RSS), we take the gradient of the log-likelihood with respect to w, and set it equal to zero:

$$\frac{\partial}{\partial w} \left(-\frac{\beta}{2} \sum_{i=1}^{m} (t_i - X_i w)^2 \right) = 0$$

- Expanding the gradient:

$$\frac{\partial}{\partial w} \left(\sum_{i=1}^{m} (t_i - X_i w)^2 \right) = -2X^\top (t - Xw)$$

Set this gradient equal to zero:

$$-2X^\top (t - Xw) = 0$$

Simplifying:

$$X^\top X w = X^\top t$$

4. **Solve for w**:

The solution to this equation is the **normal equation** for linear regression:

- $$w = (X^\top X)^{-1} X^\top t$$

This gives the value of w that maximizes the likelihood, which is the **least squares solution**.

BIAS - VARIANCE TRADEOFF
MEAN SQUARED ERROR

1. Expected Loss (MSE)

The **Mean Squared Error (MSE)** measures the average squared difference between the predicted values $\hat{y}$ and the true values y. Mathematically, the MSE for a regression model is given by:

$$\text{MSE} = \mathbb{E}[(\hat{y} - y)^2]$$

Where:

- $\hat{y}$ is the predicted value from the model,
- y is the true value.

The MSE is an important metric because it quantifies the accuracy of the model by averaging the squared errors over all data points.

2. Bias-Variance Decomposition

The **bias-variance decomposition** helps explain the sources of error in a model's predictions. The total expected error (MSE) can be decomposed into three terms:

$$\text{MSE} = \underbrace{(\mathbb{E}[\hat{y}] - y)^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}[(\hat{y} - \mathbb{E}[\hat{y}])^2]}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Irreducible Error}}$$

Where:

- **Bias**: Measures how far the average prediction $\mathbb{E}[\hat{y}]$ is from the true value y. It represents the error introduced by the model's assumptions. A high bias indicates that the model is too simplistic and underfits the data.

$$\text{Bias}^2 = (\mathbb{E}[\hat{y}] - y)^2$$

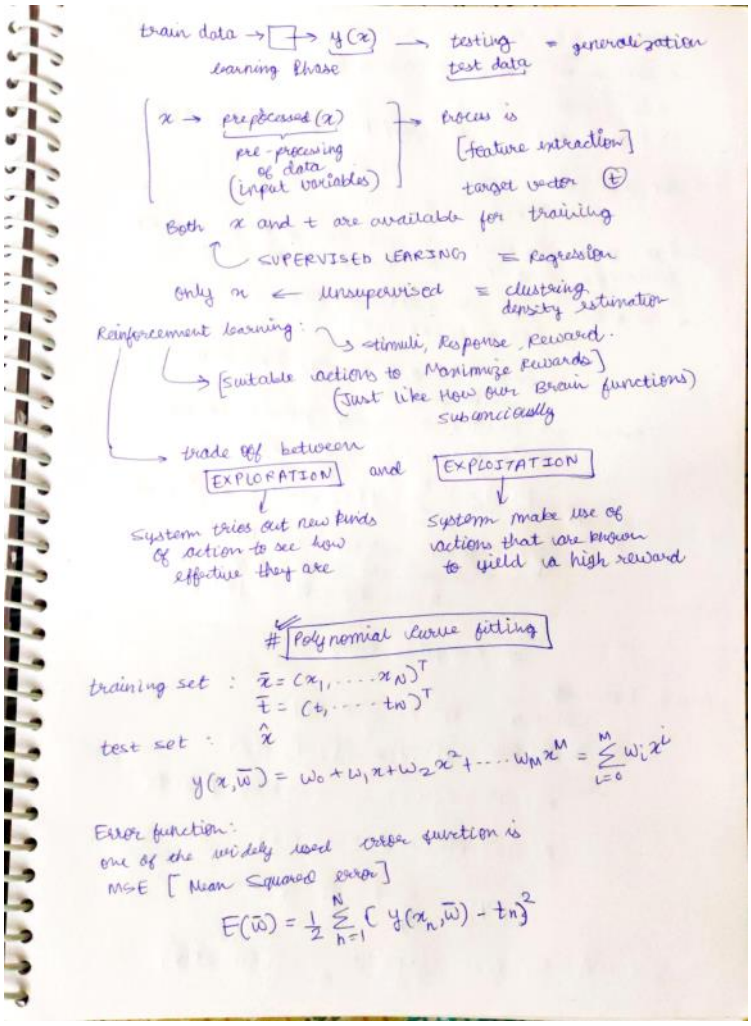
- **Variance**: Measures how much the predictions $\hat{y}$ vary around their expected value $\mathbb{E}[\hat{y}]$. High variance indicates that the model is too sensitive to the specific training data and overfits the data.

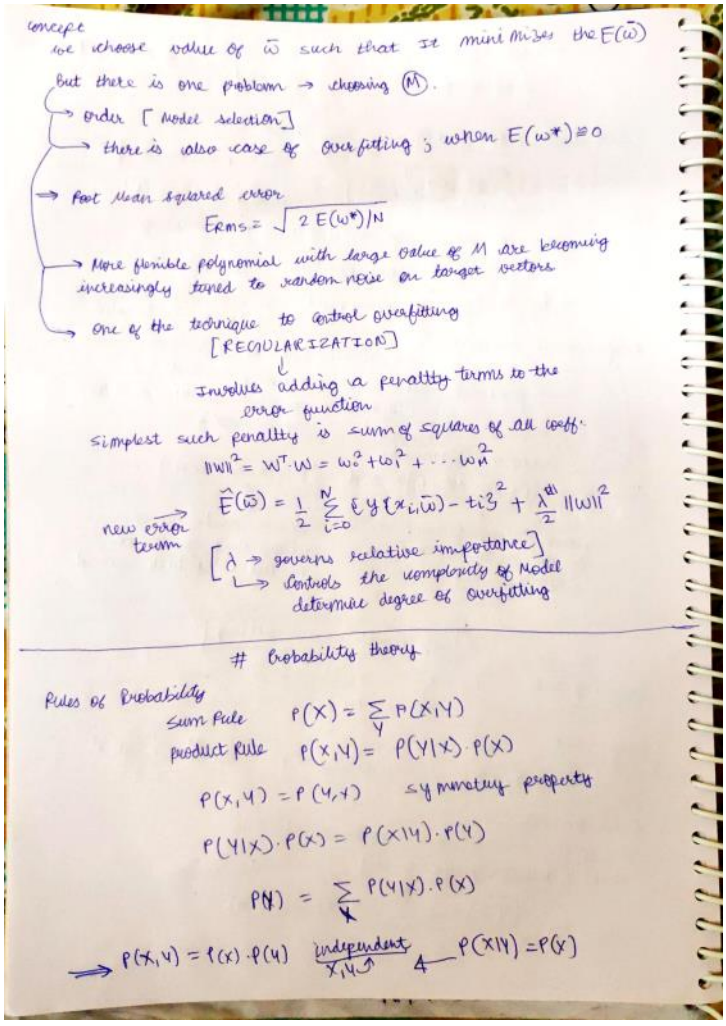
$$\text{Variance} = \mathbb{E}[(\hat{y} - \mathbb{E}[\hat{y}])^2]$$

- **Irreducible Error**: Represents the inherent noise σ^2 in the data that no model can capture. This error is due to randomness or measurement errors and cannot be reduced by improving the model.

Bias-Variance Tradeoff

- A model with **high bias** (e.g., a simple model) may underfit the data, meaning it misses important patterns and relationships, resulting in higher errors.
- A model with **high variance** (e.g., a complex model) may overfit the training data, meaning it captures noise or fluctuations specific to the training set, which increases error when making predictions on unseen data.
- The goal in model selection is to balance bias and variance to minimize the total error (MSE).





concept
we choose value of $\bar{w}$ such that it minimizes the $E(\bar{w})$
but there is one problem → choosing (M).
→ order [Model selection]
→ there is also case of overfitting ; when $E(w^*) \cong 0$
⇒ Root Mean squared error
$E_{RMS} = \sqrt{2E(w^*)/N}$
→ More flexible polynomial with large value of M are becoming increasingly tuned to random noise on target vectors
→ One of the technique to control overfitting
[REGULARIZATION]
↓
Involves adding a penalty terms to the error function
simplest such penalty is sum of squares of all coeff.
$\|w\|^2 = w^T \cdot w = w_0^2 + w_1^2 + \cdots w_n^2$
new error term → $\tilde{E}(\bar{w}) = \frac{1}{2}\sum_{i=0}^{N} \{y(x_i, \bar{w}) - t_i\}^2 + \frac{\lambda}{2}\|w\|^2$
[λ → governs relative importance]
↳ controls the complexity of model determine degree of overfitting

---

Probability theory

Rules of Probability
Sum Rule $P(X) = \sum_Y P(X,Y)$
product Rule $P(X,Y) = P(Y|X) \cdot P(X)$
$P(X,Y) = P(Y,X)$ symmetry property
$P(Y|X) \cdot P(X) = P(X|Y) \cdot P(Y)$
$P(Y) = \sum_X P(Y|X) \cdot P(X)$
⇒ $P(X,Y) = P(X) \cdot P(Y)$ independent X,Y ↑ ↓ $P(X|Y) = P(X)$

Probability density

$$P(x \in (a,b)) = \int_a^b p(x)\,dx$$

$p(x) \geq 0$ — PDF

$$\int_{-\infty}^{\infty} p(x)\,dx = 1$$

Change of variable: $x = g(y)$

$$f(x) = \tilde{f}(y) = f(g(y))$$

$$p_y(y) = p_x(g(y))\,|g'(y)|$$

$\rightarrow$ $$P(z) = \int_{-\infty}^{z} p(x)\,dx$$ — CDF

$\rightarrow$ $P'(x) = p(x)$

Joint PDF $\rightarrow$ $\bar{x} = (x_1, x_2, \cdots x_D)$

$p(\bar{x}) = p(x_1, x_2, \cdots x_D)$

$p(\bar{x}) \geq 0$

$$\int_{-\infty}^{\infty} p(\bar{x})\,d\bar{x} = 1$$

for continuous PDFs: $p(x) = \int_y p(x,y)\,dy$

$p(x,y) = p(y|x)\,p(x)$

EXPECTATION

expectation of some $f(x)$ is average value of $f(x)$ denote by

$$E(f) = \sum_x p(x) f(x)$$

$$E(f) = \int_x p(x) f(x)\,dx$$

$$E_x(f(x,y)) = \int_x p(x) \cdot f(x,y)\,dx = g(y)$$

$$E_x(f|y) = \int p(x|y) \cdot f(x)\,dx$$

VARIANCE

$$\text{var}[f] = E\left[\left(E(f(x)) - f(x)\right)^2\right]$$

$$\text{Var}[X] = E[x^2] - (E[x])^2$$

COVARIANCE

$$cov[x,y] = \mathbb{E}_{x,y}\left[\{x - \mathbb{E}(x)\}\{y - \mathbb{E}(y)\}\right]$$

$$= \mathbb{E}_{x,y}[xy] - \mathbb{E}(x)\mathbb{E}[y]$$

$$cov[\bar{x},\bar{y}] = \mathbb{E}_{\bar{x},\bar{y}}[\bar{x}\bar{y}^T] - \mathbb{E}[\bar{x}]\mathbb{E}[\bar{y}^T]$$

$\rightarrow$ $cov[x,x] \triangleq cov[x]$

Bayesian Probability

$$P(\bar{w}|D) = \frac{P(D|\bar{w}) \cdot P(\bar{w})}{P(D)}$$

($P(D|\bar{w})$: likelihood function)

$\bar{D}$ is some observed data

$D = \{t_1, t_2, \cdots t_n\}$

[posterior $\propto$ likelihood $\times$ prior]

Bayes' theorem

$$P(D) = \int P(D|\bar{w}) \cdot P(\bar{w})\, d\bar{w}$$

17/9/2024

Gaussian distribution

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\int_{-\infty}^{\infty} \mathcal{N}(x|\mu,\sigma^2) = 1 \rightarrow \text{normalized}$$

$$\mathbb{E}[x] = \mu$$

$$\mathbb{E}[x^2] = \mu^2 + \sigma^2$$

$$var[x] = \sigma^2$$

D-dimensional:

$$\mathcal{N}(\vec{x}|\vec{\mu},\vec{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\vec{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \vec{\Sigma}^{-1}(\vec{x}-\vec{\mu})\right)$$

$\vec{\mu}$ $\rightarrow$ mean vector

$\vec{\Sigma}$ $\rightarrow$ covariance Matrix

i.i.d (independent & identically distributed)

$$p(\vec{x}/\mu,\sigma^2) = \prod_{i=1}^{N} N(x_i/\mu,\sigma^2) \quad \text{given } \mu,\sigma^2$$

likelihood

log likelihood $\ln p(\vec{x}/\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$

↳ Maximise this wrt to μ, σ^2 → we get

Maximum likelihood solution

curve fitting

$\vec{x}$ (input), $\vec{t}$ (target) } → assume gaussian distribution with mean $= y(\vec{x},\vec{w})$ (weights)

$$p(\vec{t}/\vec{x},\vec{w},\beta) = N(t/y(\vec{x},\vec{w}),\beta^{-1})$$

⇒ β = precision parameter (inverse of variance of distribution)

step ① Now we use training data $(\vec{x},\vec{t})$ to determine values of $\vec{w}$ and β by Maximum LIKELIHOOD.

$$p(\vec{t}/\vec{x},\vec{w},\beta) = \prod_{i=1}^{N} N(t_n/x_n,\vec{w},\beta)$$

step② Maximizing log likelihood is equivalent to Minimizing sum-of-squares error.

BAYESIAN APPROACH

→ introduce polynomial distribution over $\vec{w}$

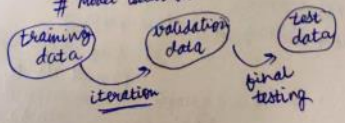
$$p(\vec{w}/\alpha) = N(\vec{w}/\vec{0},\alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{\frac{(M+1)}{2}} \exp\left(-\frac{\alpha}{2}\vec{w}^T\vec{w}\right)$$

[α → Hyperparameters] → find of (Hyperparameter tuning)

given: training data $\vec{x}$ and $\vec{t}$
new test point x
goal: predict the value of t
} → Inference

$$p(t/x,\vec{x},\vec{t}) = \int p(t/x,\vec{w})\cdot p(\vec{w}/\vec{x},\vec{t})\,d\vec{w}$$

Model selection



⇒ Minimizing misclassification rate (DECESION THEORY)
→ divide input space $\vec{x}$ into K region R_k are assigned to class C_k. [decesion boundaries]

$$P(\text{mistake}) = p(x \in R_1, C_2) + p(x \in R_2, C_1)$$

$$P(\text{correct}) = \sum_{k=1}^{K} p(x \in R_k | C_k) \, dx$$

M2: healthy person → cancer
M1: cancer patient → healthy

⇒ Minimizing expected loss
↳ Mistake 1 weight → Mistake 2 weight
→ loss function / cost function → minimize this
→ utility function → maximize this

⇒ rejection option
↳ introducing θ (threshold value) and rejecting those input x for which largest of posterior prob. $p(C_k|x)$ is less than $= \theta_k$.
↳ extend the rejection criteria to minimize loss.

⇒ Inference and decesion
classification
↙ Inference stage → use use training data to learn a model $P(C_k|x)$
↘ decision stage → use posteri probabilities to make optimal class assignment

→ alternative approach
↳ simply learn a function that map x to decesion directly
[DISCRIMINANT FUNCTIONS]

ⓐ Inference, decesion → posterior probability
ⓑ Discriminant Model

Better → ① Minimizing Risk → loss Matrix
② Reject options →
③ Compensating for class priors → Need of Balanced data set (equal no. of data sets corresponding to each class)

④ Model Combinations
x-Ray Image (X_I)
Blood sample (X_B)
$[P(x_I, x_B | C_k) = p(x_I | C_k) \, p(x_B | C_k)]$ (Conditional Independence)

Loss functions for Regression

Consider Basic Squared loss:

$$L(t, y(x)) = \{y(x) - t\}^2$$

Average loss

$$E[L(t, y(x))] = \iint \{y(x) - t\}^2 p(x,t)\, dx\, dt$$

$$\frac{\delta E[L]}{\delta y(x)} = 2 \int (y(x) - t)\, p(x,t)\, dt = 0$$

$$y(x) = \int t \cdot \frac{p(x,t)}{p(x)}\, dt = \int t \cdot p(t|x) = E(t|x)$$

Regression function

Minkowski loss → generalized version of squared loss.

$$\left[L_q(t, y(x)) = \{y(x) - t\}^q\right]$$

[INFORMATION THEORY]

$p(x)$ → probability distribution

$h(x)$ → quantity & monotonic function of $p(x)$

If we have 2 events x and y, and they are unrelated then information gained from both of them is sum of them $h(x,y) = h(x) + h(y)$

Similarly $p(x,y) = p(x) \cdot p(y)$ (Independent)

$$h(x) = -\log_2 p(x)$$

→ Average amount of information transmitted is (wrt $p(x)$)

$$H[x] = -\sum_x p(x) \log_2(x)$$ unit → bits

expectation of $h(x)$ wrt $p(x)$ — called ENTROPY of random var

↳ Huffman Encoding

⇒ Entropy is a lower bound on no. of bits to transmit state of random variables.

→ in case of natural log: units → nat

↳ choosing objects from N objects in order; N! ways

In i^{th} bin; reordering objects $n_i!$

$W = \frac{N!}{\prod n_i!}$ ⇒ Multiplicity

Entropy is defined as log of multiplicity scaled to some const.

$$H = \frac{1}{N} \log W = \frac{1}{N} \ln N! - \frac{1}{N} \sum_i \ln n_i!$$

Stirling approx. $\log N! \cong N \log N - N$ as $N \to \infty$

$$H = -\lim_{N \to \infty} \sum_i \left(\frac{n_i}{N}\right) \log\left(\frac{n_i}{N}\right) = -\sum_i p_i \ln p_i$$

Specific arrangement of object in bins → MICROSTATE

overall distribution of N objects → MACROSTATE

→ W is also weight of Macrostate.

$$H[p] = -\sum_i p(x_i) \log p(x_i) \qquad P(X = x_i) = p(x_i)$$

$$H[x] = -\int p(x) \ln p(x)\, dx$$ Continuous

→ In discrete distrb ⇒ Max entropy [Uniform distro]

↳ Continuous → [Gaussian distro]

$$\int_{-\infty}^{\infty} p(x)\, dx = 1$$

$$\int_{-\infty}^{\infty} x\, p(x)\, dx = \mu$$ Mean

$$\int_{-\infty}^{\infty} (x - \mu)^2 p(x)\, dx = \sigma^2$$ variance

$$H[y|x] = -\iint p(y,x) \ln(p(y|x))\, dy\, dx$$ Conditional entropy

$$H[x,y] = H[y|x] + H[x]$$

→ $f(x)$ → CONVEX function

property that every chord lies on the function.

Jensen's Inequality $f(\mathbb{E}(x)) \leq \mathbb{E}[f(x)]$

$$f\left(\int x p(x)\, dx\right) \leq \int f(x) p(x)\, dx$$

Ch-2 : Probability Distro.

learn: model probability distribution $p(x)$ of random var. X given finite set of $X \in \{x_1, x_2, \dots, x_N\}$ of observation

[DENSITY ESTIMATION]

① Bernoulli distro.

$$\text{Bern}(x|\mu) = \mu^x (1-\mu)^{1-x}$$

$$E(x) = \mu$$

$$\text{Var}(x) = \mu(1-\mu)$$

construction: $D = \{x_1, x_2, \dots x_N\}$

$$P(D|\mu) = \prod_{n=1}^{N} P(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n} (1-\mu)^{1-x_n}$$

log likelihood function: $\ln p(D|\mu) = \sum_{n=1}^{N} \{x_n \ln \mu + (1-x_n) \ln(1-\mu)\}$

↓ $d/d\mu$

get Maximum likelihood estimator

$$\mu_{ML} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

→ same as sample mean

② Binomial distro.

$$\text{Bin}(m|N,\mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m}$$

$$E[m] = N\mu \; ; \; \text{Var}[m] = N\mu(1-\mu)$$

③ Beta distro

$$\text{Beta}(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1} (1-\mu)^{b-1}$$

$$\int_0^1 \text{Beta}(\mu|a,b)\, d\mu = 1$$

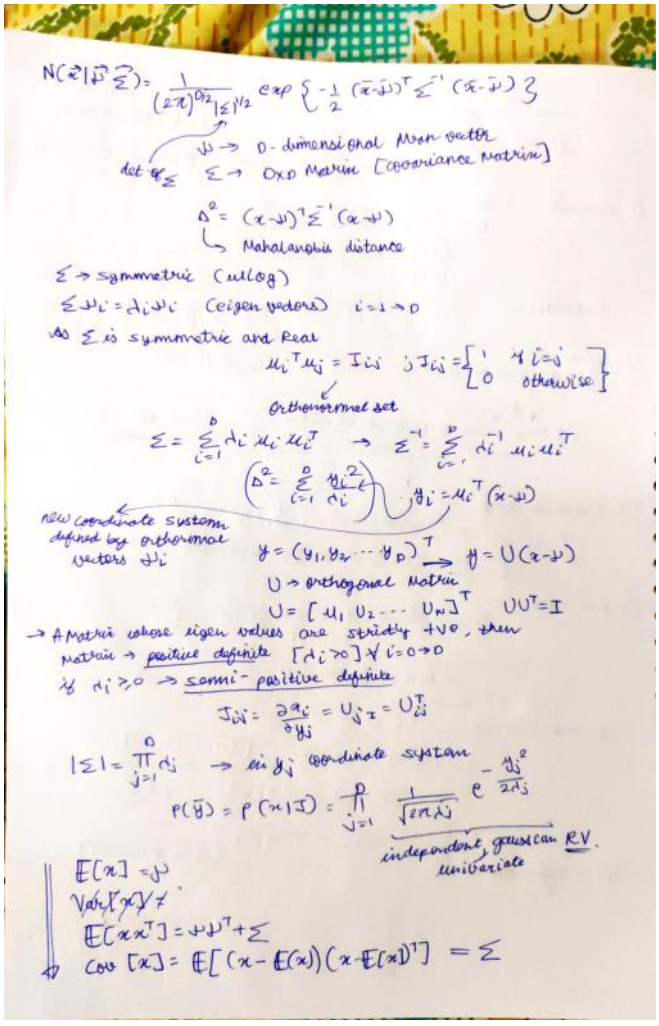
gamma function: $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x}\, dx$

for natural number: $\boxed{\Gamma(n) = (n-1)!}$

$$\Gamma(\alpha+1) = \alpha\, \Gamma(\alpha) \qquad \Gamma\left(\tfrac{1}{2}\right) = \sqrt{\pi}$$

$$E[\mu] = \frac{a}{a+b} \; ; \; \text{Var}[\mu] = \frac{ab}{(a+b)^2(a+b+1)}$$

$a, b \rightarrow$ hyperparameters


$N(\vec{x}|\vec{\mu},\vec{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right\}$
$\det_\Sigma$
$\mu \to$ D-dimensional Mean vector
$\Sigma \to$ DxD Matrix [covariance Matrix]
$\Delta^2 = (x-\mu)^T\Sigma^{-1}(x-\mu)$
↳ Mahalanobis distance
$\Sigma \to$ symmetric (ullog)
$\Sigma u_i = \lambda_i u_i$ (eigen vectors) $i=1 \to D$
As Σ is symmetric and Real
$u_i^T u_j = I_{ij}$; $I_{ij} = \begin{cases} 1 & \forall i=j \\ 0 & \text{otherwise} \end{cases}$
orthonormal set
$\Sigma = \sum_{i=1}^{D} \lambda_i u_i u_i^T \rightarrow \Sigma^{-1} = \sum_{i=1}^{D} \lambda_i^{-1} u_i u_i^T$
$\left(\Delta^2 = \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i}\right)$; $y_i = u_i^T(x-\mu)$
new coordinate system defined by orthonormal vectors u_i
$y = (y_1, y_2 \cdots y_D)^T \rightarrow y = U(x-\mu)$
$U \to$ orthogonal Matrix
$U = [u_1\ u_2 \cdots u_N]^T$ $UU^T = I$
→ A Matrix whose eigen values are strictly +ve, then Matrix → positive definite $[\lambda_i > 0]\ \forall\ i=0 \to D$
if $\lambda_i \geq 0$ → semi-positive definite
$J_{ij} = \frac{\partial x_i}{\partial y_j} = U_{ji} = U_{ij}^T$
$|\Sigma| = \prod_{j=1}^{D} \lambda_j$ → in y_j coordinate system
$p(\vec{y}) = p(x|J) = \prod_{j=1}^{D} \frac{1}{\sqrt{2\pi\lambda_j}} e^{-\frac{y_j^2}{2\lambda_j}}$
independent, gaussian univariate R.V.
$E[x] = \mu$
$Var[x] = ?$
$E[xx^T] = \mu\mu^T + \Sigma$
$\text{Cov}[x] = E[(x-E(x))(x-E[x])^T] = \Sigma$

Conditional gaussian distrb.

$N(\bar{x} \mid \bar{\mu}, \Sigma)$ — $\bar{x}$ partitioned into M components x_a + remaining D−M —" x_b

$$\bar{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \rightarrow \bar{\mu} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \qquad \Sigma^T = \Sigma \;\&\; \Sigma_{ba}{}^T = \Sigma_{ab}$$

$\Lambda = \Sigma^{-1}$ — Precision Matrix

$$\Lambda = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}$$

$P(x) = P(x_a, x_b)$ — Joint distribution ; let say x_b is observed value. Now find prob. distrn over x_a

coefficient in exponent is

$$(x-\mu)^T \Sigma (x-\mu) = (x_a-\mu_a)^T \Lambda_{aa}(x_a-\mu_a) + (x_a-\mu_a)^T \Lambda_{ab}(x_b-\mu_b) + (x_b-\mu_b)^T \Lambda_{ba}(x_a-\mu_a) + (x_b-\mu_b)^T \Lambda_{bb}(x_b-\mu_b).$$

$P(x_a \mid x_b)$ → also gaussian

Now find its Mean & Variance $(\mu_{a|b}, \Sigma_{a|b})$

Conditional prob.

$$\Sigma_{a|b} = \Lambda_{aa}^{-1}$$

$$\mu_{a|b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b)$$

$P(x_a \mid x_b) = N(x_a \mid \mu_{a|b}, \Sigma_{a|b})$ → gaussian distrb.

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{pmatrix}$$

$M = (A - BD^{-1}C)^{-1}$

$M^{-1} = A - BD^{-1}C$

Shur complement of matrix X

$$\begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}$$

$$\Lambda_{aa} = (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1}$$

$$\Lambda_{ab} = -(\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1}\Sigma_{ab}\Sigma_{bb}^{-1}$$

put these → we got

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b)$$

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

→ Mean of conditional dist $P(x_a \mid x_b)$ is linear function of x_b and covariance Matrix is independent of x_a.

[Linear Gaussian Model]

Marginal Gaussian distro.

$$p(x_a) = \int p(x_a, x_b)\, dx_b$$

$\mathbb{E}[x_a] = \mu_a$

$\text{cov}[x_a] = \Sigma_{aa}$

Gaussian Marginal distribution

$p(x)$

Gaussian Conditional distribution

→ linear gaussian model

↳ $y = Ax + b$

$p(x) = \mathcal{N}(x \mid \mu, \Lambda^{-1})$

$p(y \mid x) = \mathcal{N}(y \mid Ax + b, L^{-1})$

$z = \begin{pmatrix} x \\ y \end{pmatrix} \rightsquigarrow \mathbb{E}[z] = \begin{pmatrix} \mu \\ A\mu + b \end{pmatrix}$

$\mathbb{E}(y) = A\mu + b$

$\text{cov}[y] = L^{-1} + A\Lambda^{-1}A^T$

18/9/24

maximum likelihood for gaussian

$$\ln p(\vec{X} \mid \mu, \Sigma) = -\frac{ND}{2}\ln 2\pi - \frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(\vec{x}_N - \mu)^T \Sigma^{-1}(\vec{x}_n - \mu)$$

depends on 2 terms

$X = (x_1, x_2, \ldots x_N)^T$

↳ each obs^n x_n assumed to be drawn independently from gaussian distribution.

NXD

$\sum_{n=1}^{N} \vec{x}_n$

$\sum_{n=1}^{N} \vec{x}_n \vec{x}_n^T$

Sufficient statistics for gaussian distro.

$$\frac{\partial \ln p(\vec{x} \mid \mu, \Sigma)}{\partial \mu} = \sum_{n=1}^{N} \Sigma^{-1}(x_n - \mu) = 0$$

→ Maximum likelihood estimate of mean

$$\vec{\mu}_{MLE} = \frac{1}{N}\sum_{n=1}^{N} \vec{x}_n$$

$$\Sigma_{MLE} = \frac{1}{N}\sum_{n=1}^{N} (x_n - \mu_{ML})(x_n - \mu_{ML})^T$$

Bayesian Inference

$p(\mu \mid X) \propto p(X \mid \mu) \cdot p(\mu)$

student's t distribution

we have univariate gaussian $\mathcal{N}(x \mid \mu, \tau^{-1})$ together with prior gamma $\text{Gam}(\tau \mid a, b)$

$$p(x \mid \mu, a, b) = \int_0^\infty \mathcal{N}(x \mid \mu, \tau^{-1}) \cdot \text{Gam}(\tau \mid a, b)\, d\tau$$

New parameter $\nu = 2a$, $\lambda = a/b$ $= St(x \mid \mu, \lambda, \nu)$

$St(x|\mu,\lambda,\nu)$ → degrees of freedom
precision of t distro
for $\nu=1$ St → Cauchy distro
for $\nu\to\infty$ St → Gaussian $N(x|\mu,\lambda^{-1})$

Periodic var.

$p(\theta)\geq 0$

$\int_0^{2\pi} p(\theta)\,d\theta = 1$

$p(\theta+2\pi)=p(\theta)$

$p(\theta+k2\pi)=p(\theta)$

$\vec{x}=(\vec{r}\cos\vec{\theta}, \vec{r}\sin\vec{\theta})$

⇒ Mixtures of Gaussians
Superposition of K gaussian densities

$$p(\vec{x})=\sum_{k=1}^{K}\pi_k N(\vec{x}|\mu_k,\Sigma_k)$$

Mixing coefficients

$\left(\sum_{k=1}^{K}\pi_k=1\right)$
$p(\vec{x})\geq 0$
↳ $\pi_k\geq 0$

Exponential family

$p(\vec{x}|\eta)=h(x)\cdot g(\eta)\exp(\eta^T u(x))$

$g(\eta)\int h(\vec{x})\exp(\eta^T u(x))dx=1$

η → parameter natural

Normalization factor

① Bernoulli:

$p(x|\mu)=\mu^x(1-\mu)^{1-x}$

$p(x|\mu)=\exp(x\ln\mu+(1-x)\ln(1-\mu))$

$=(1-\mu)\exp\left\{\ln\left(\frac{\mu}{1-\mu}\right)x\right\}$ → $\eta=\ln\left(\frac{\mu}{1-\mu}\right)$

$\mu=\sigma(\eta)$ → logistic sigmoid function

$\mu=\frac{1}{1+\exp(-\eta)}$

$p(x|\eta)=\sigma(-\eta)\exp(\eta x)$

$1-\sigma(\eta)=\sigma(-\eta)$
$u(x)=x$
$h(x)=1$
$g(\eta)=\sigma(-\eta)$
→ for Bernoulli

② Consider Multinomial distro.

$p(x|\mu)=\prod_{k=1}^{M}\mu_k^{x_k}=\exp\left(\sum_{k=1}^{M}x_k\ln\mu_k\right)$; $p(x|\eta_i)=\exp(\eta^T x)$

$\sum_{k=1}^{M}\mu_k=1$

$\mu(x)=x$
$u(x)=1$
$g(\eta)=1$

$\eta_k=\ln\mu_k$
$\eta=\{\eta_1,\eta_2,\dots\eta_M\}$

Softmax function

$\ln\left(\frac{\mu_k}{1-\sum_j\mu_k}\right)=\eta_k$ → $\mu_k=\sigma(\eta_k)$ ⇒ $\mu_k=\frac{e^{\eta_k}}{1+\sum_j e^{\eta_j}}$

Maximum likelihood & sufficient statistics

$$p(\vec{x}|\theta) = h(x)\, g(\eta)\, \exp\left(\eta^T u(x)\right)$$

$$\nabla g(\eta) \int h(x)\, e^{\eta^T u(x)}\, dx$$

$$g(\eta) \int h(x)\, e^{\eta^T u(x)} \cdot u(x)\, dx = 0$$

$$-\nabla \ln g(\eta) = E[u(x)]$$

$\sum_n u(x_n)$ — sufficient stats

$$-\nabla \ln g(\eta_{MLE}) = \frac{1}{N} \sum_{n=1}^{N} u(x_n)$$

Non parametric methods

Parametric → might give give poor model result in poor prediction

Histogram density models

↳ Partitions x into distinct bins with width Δ_i then count number n_i of obs. x falling into bin i.

$$p_i = \frac{n_i}{N \Delta_i}$$ for normalization

$\int p(x)\, dx = 1$ ↳ often bins chosen are of same width $\Delta_i = \Delta$.

(neither too large nor too small)

$\Delta \to$ very small [spiky too] $\Delta \to$ very large [too smooth] → this model also dependent of choice of edge location for bins.

Limitations of Histogram

↳ scaling with dimensionality

↳ divide each variable with D-dim space into M bins, the total no. of bins = M^D → exponential scaling

Kernel Density Estimators

→ Let we have collected a data set comprising N observation drawn from $p(x)$.

→ each data point has prob. P of falling inside within R (region).

→ total number of point k that lie inside R. [distrib. to Binomial dist]

$$\text{Bin}(k|N,P) = \frac{N!}{k!(N-k)!} P^k (1-P)^{1-k}$$

for large N, $k = NP$

→ Assume R is sufficiently small that prob. density $p(x)$ is roughly constant

$P \simeq p(x) \cdot V_R$ ← volume of R.

$$p(x) = \frac{k}{NV}$$

$p(x) = \frac{K}{NV}$

fix K and determine value of V → K-nearest neighbour

fix V and determine K from given data → kernel density estimation

→ In order to count number K of points falling into region;

↳ f^n: $k(u) = \begin{cases} 1 & |u_i| \le 1/2 \\ 0 & \text{oth} \end{cases}$

kernel function — parzen window

$k((x - x_n)/h)$ → value is 1 if x_n lies inside hypercube of side h centered on x.

→ total number of data points lying inside this cube

$$K = \sum_{n=1}^{N} k\left(\frac{x - x_n}{h}\right)$$

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^D} k\left(\frac{x - x_n}{h}\right) ; \quad V = h^D$$

volume of hypercube of side h in D dimension

→ If we choose smoother kernel function ⇒ p(x) → smoother

→ consider k(x) → gaussian

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{1/2}} \exp\left(\frac{\|x - x_n\|^2}{2h^2}\right)$$

h → (standard deviation of gaussian component)

h → smoothing parameter
↳ Bandwidth
↳ small h [sensitivity to noise]
↳ large h [over-smoothing]

[h is analogous to choice of bin in histogram]

condition
$k(x) \ge 0$
$\int k(x) dx = 1$

Nearest neighbour methods

problem in KDE: parameter h is fixed for all kernels
In region of high density; large value of h may lead to over-smooth
In region of low density; ↳ Noisy estimate

→ general result of <u>location density estimation</u>

$p(x) = \frac{K}{NV}$

we consider fixed value of K and find appropriate value of V using data.

Process: we consider a small sphere centered on point $\vec{x}$, where we wish to estimate $p(\vec{x})$
- we grow radius of sphere until it contains precisely K data points.
- estimate of density $p(x)$ is given by setting V to volume of resulting sphere.

K Nearest neighbours

⇒ value of K → degree of smoothing
- for optimum K → neither too large nor too small.

⇒ solving: Problem of CLASSIFICATION.

Let us suppose we have data set comprising of (N) points in total and N_K points in each class C_K

$\sum_K N_K = N$

given:

problem: classify new point x
- draw a sphere centered on x containing precisely K points
- suppose this sphere has volume V and contains K_K points from each class C_K

$$p(x|C_K) = \frac{K_K}{N_K V} \;;$$

$$p(x) = \frac{K}{NV} \;;$$

$$p(C_K) = \frac{N_K}{N} \;;$$

using Bayes theorem; $p(C_K|x) = \frac{K_K}{K}$

→ to minimize probability of misclassification ⇒ this is done by assigning the test point x to class having maximum probability.

KNN & KDE
- require entire training set to be stored
- leading to expensive computation.

linear models for Regressions :

Supervised learning

goal: predict value of one or more continuous target var given

D dimensional x of input vector

linear Regression → linear combination of basis functions.

(fixed set of some non-linear fn of input var.)

linear Basis function Models → Minimize expected value of Loss functions

Simplest linear model

$$y(x,w) = w_0 + w_1 x + \cdots w_D x_D$$

$$x = (x_1, x_2 \cdots x_D)^T$$

$$y(x,w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$$ Basis functions

w_0 allows for any fixed offset in data ↳ BIAS parameter

additional dummy basis fn $\phi_0(x) = 1$

$$\boxed{y(x,w) = \sum_{j=0}^{M-1} w_j \phi_j(x)} = w^T \phi(x)$$

$w = (w_0\ w_1 \cdots w_{M-1})^T$ $\phi = (\phi_0\ \phi_1 \cdots \phi_{M-1})^T$

for fixed pre-processing or feature extraction

similarly; polynomial regression: consider single i/p var x ↳ $\phi_j(x) = x^j$

other Basis functions: Gaussian BF $\phi_j(x) = e^{-\frac{(x-\mu_j)^2}{2s^2}}$ $s \rightarrow$ spatial scale, $\mu_j \rightarrow$ location of BF

Sigmoidal BF $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$ $\sigma(a) = \frac{1}{1+e^{-a}}$

Another Basis: Fourier Basis ↳ expansion in sinusoidal fn: ↳ each represent specific frequency and has a spatial extent.

Maximum likelihood & least square

$$t = y(x,w) + \epsilon$$

$y(x,w)$ ← deterministic function; ϵ ← additive Noise

$\epsilon \rightarrow$ 0 mean gaussian RV with precision β

$$p(t|x,w,\beta) = \mathcal{N}(t|y(x,w),\beta^{-1})$$

$$\mathbb{E}[t|x] = \int t\, p(t|x)\, dt = y(x,w)$$

gaussian noise assumption implies that conditional distribution of t given x is unimodal

Consider: $X = \{x_1, x_2, \cdots x_n\}$

$\vec{t} = \{t_1, t_2, \cdots t_n\}$

$$p(\vec{t}|X,w,\beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|w^T\phi(x_n),\beta^{-1})$$

MLF:

$$\ln p(\vec{t}|w,\beta) = \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi - \beta E_D(w)$$

Sum of squared error functions

$$E_D(w) = \frac{1}{2}\sum (t_n - w^T\phi(x_n))^2$$

gradient of MLF

$$\nabla \ln(p(\vec{t}|w,\beta)) = \sum_{n=1}^{N} \{t_n - w^T\phi(x_n)\}\phi(x_n)^T = 0$$

$$0 = \sum_{n=1}^{N} t_n\phi(x_n)^T - w^T\sum_{n=1}^{N}\phi(x_n)\phi(x_n)^T$$

$$w_{MLE} = (\Phi^T\Phi)^{-1}\Phi^T t$$

Normal eqn for least square problems

$\Phi \rightarrow N \times M$ (design matrix)

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_{M-1}(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_{M-1}(x_2) \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_{M-1}(x_N) \end{pmatrix}$$

$$\Phi^\dagger = (\Phi^T\Phi)^{-1}\Phi^T$$ Pseudo Inverse of Matrix Φ

$N \times M$, $M \times N$

$AA^T = A^\dagger$ ($N \times N$)

$$A^\dagger A = I$$

$$A^\dagger A \cdot A^T = A^T$$

$$\boxed{A^\dagger = A^T(A\cdot A^T)^{-1}}$$

$$A \cdot A^\dagger = I$$

$$(A^T A)A^\dagger = A^T$$

$$\boxed{A^\dagger = (A^T A)^{-1}A^T}$$

$A \cdot A^{-1} = I \quad (AB)^{-1} = B^{-1}A^{-1}$

$$E_D(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w_0 - \sum_{j=1}^{M-1} w_j \phi_j(x_n)\}^2$$

$0 = \frac{d}{dw_0}$ ↓

↳ get $w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \bar{\phi}_j$

$$\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n \quad ; \quad \bar{\phi}_j = \frac{1}{N}\sum_{n=1}^{N}\phi_j(x_n)$$

⇒ Bias w_0 compensate for differences in average (over training set) of target values and weighted sum of basis functions.

$\log p(t|w,\beta) \rightarrow \frac{d}{d\beta} \rightarrow$ gives

$$\frac{1}{\beta_{MLE}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - w_{MLE}^T \phi(x_n)\}^2$$

Maximize log likelihood wrt β

→ inversion of noise precision is given by residual variance of target value around regression function.

Geometry of least squares

$t = (t_1, t_2, \ldots t_N)$

$y = y(x_N, w)$

$y = y(x, w)$

$y_i = y(x_i, w)$

↙ Sum of squares.

N^{th} dimensional vector whose n value is ↗

→ Sum of ~~least~~ squares is then equal to squared Euclidean distance between $\underline{y}$ and $\underline{t}$.

the least square solution $\underline{w}$ corresponds to that choice of y that lies in subspace S and that is closest to $\underline{t}$.

Solution corresponds to orthogonal projections of $\underline{t}$ onto subspace $\underline{S}$.

$\underline{y} = \underline{\phi} w_{ML}$ [If ϕ is singular, then adding regularization term is beneficial]

Sequential learning

→ If data set is very large (Sequential algo / on-line algo)

↳ stochastic gradient descent ↳ update parameter vector

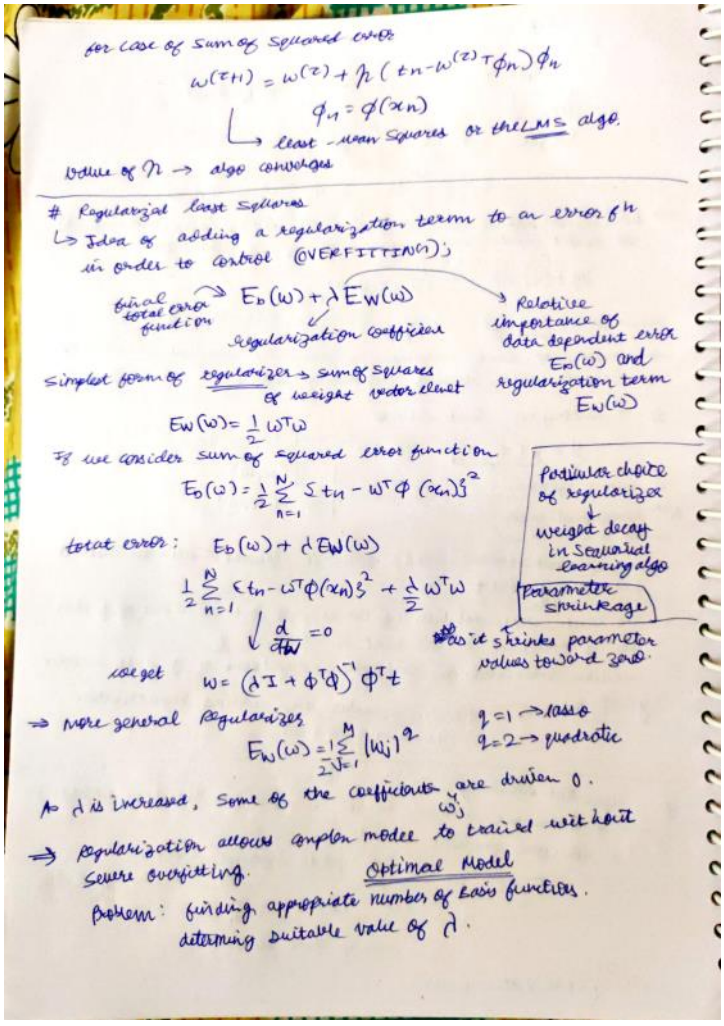
$\tau \rightarrow$ no of iterations

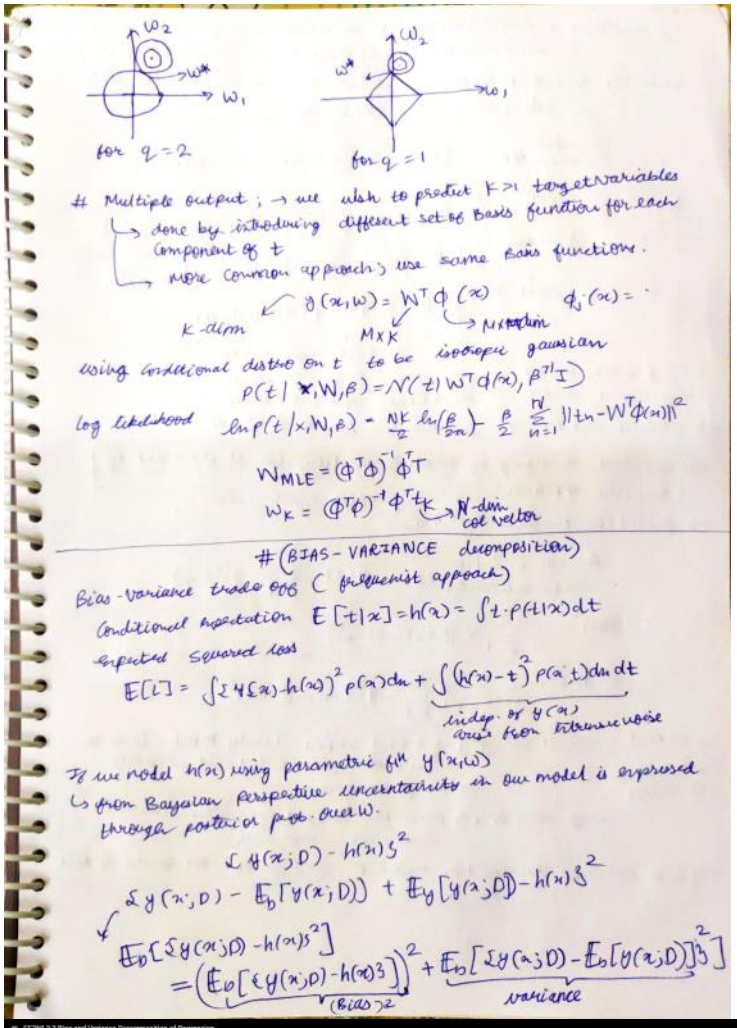
$E = \sum_n E_n$

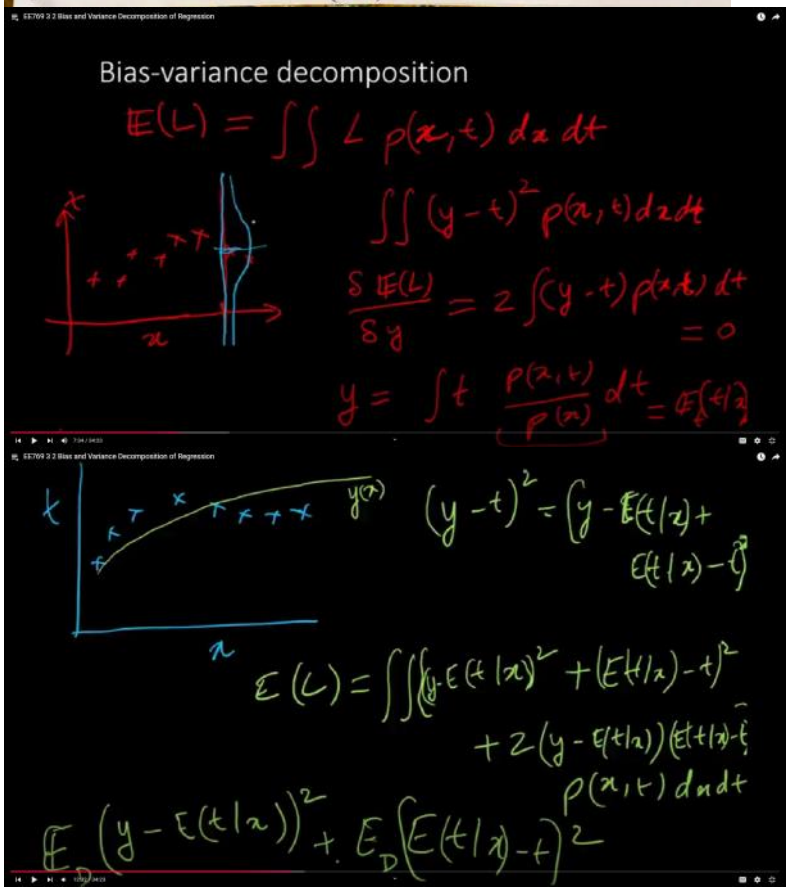
$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n$$

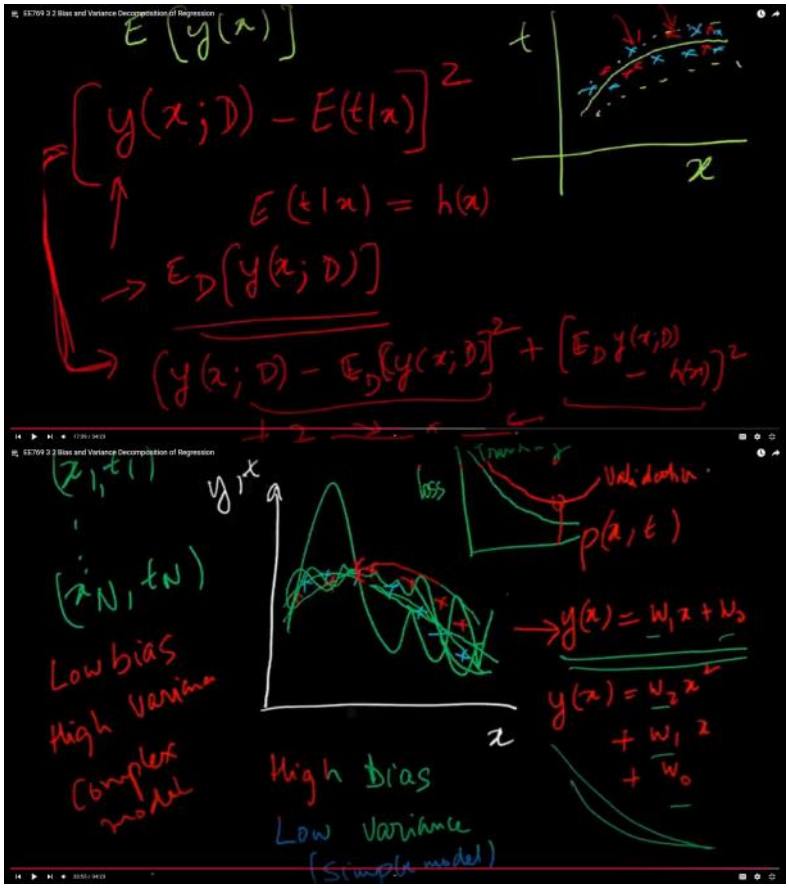
$\eta \rightarrow$ learning rate parameter

$w \rightarrow w^{(0)}$ (some starting vector)


for case of sum of squared error
$w^{(\tau+1)} = w^{(\tau)} + \eta (t_n - w^{(\tau)T}\phi_n)\phi_n$
$\phi_n = \phi(x_n)$
↳ least - mean squares or the LMS algo.
Value of η → algo converges
Regularized least squares
↳ Idea of adding a regularization term to an error fn in order to control OVERFITTING;
final total error function → $E_D(w) + \lambda E_W(w)$
regularization coefficient
Relative importance of data dependent error $E_D(w)$ and regularization term $E_W(w)$
simplest form of regularizer → sum of squares of weight vector elements
$E_W(w) = \frac{1}{2} w^T w$
If we consider sum of squared error function
$E_D(w) = \frac{1}{2}\sum_{n=1}^{N} \{t_n - w^T\phi(x_n)\}^2$
total error: $E_D(w) + \lambda E_W(w)$
$\frac{1}{2}\sum_{n=1}^{N} \{t_n - w^T\phi(x_n)\}^2 + \frac{\lambda}{2} w^T w$
$\downarrow \frac{d}{dw} = 0$
we get $w = (\lambda I + \Phi^T\Phi)^{-1}\Phi^T t$
Particular choice of regularizer
↓
weight decay in sequential learning algo
Parameter shrinkage
as it shrinks parameter values toward zero.
⇒ more general regularizer,
$E_W(w) = \frac{1}{2}\sum_{j=1}^{M} |w_j|^q$
$q=1$ → lasso
$q=2$ → quadratic
As λ is increased, some of the coefficients w_j are driven 0.
⇒ Regularization allows complex models to be trained with out severe overfitting.
Optimal Model
Problem: finding appropriate number of basis functions.
determining suitable value of λ.


for q = 2
for q = 1
Multiple output ; → we wish to predict K > 1 target variables
↳ done by introducing different set of basis function for each component of t
↳ more common approach : use same basis functions.
K-dim
M x K
using conditional distro on t to be isotropic gaussian
log likelihood
$W_{MLE} = (\Phi^T\Phi)^{-1}\Phi^T T$
$w_k = (\Phi^T\Phi)^{-1}\Phi^T t_k$ → N-dim col vector
(BIAS - VARIANCE decomposition)
Bias-variance trade off (frequentist approach)
Conditional expectation $E[t|x] = h(x) = \int t\, p(t|x)\, dt$
Expected squared loss
$E[L] = \int \{y(x) - h(x)\}^2 p(x)\, dx + \int (h(x) - t)^2 p(x,t)\, dx\, dt$
indep. of y(x) arises from intrinsic noise
If we model h(x) using parametric fn y(x,w)
↳ from Bayesian perspective uncertainty in our model is expressed through posterior prob. over w.
$\{y(x;D) - h(x)\}^2$
$\{y(x;D) - E_D[y(x;D)]\} + E_D[y(x;D)] - h(x)\}^2$
$E_D[\{y(x;D) - h(x)\}^2] = \{E_D[y(x;D) - h(x)]\}^2 + E_D[\{y(x;D) - E_D[y(x;D)]\}^2]$
(bias)^2
variance


EE769 3.2 Bias and Variance Decomposition of Regression
Bias-variance decomposition
$E(L) = \int\int L\, p(x,t)\, dx\, dt$
$\int\int (y - t)^2 p(x,t)\, dx\, dt$
$\frac{\delta E(L)}{\delta y} = 2\int (y - t)\, p(x,t)\, dt = 0$
$y = \int t\, \frac{p(x,t)}{p(x)}\, dt = E_t[t|x]$
EE769 3.2 Bias and Variance Decomposition of Regression
y(x)
$(y - t)^2 = \{y - E(t|x) + E(t|x) - t\}^2$
$E(L) = \int\int \{y - E(t|x)\}^2 + (E(t|x) - t)^2 + 2(y - E(t|x))(E(t|x) - t)\, p(x,t)\, dx\, dt$
$E_D\{y - E(t|x)\}^2 + E_D\{E(t|x) - t\}^2$



Squared Bias → represent extent to which avg. prediction over all data set differs from desired regression fn

Variance → extent to which solutions for each individual data sets vary around their avg.

Expected Squared loss = (bias)² + (variance) + (noise)

$$(\text{bias})^2 = \int \{ \mathbb{E}_D[y(x;D)] - h(x) \}^2 p(x)\, dx$$

$$(\text{Variance}) = \int \mathbb{E}_D \left[\{ y(x;D) - \mathbb{E}[y(x;D)] \}^2 \right] p(x)\, dx$$

$$\text{noise} = \underbrace{\int \{ h(x) - t \}^2 p(x,t)\, dx\, dt}_{\text{constant noise terms}}$$

Very flexible model → low bias, high variance
Very rigid model → high bias, low variance
⭐ optimal model → Balance b/w bias & variance
⇒ [weighted averaging of multiple solutions lies at the heart of Bayesian approach]
→ prediction function $y^{(\omega)}(x)$ after regularization

Average prediction can be estimated by $\bar{y}(x) = \frac{1}{L} \sum_{l=1}^{L} y^{(l)}(x)$

$$(\text{bias})^2 = \frac{1}{N} \sum_{n=1}^{N} \{ \bar{y}(x_n) - h(x_n) \}^2$$

$$\text{variance} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{L} \sum_{l=1}^{L} \{ y^{(l)}(x_n) - \bar{y}(x_n) \}^2$$

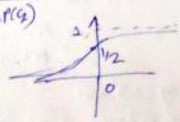
⇒ In book; small value of (λ) → model become finely tuned to noise on each individual data set, leading to large variance
Regularization parameter
large value of λ → pulls the weight parameters toward 0. leading to large bias.

⇒ Bias-Variance decomposition based on averages of test sets of data

Linear Model for Classification

→ goal: we Model: class conditional density $p(x|C_k)$ and class prob. $p(C_k)$

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)}$$



⇐ shape (Logistic sigmoid function $= \frac{1}{1+e^{-a}} = \sigma(a)$)

Symmetry property $\sigma(-a) = 1 - \sigma(a)$

$a = \ln\left(\frac{p(x|C_1)\cdot p(C_1)}{p(x|C_2)\cdot p(C_2)}\right)$ Logit fn log odds.

⇒ generalised linear model

$$p(C_k|x) = \frac{p(x|C_k)\cdot p(C_k)}{\sum_j p(x|C_j)p(C_j)}$$

Normalized exponential multiclass generalization of (logistic sigmoid) $= \frac{e^{a_k}}{\sum_j e^{a_j}}$

where $a_k = \ln(p(x|C_k)\cdot p(C_k))$

⇒ Also known as softmax function [=smooth versions max function] if $a_k \gg a_j$; then $p(C_k|x) = 1$ and $p(C_j|x) \simeq 0$.

let assume class conditional densities are gaussian assumption

$$p(x|C_k) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(\frac{-1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right)$$

consider case of 2 classes:

$$p(C_1|x) = \sigma(w^Tx + w_0)$$

$$w = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma^{-1}\mu_2 + \ln\left(\frac{p(C_1)}{p(C_2)}\right)$$

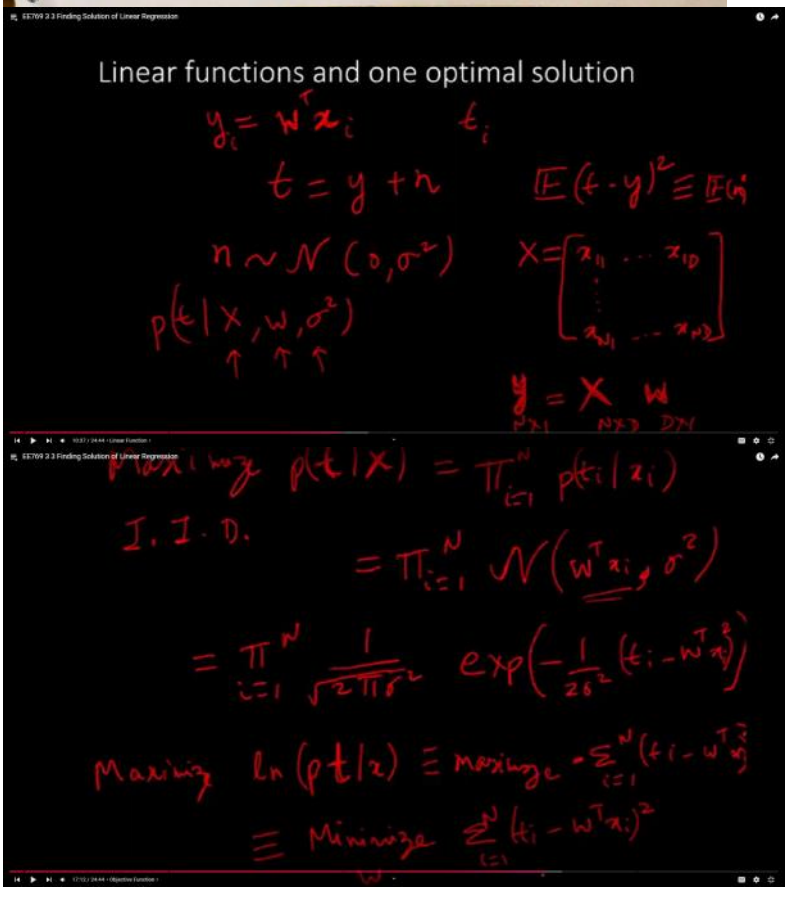
for general case of K classes:

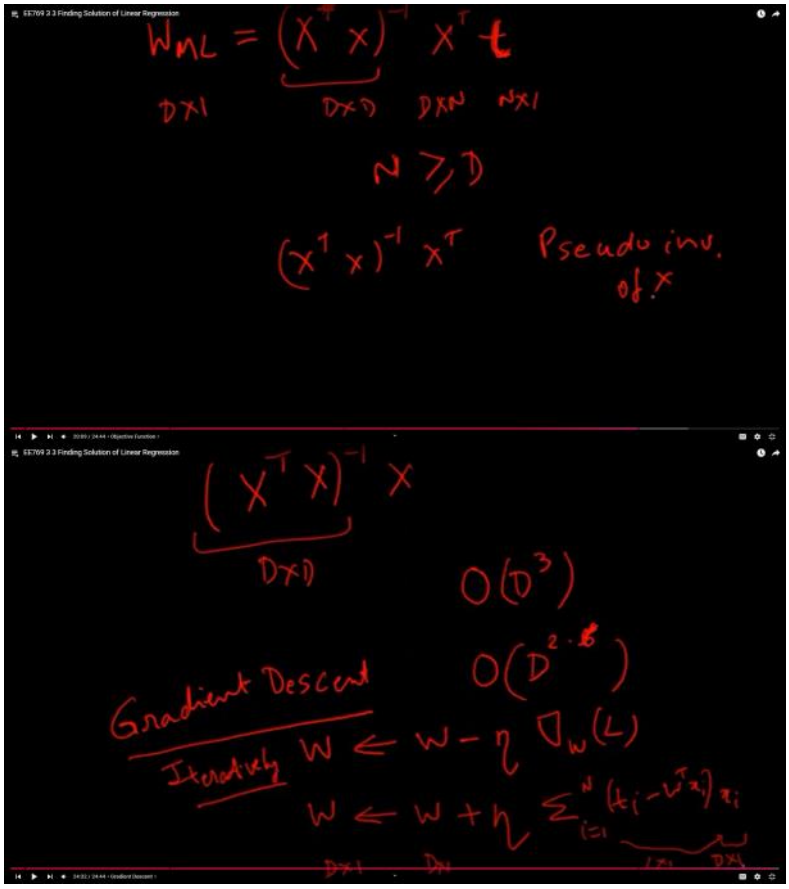
$$a_k(x) = w_k^T x + w_{k0}$$

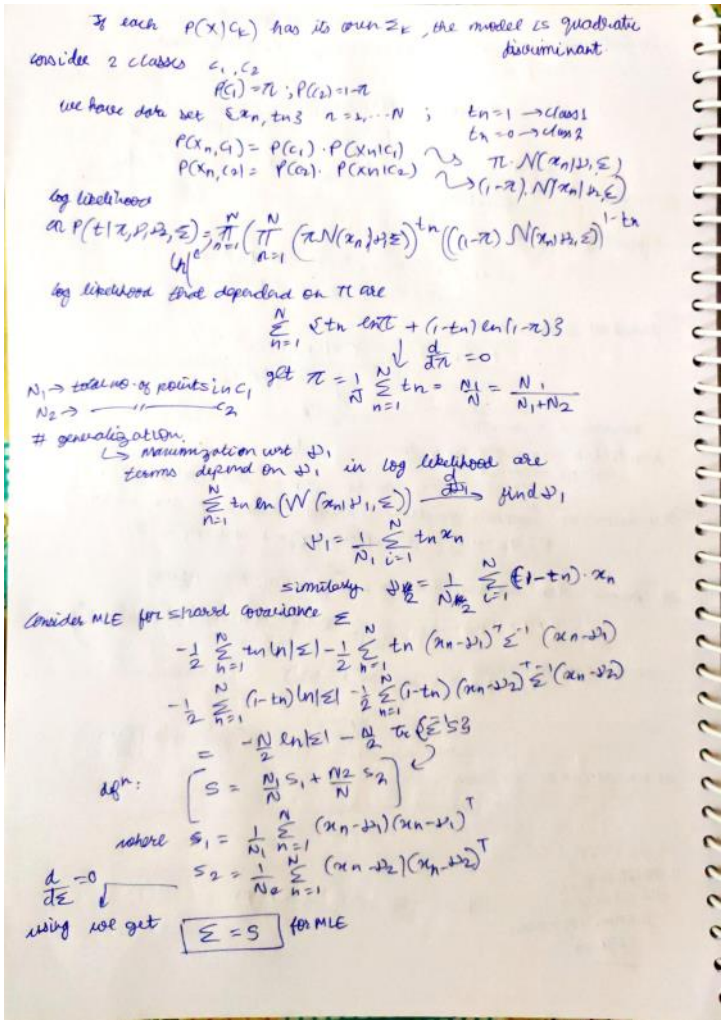
$$w_k = \Sigma^{-1}\mu_k$$

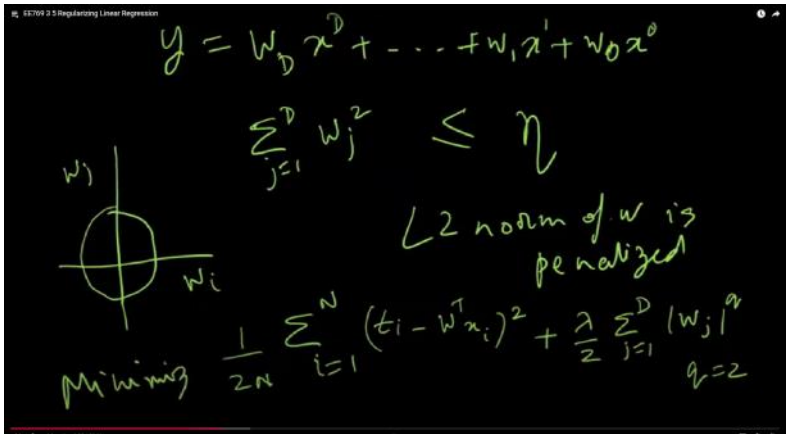
$$w_{k0} = -\frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k^T + \ln(p(C_k))$$

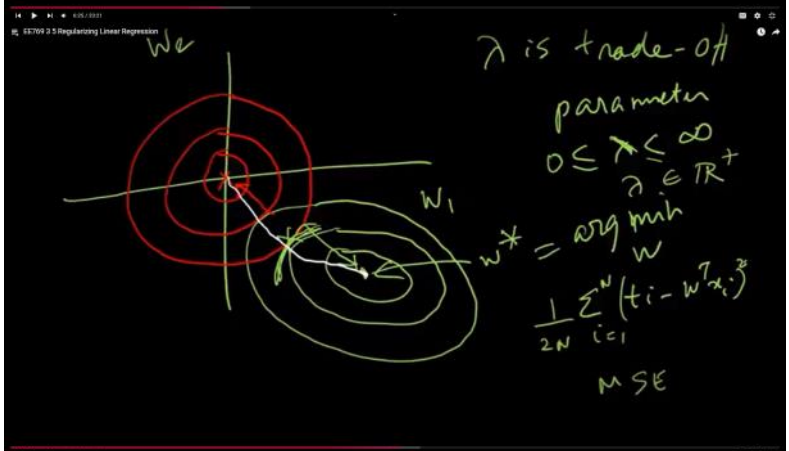
assumption that all shared a common covariance Matrix

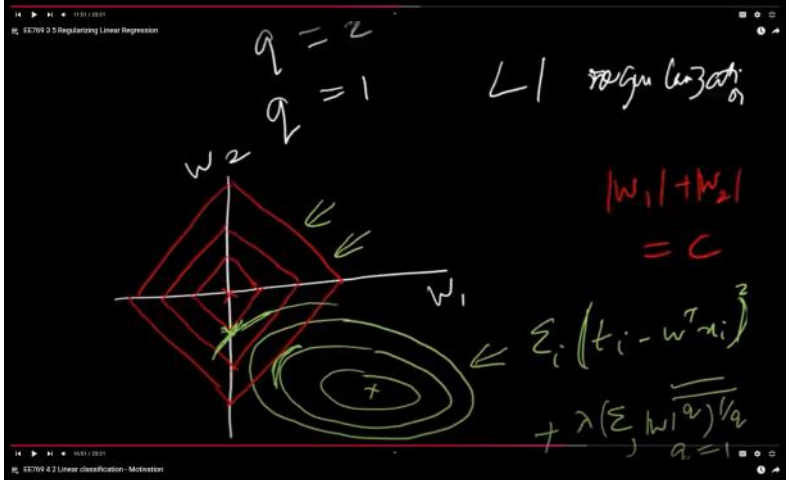


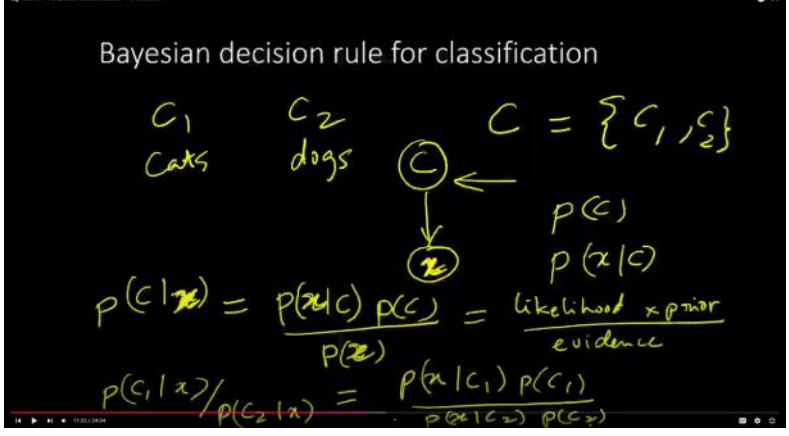

EE769 2.3 Finding Solution of Linear Regression
$W_{ML} = (X^T X)^{-1} X^T t$
D×1 D×D D×N N×1
$N \geq D$
$(X^T X)^{-1} X^T$ Pseudo inv. of X
EE769 2.3 Finding Solution of Linear Regression
$(X^T X)^{-1} X$
D×D
$O(D^3)$
$O(D^{2 \cdot 6})$
Gradient Descent
Iteratively $W \leftarrow W - \eta \nabla_w(L)$
$W \leftarrow W + \eta \sum_{i=1}^{N} (t_i - W^T x_i) x_i$
D×1 D×N

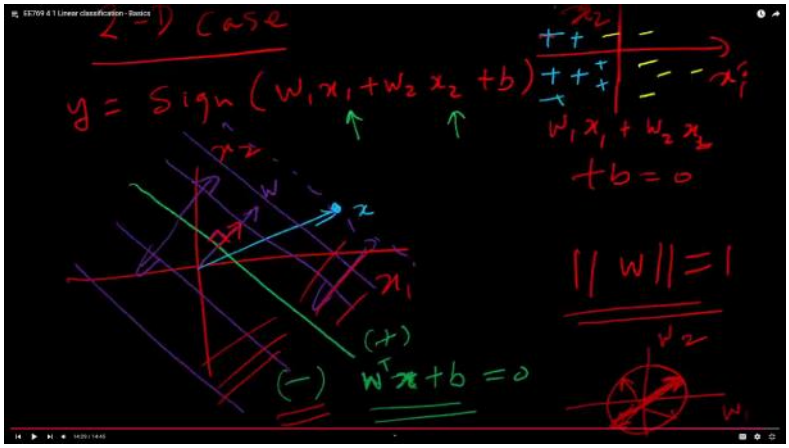

If each $P(x|C_k)$ has its own Σ_k, the model is quadratic discriminant
consider 2 classes C_1, C_2
$P(C_1) = \pi$; $P(C_2) = 1-\pi$
we have data set $\{x_n, t_n\}$ $n = 1, \cdots N$; $t_n = 1 \rightarrow$ class 1, $t_n = 0 \rightarrow$ class 2
$P(x_n, C_1) = P(C_1) \cdot P(x_n|C_1) \rightsquigarrow \pi \cdot N(x_n|\mu_1, \Sigma)$
$P(x_n, C_2) = P(C_2) \cdot P(x_n|C_2) \rightsquigarrow (1-\pi) \cdot N(x_n|\mu_2, \Sigma)$
log likelihood
$P(t|\pi, \mu_1, \mu_2, \Sigma) = \prod_{n=1}^{N} \left(\pi N(x_n|\mu_1, \Sigma)\right)^{t_n} \left((1-\pi) N(x_n|\mu_2, \Sigma)\right)^{1-t_n}$
log likelihood that depended on π are
$\sum_{n=1}^{N} \{t_n \ln \pi + (1-t_n) \ln(1-\pi)\}$
$N_1 \rightarrow$ total no. of points in C_1
$N_2 \rightarrow$ " " C_2
get $\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$
generalization
↳ maximization wrt μ_1
terms depend on μ_1 in log likelihood are
$\sum_{n=1}^{N} t_n \ln N(x_n|\mu_1, \Sigma)$ → find μ_1
$\mu_1 = \frac{1}{N_1} \sum_{i=1}^{N} t_n x_n$
similarly $\mu_2 = \frac{1}{N_2} \sum_{i=1}^{N} (1-t_n) \cdot x_n$
Consider MLE for shared covariance Σ
$-\frac{1}{2} \sum_{n=1}^{N} t_n \ln|\Sigma| - \frac{1}{2} \sum_{n=1}^{N} t_n (x_n - \mu_1)^T \Sigma^{-1} (x_n - \mu_1)$
$-\frac{1}{2} \sum_{n=1}^{N} (1-t_n) \ln|\Sigma| - \frac{1}{2} \sum_{n=1}^{N} (1-t_n) (x_n - \mu_2)^T \Sigma^{-1} (x_n - \mu_2)$
$= -\frac{N}{2} \ln|\Sigma| - \frac{N}{2} \mathrm{Tr}\{\Sigma^{-1} S\}$
defn: $S = \frac{N_1}{N} S_1 + \frac{N_2}{N} S_2$
where $S_1 = \frac{1}{N_1} \sum_{n=1}^{N} (x_n - \mu_1)(x_n - \mu_1)^T$
$S_2 = \frac{1}{N_2} \sum_{n=1}^{N} (x_n - \mu_2)(x_n - \mu_2)^T$
$\frac{d}{d\Sigma} = 0$
using we get $\Sigma = S$ for MLE


EE769 3 5 Regularizing Linear Regression
$y = w_D x^D + \dots + w_1 x^1 + w_0 x^0$
$\sum_{j=1}^{D} w_j^2 \leq \eta$
w_j
w_i
L2 norm of w is penalized
Minimizing $\frac{1}{2N} \sum_{i=1}^{N} (t_i - w^T x_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{D} |w_j|^q$
$q=2$

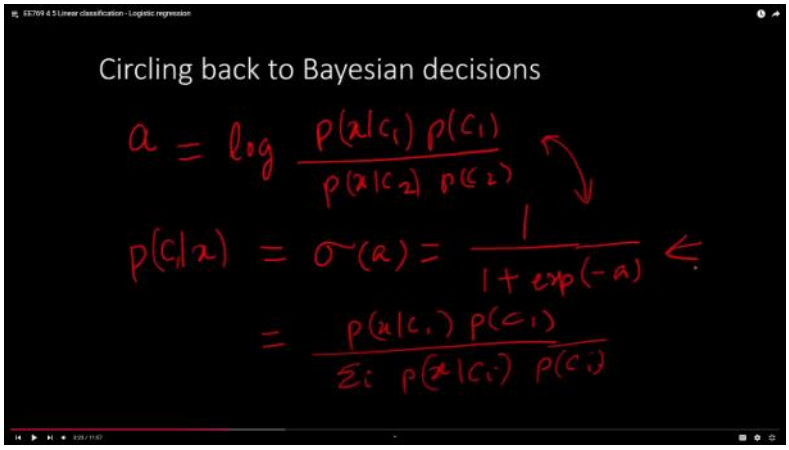

EE769 3 5 Regularizing Linear Regression
We
w_1
w^*
λ is trade-off parameter
$0 \leq \lambda \leq \infty$
$\lambda \in \mathbb{R}^+$
$w^* = \arg\min_w \frac{1}{2N} \sum_{i=1}^{N} (t_i - w^T x_i)^2$
MSE

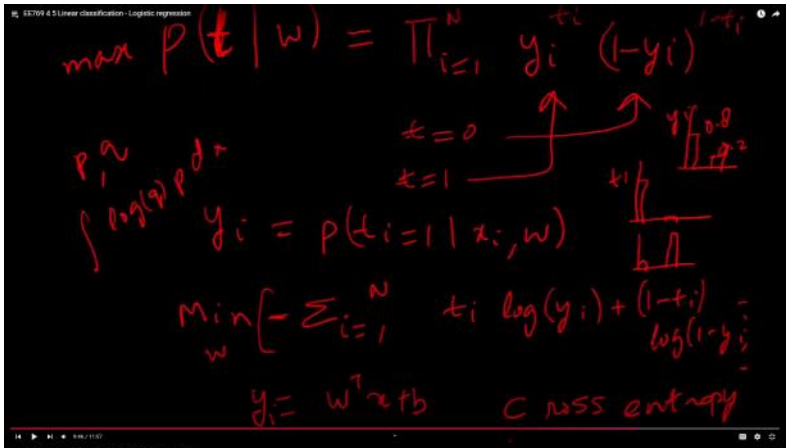

EE769 3 5 Regularizing Linear Regression
$q=2$
$q=1$
L1 regularization
w_2
w_1
$|w_1| + |w_2| = c$
$\sum_i (t_i - w^T x_i)^2$
$+ \lambda (\sum_j |w_j|^q)^{1/q}$
$q=1$

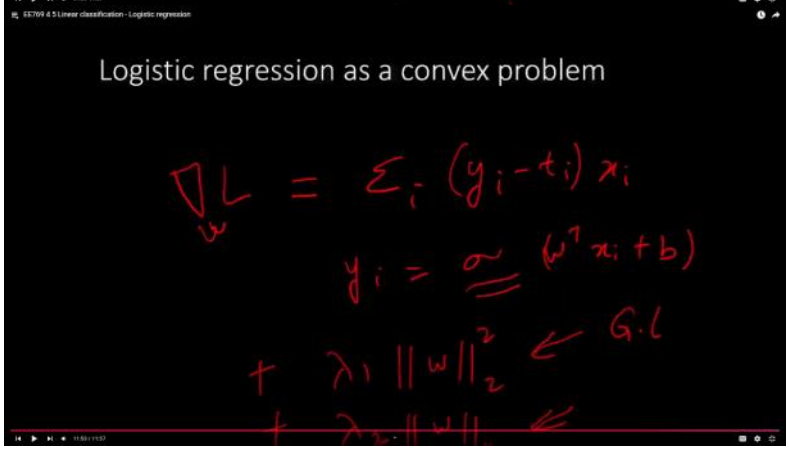

EE769 4 2 Linear classification - Motivation
Bayesian decision rule for classification
C_1
Cats
C_2
dogs
$C = \{C_1, C_2\}$
C
x_c
$p(C)$
$p(x|C)$
$p(C|x_c) = \frac{p(x_c|C)\, p(C)}{p(x_c)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$
$p(C_1|x) / p(C_2|x) = \frac{p(x|C_1)\, p(C_1)}{p(x|C_2)\, p(C_2)}$



LOGISTIC REGRESSION







Decision Rule:

Class x as C_k if $P(C_k|x) > P(C_j|x)$ for $\forall j \neq k$

$P(x|C_k) \cdot P(C_k) > P(x|C_j) \cdot P(C_j) \quad \forall j \neq k$

Intuition:

Likelihood $P(x|C_k)$ reflects how probable the feature vector x is given class C_k.

Prior $P(C_k)$ how probable is class C_k given no data

Posterior $P(C_k|x)$ update probability of class C_k after observing x.

→ for binary classification

class is 1 if $P(t=1|x) > P(t=0|x)$

$P(x|t=1) \cdot P(t=1) > P(x|t=0) \cdot P(t=0)$

Gaussian Assumption with same class cov matrix (same)

$$P(x|C_k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(\frac{-1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)\right)$$

$d \rightarrow$ dim of x

$\mu_k \rightarrow$ mean vector of class C_k

$\Sigma \rightarrow$ cov matrix

↳ Decision Rule $P(C_k|x) \propto P(C_k|x) \cdot P(C_k)$

(class conditional gaussian dist)

$P(x|t=k) \equiv P(x|C_k) = N(\mu_j, \Sigma_j)$

Decision Boundary is linear when all have same cov matrix ($\Sigma_k = \Sigma$)

$P(x|C_k) = N(\mu_k, \Sigma_k)$

$P(x|C_j) = N(\mu_j, \Sigma_j)$ Assume $\Sigma_k = \Sigma_j = \Sigma$ given

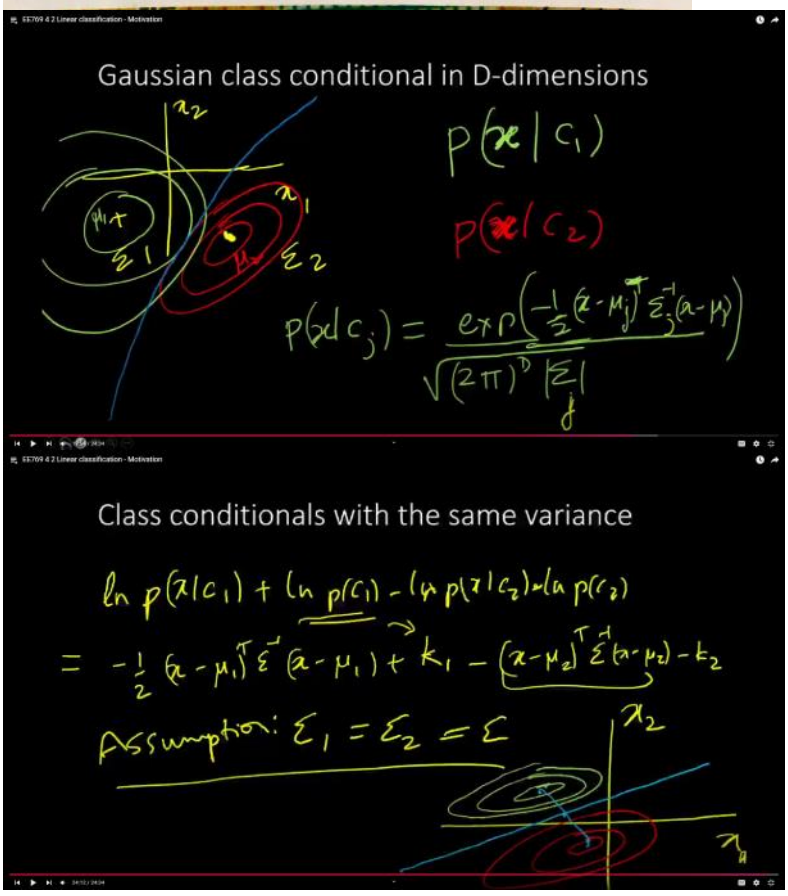
$P(x|C_k) \cdot P(C_k) > P(x|C_j) P(C_j)$

→ want Boundary Put equality and get condition

$N(\mu_k, \Sigma) P(C_k) = N(\mu_j, \Sigma) \cdot P(C_j)$

↳ After solving we get Boundary as $\log\left(\frac{P(C_j)}{P(C_k)}\right)$

($\log[P(C_k|x)]$ / $\log[P(C_j|x)]$ is linear in x only if $\Sigma_j = \Sigma_k$)



decision boundary proof under gaussian assumption (only two classes)

$P(t=1|x) > P(t=0|x)$

$P(x|t=1) \cdot P(t=1) > P(x|t=0) \cdot P(t=0)$

$P(t=1) \equiv P(C_1)$

$P(t=0) \equiv P(C_0)$

$$\frac{P(x|t=1)}{P(x|t=0)} > \frac{P(t=0)}{P(t=1)}$$

$$\log\left(\frac{\exp\left(-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right)}{\exp\left(-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right)} > \frac{P(t=0)}{P(t=1)}\right)$$

$$-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0) > \log\left(\frac{P(t=0)}{P(t=1)}\right)$$

→ after grouping

$$\underbrace{2(\mu_1-\mu_0)^T\Sigma^{-1}}_{w^T}x + \underbrace{\mu_1^T\Sigma^{-1}\mu_0 - \mu_1^T\Sigma^{-1}\mu_1 + 2\log\left(\frac{P(t=1)}{P(t=0)}\right)}_{b} > 0$$

$w^Tx + b > 0$

$w^Tx + b = 0$ → Boundary condition

Bayesian Classifier Algorithm

① calculate prior probabilities

$P(t=1) = \frac{n_1}{N}$ $\quad [n_1 + n_0 = N]$

$P(t=0) = \frac{n_0}{N}$ ↓ sample where class $i=1$

② estimate conditional prob.

$\mu_0 = \frac{1}{|X_0|}\sum_{x \in X_0} x$ $\quad \mu_1 = \frac{1}{|X_1|}\sum_{x \in X_1} x$

X_0 → set of x where class is 0

X_1 similarly

$X = X_0 \cup X_1$

(if $x \in X_0$ then $y = \mu_0$; if $x \in X_1$ then $y = \mu_1$)

③ calculate shared covariance matrix

$$\Sigma = \frac{1}{n}\sum_{x \in X}(x-\mu_x)(x-\mu_x)$$

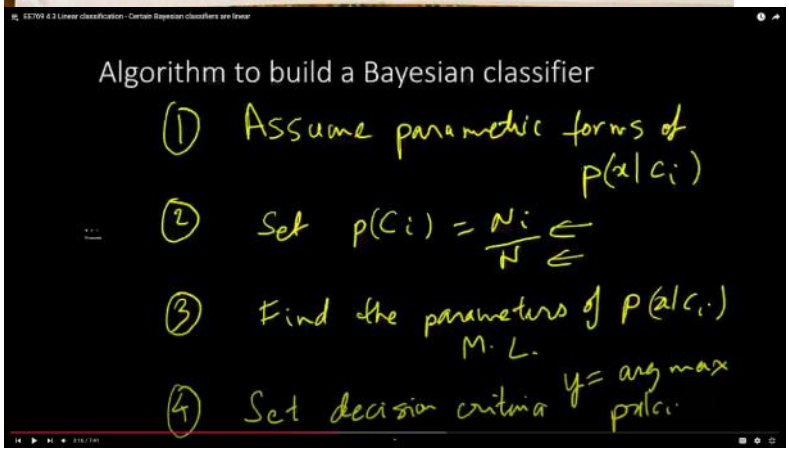
$P(x|t=0) = N(\mu_0, \Sigma)$

$P(x|t=1) = N(\mu_1, \Sigma)$

④ Apply decision Rule

$P(x|t=0) \cdot P(t=0) > P(x|t=1) \cdot P(t=1)$

for class 0.



Gradient descent for linear classifier

Hypothesis ; $h_i = \log\left(\frac{P(t_i=1|x_i)}{P(t_i=0|x_i)}\right) = w^T x_i + b$

(log of odds ratio)

Probability ; $y_i = P(t_i=1|x_i) = \sigma(h_i) = \frac{1}{1+e^{-h_i}}$

BCE Loss ; (Binary Cross Entropy) $L_i = -t_i \log y_i - (1-t_i)\log(1-y_i)$

→ find gradient of loss wrt w_k $\frac{\partial L_i}{\partial w_k}$

$$\frac{\partial L}{\partial y_i} = -\frac{t_i}{y_i} + \frac{1-t_i}{1-y_i}$$

$$\frac{\partial y_i}{\partial h_i} = y_i(1-y_i)$$

$$\frac{\partial h_i}{\partial w_k} = x_{ik}$$

$$\frac{\partial L}{\partial w_k} = \frac{\partial L}{\partial y_i} \cdot \frac{\partial y_i}{\partial h_i} \cdot \frac{\partial h_i}{\partial w_k} = \left(-\frac{t_i}{y_i} - \frac{(1-t_i)}{(1-y_i)}\right) y_i(1-y_i) x_{ik}$$

$$\boxed{\frac{\partial L}{\partial w_k} = (y_i - t_i) \cdot x_{ik}}$$

→ gradient descent update rule

$$w_{new} = w_{old} - \eta \sum_i (y_i - t_i) x_i$$

(η → learning rate)

Adding Regularization

L1 (Lasso) $L_{L1} = L + \lambda \sum_k |w_k|$

$$\frac{\partial L_{L1}}{\partial w_k} = \frac{\partial L}{\partial w_k} + \lambda \cdot \text{sign}(w_k)$$

$$w_{new} = w_{old} - \eta \left(\sum_i (y_i - t_i) x_{ik} + \lambda \text{ sign}(w_k)_{old} \right)$$

L2 (Ridge) $L_{L2} = L + \frac{\lambda}{2} \sum_k w_k^2$

$$\frac{\partial L_{L2}}{\partial w_k} = \frac{\partial L}{\partial w_k} + \lambda w_k$$

$$w_{new} = w_{old} - \eta \left(\sum_i (y_i - t_i) x_{ik} + \lambda w_{k_{old}} \right)$$

Elastic Net [$L_1 + L_2$]

$L_{EN} = L + \lambda_1 \sum_k |w_k| + \lambda_2 \sum_k w_k^2$

$w_{new} = w_{old} - \eta \left(\sum_i (y_i - t_i) x_i + \lambda_1 \text{sign}(w_{old}) + 2\lambda_2 w_{old} \right)$

L_1 → eliminate var.
↳ doesn't encourage two correlated var to have same weight

L_2 → grouping effect on correlated var
↳ encourage two correlated var to have same weight

Risk are not symmetric like FP VS FN
↳ goal: minimize expected risk

① Accuracy (A) = $\frac{TP + TN}{TP + TN + FP + FN}$

② Precision (P) = $\frac{TP}{TP + FP}$

③ Recall (R) (Sensitivity) = $\frac{TP}{TP + FN}$

④ Specificity (S) = $\frac{TN}{TN + FP}$

⑤ F_1 score = $2 \cdot \frac{PR}{P+R}$

⑤ ROC curve: → plot of True Positive Rate (Recall) against False positive Rate (Specificity)

AUC → Area under ROC] higher AUC → better performing model)
↳ $\in [0, 1]$
→ By varying decision threshold

Lagrange Multiplier

Problem: $f(x_1, x_2)$ Maximization subject to $g(x_1, x_2) = 0$

$\nabla g(x) = \frac{\partial g}{\partial x}$

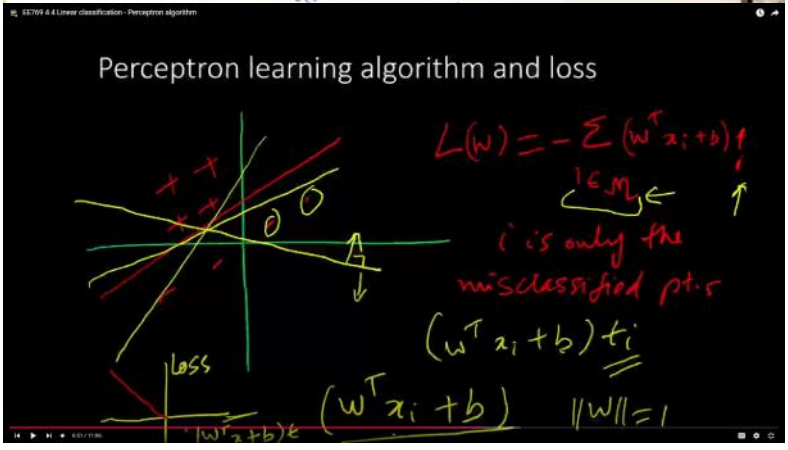
$g(x + \epsilon) \approx g(x) + \nabla g(x) \cdot \epsilon^T$

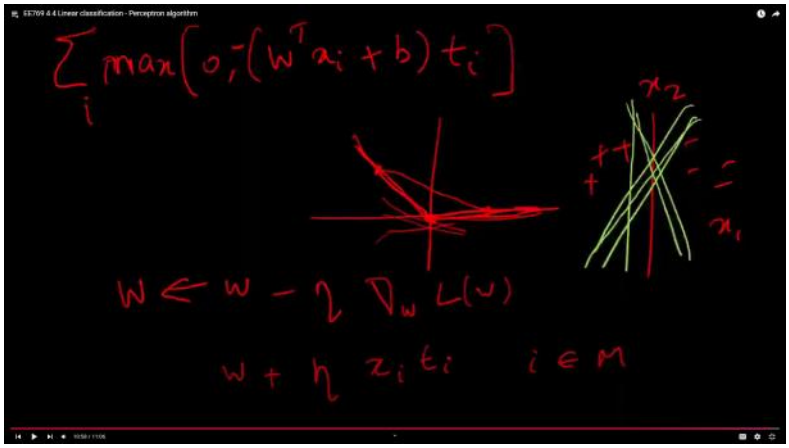
$\nabla f + \lambda \nabla g = 0$
↳ Lagrange Multiplier

$L(x, \lambda) = f(x) + \lambda g(x)$

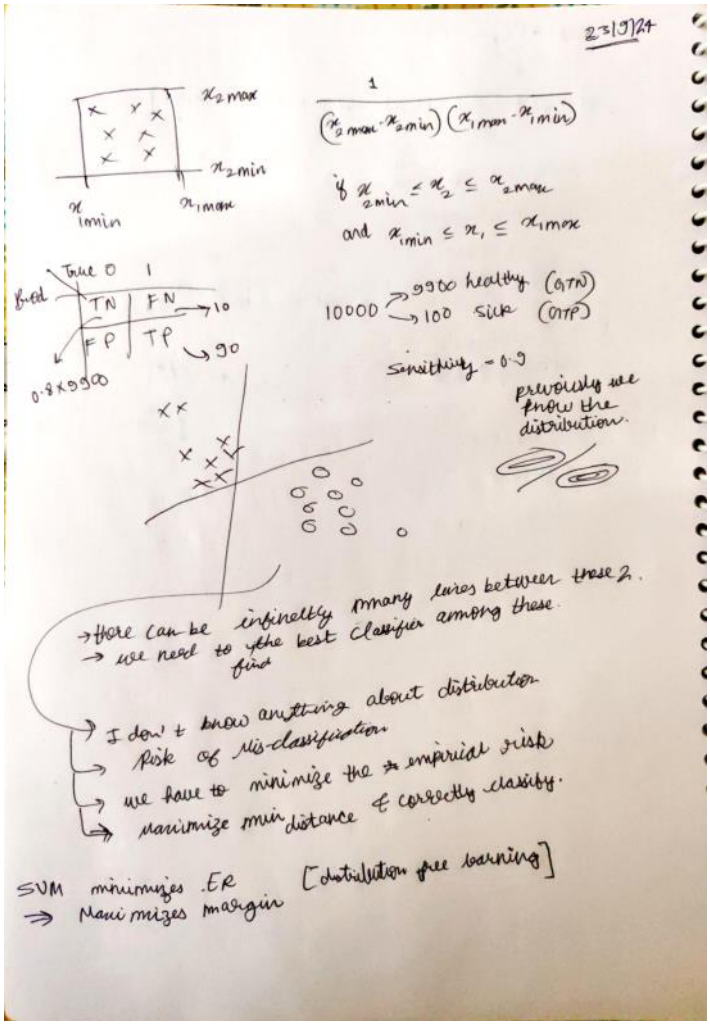
Constrained stationarity condition $\nabla_x L = 0$

$\frac{\partial L}{\partial \lambda} = 0 \rightarrow g(x) = 0$

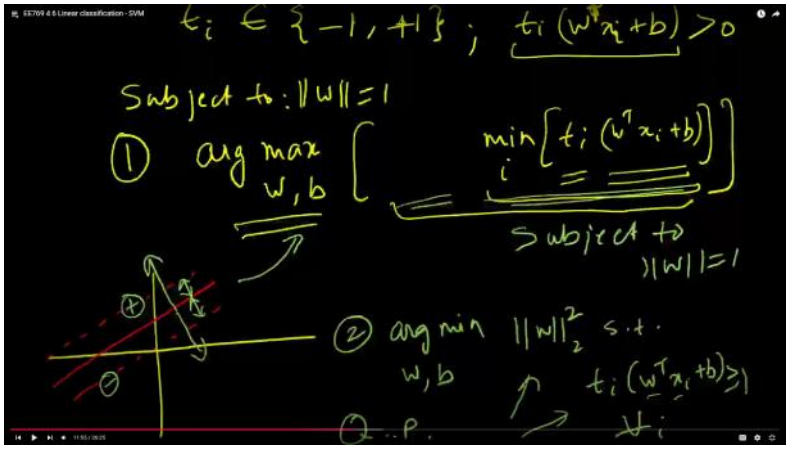


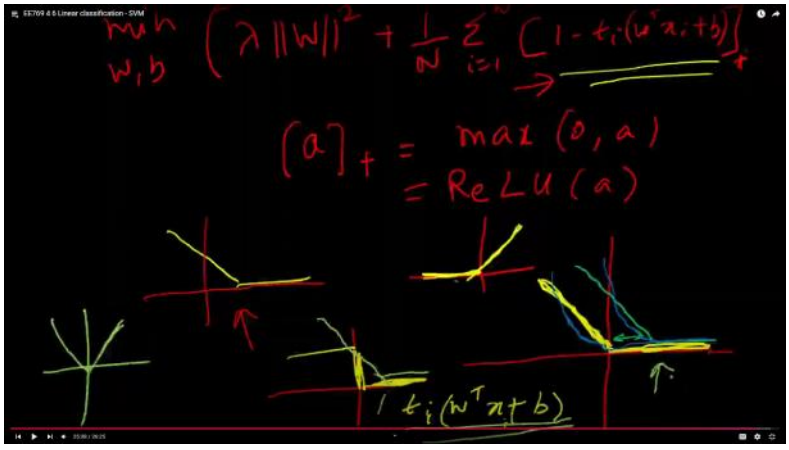


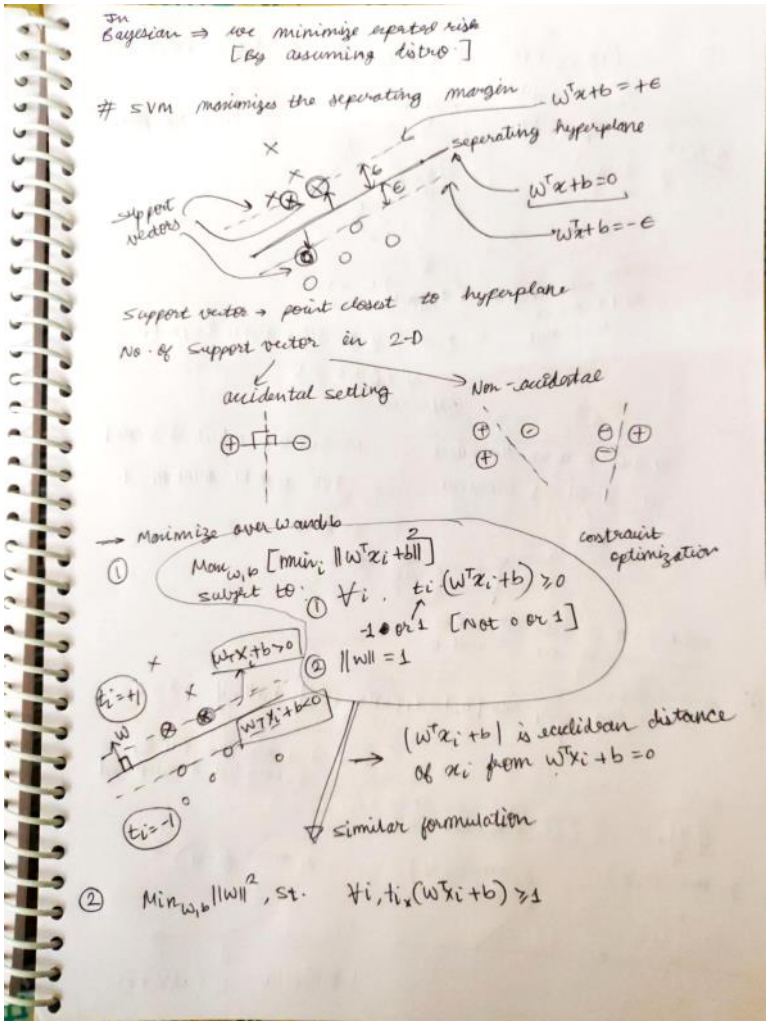
AFTER MIDSEM

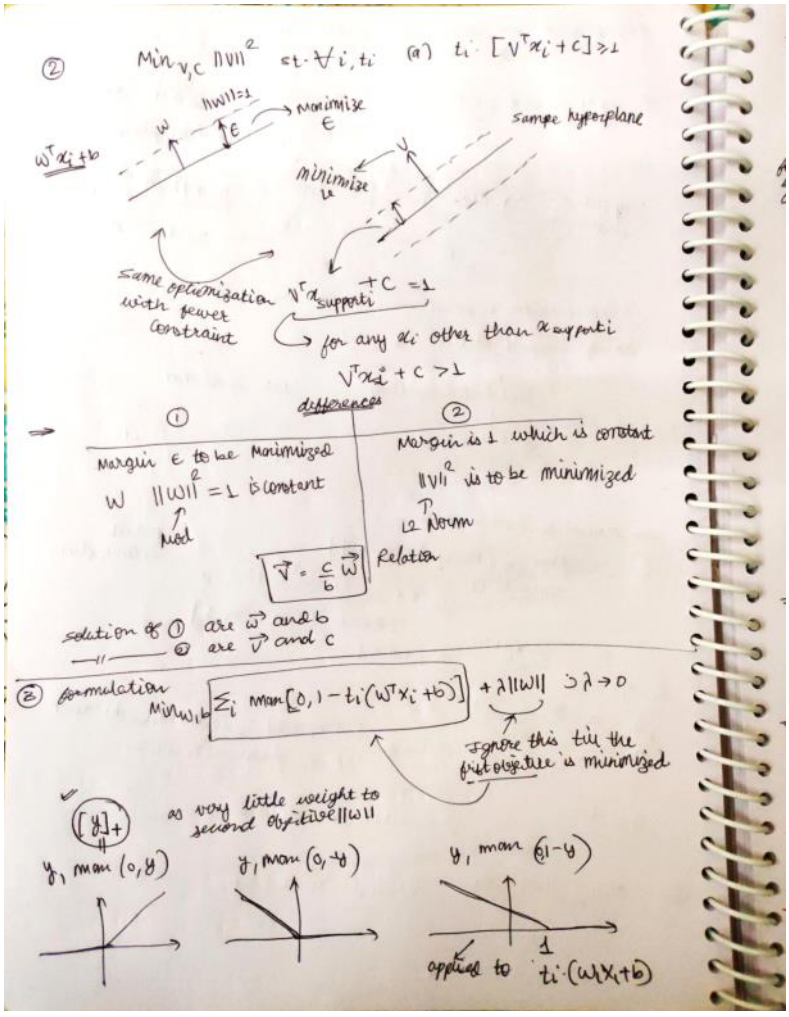


SVM- EMPIRICAL RISK MINIMIZATION
MARGIN MAXIMIZATION




EE769 4.6 Linear classification - SVM
$\min_{w,b} \left(\lambda \|w\|^2 + \frac{1}{N} \sum_{i=1}^{N} [1 - t_i(w^T x_i + b)]_+ \right)$
$[a]_+ = \max(0, a)$
$= \text{ReLU}(a)$
$1 - t_i(w^T x_i + b)$


In Bayesian ⇒ we minimize expected risk
[By assuming distro]
SVM maximizes the separating margin
$w^T x + b = +\epsilon$
separating hyperplane
$w^T x + b = 0$
$w^T x + b = -\epsilon$
support vectors
Support vector → point closest to hyperplane
No. of support vector in 2-D
accidental setting
Non-accidental
Maximize over w and b
constraint optimization
① $\max_{w,b} [\min_i \|w^T x_i + b\|^2]$
subject to: ① $\forall i,\ t_i (w^T x_i + b) \geq 0$
-1 or 1 [not 0 or 1]
② $\|w\| = 1$
$w^T x_i + b > 0$
$t_i = +1$
$w^T x_i + b < 0$
$t_i = -1$
$|w^T x_i + b|$ is euclidean distance of x_i from $w^T x_i + b = 0$
similar formulation
② $\min_{w,b} \|w\|^2$, s.t. $\forall i, t_i (w^T x_i + b) \geq 1$


② $\min_{v,c} \|v\|^2$ s.t. $\forall i, t_i$ (a) $t_i \cdot [v^T x_i + c] \geq 1$
$\|w\|=1$
Maximize ϵ
$w^T x_i + b$
Same hyperplane
Minimize
Same optimization with fewer constraint
$v^T x_{support_i} + c = 1$
for any x_i other than $x_{support_i}$
$v^T x_i + c > 1$
differences
①
Margin ϵ to be Maximized
w $\|w\|^2 = 1$ is constant
Mod
②
Margin is 1 which is constant
$\|v\|^2$ is to be minimized
L2 Norm
$\vec{v} = \frac{c}{b}\vec{w}$ Relation
solution of ① are $\vec{w}$ and b
② are $\vec{v}$ and c
② formulation $\min_{w,b}$ $\sum_i \max[0, 1 - t_i(w^T x_i + b)]$ $+ \lambda\|w\|$ $\lambda \to 0$
Ignore this till the first objective is minimized
$[y]_+$
as very little weight to second objective $\|w\|$
$y, \max(0, y)$
$y, \max(0, y)$
$y, \max(0, 1-y)$
1
applied to $t_i \cdot (w_i x_i + b)$

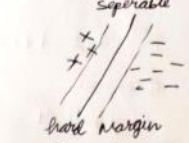
SVM Maximize separating margin 26/8/24

① $\max_{w,b} \left[\min_i \|w^T x_i + b\| \right]$ sub to

(a) $\|w\| = 1$

(b) $\forall i, t_i (w^T x_i + b) \geq 0$

assume linearly seperable



hard margin

Max $[0, y] \equiv [y]_+$

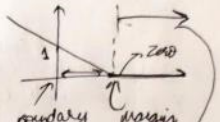
max $[0, 1-y] \equiv l_{hinge}[y]$

support → decesing the decesion boundary

~~sppt~~
→ decision boundary only (entirely) defined by only support vectors

(But like in other classifier)

→ adding new point can shift the decision boundary.



After Optimization all points will be Here (Hard SVM)

Soft Margin

No scope for regularization in Hard SVM.

Hard SVM → want to maximize Hard margin

condition → $y_i = t_i (w^T x_i + b)$ (linearly seperably)

Soft SVM

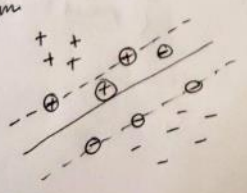
⇒ we want to minimize $\|w\|^2 + C\sum_i \xi_{ti}$ ← slack variable (ξ_i)

s.t $\forall i \quad t_i (w^T x_i + b) \geq 1 - \xi_{ti}$ $\forall \xi_{ti} \geq 0$

⇒ for most points we want slack to be zero.

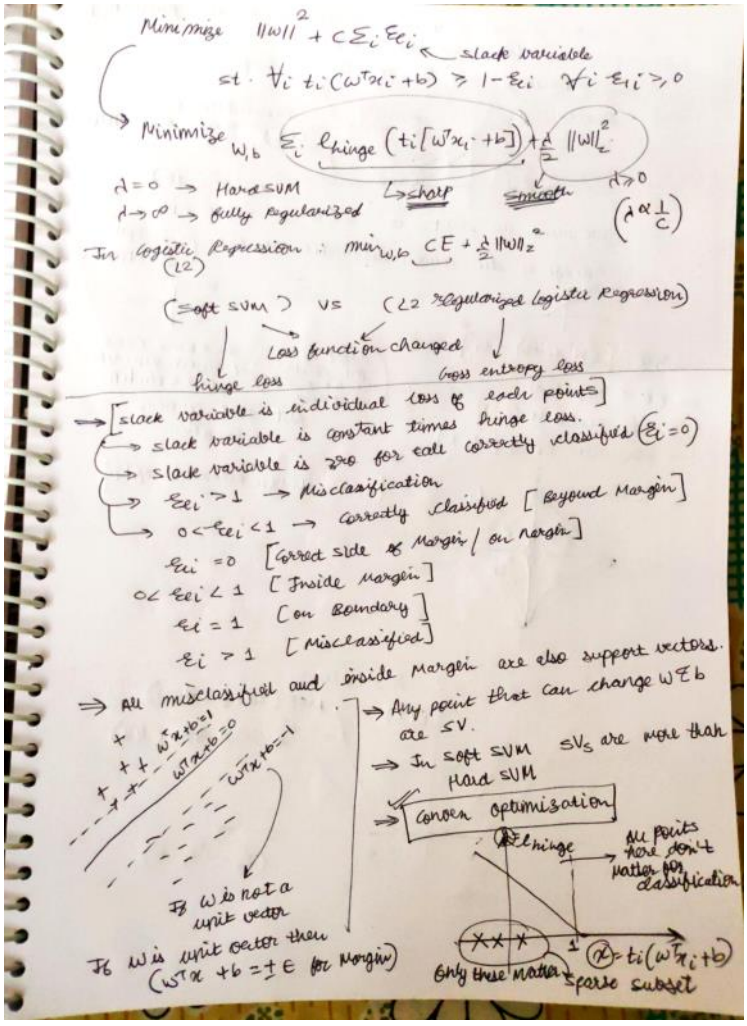
⇒ but for some point slack is greater than 0.

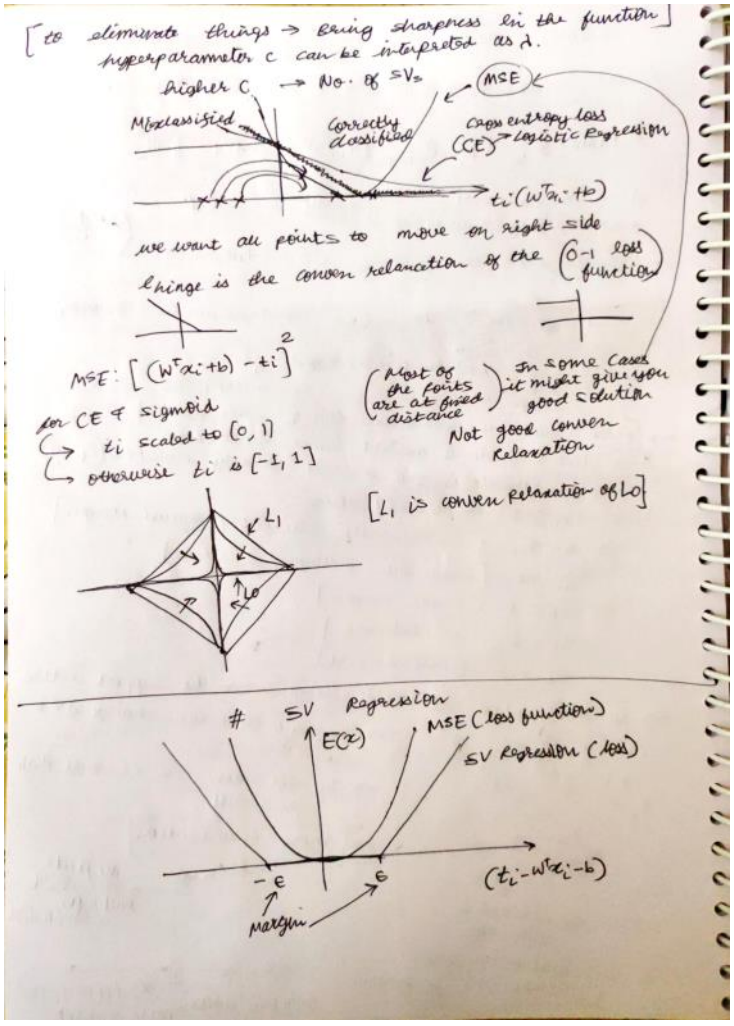
↳ ⇒ some can include in the margin or outside the margin

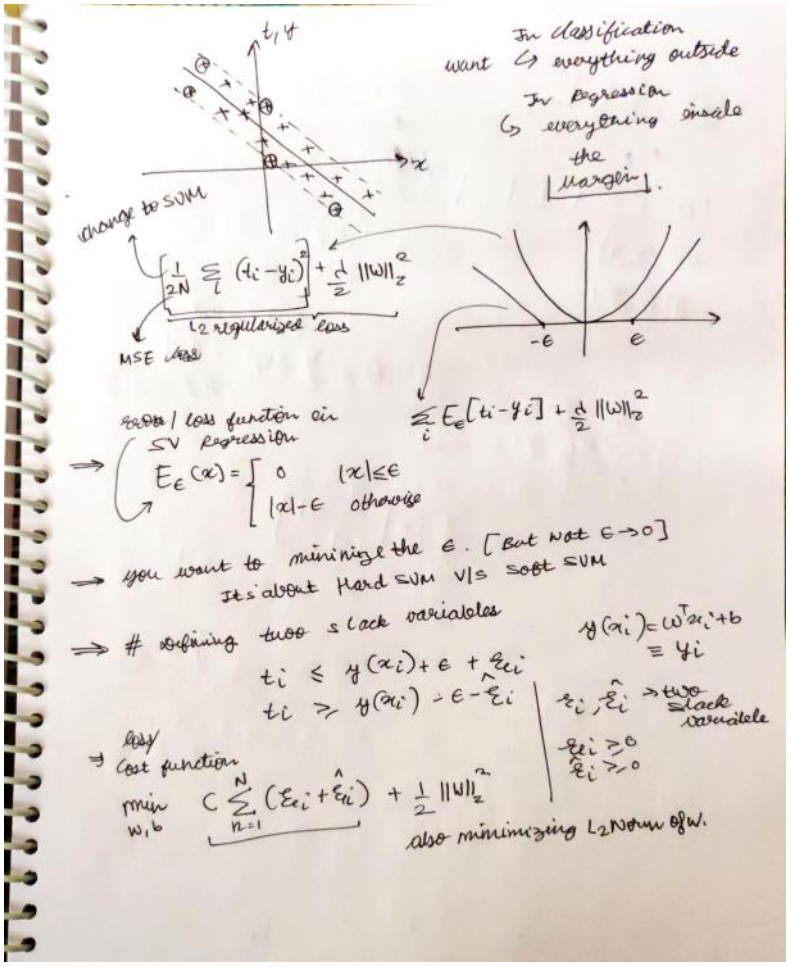


[If $C \to \infty$ all $\xi_{ti} \to 0$ → Hard SVM]

If $C = 0$ fully regularization many misclassification

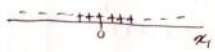

Minimize $\|w\|^2 + C\sum_i \xi_i$ ← slack variable
s.t. $\forall i\ t_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i\ \xi_i \geq 0$
Minimize $_{w,b}$ $\sum_i$ hinge $(t_i[w^T x_i + b]) + \frac{\lambda}{2}\|w\|_2^2$
$\lambda = 0$ → Hard SVM
$\lambda \to \infty$ → fully regularized
sharp
smooth
$\lambda \to 0$
$(\lambda \propto \frac{1}{C})$
In logistic regression : $\min_{w,b}$ CE $+ \frac{\lambda}{2}\|w\|_2^2$
(L2)
(soft SVM) vs (L2 regularized logistic regression)
Loss function changed
hinge loss
cross entropy loss
[slack variable is individual loss of each points]
→ slack variable is constant times hinge loss.
→ slack variable is zero for all correctly classified ($\xi_i = 0$)
$\xi_i > 1$ → Misclassification
$0 < \xi_i < 1$ → correctly classified [Beyond Margin]
$\xi_i = 0$ [Correct side of margin / on margin]
$0 < \xi_i < 1$ [Inside margin]
$\xi_i = 1$ [On boundary]
$\xi_i > 1$ [Misclassified]
⇒ All misclassified and inside margin are also support vectors.
⇒ Any point that can change w & b are SV.
⇒ In soft SVM SVs are more than Hard SVM
$w^T x + b = 1$
$w^T x + b = 0$
$w^T x + b = -1$
If w is not a unit vector
If w is unit vector then $(w^T x + b = \pm 1$ for Margin)
Convex optimization
ξ_i hinge
All points here don't matter for classification
1
$x = t_i(w^T x_i + b)$
Only these matter
Sparse subset





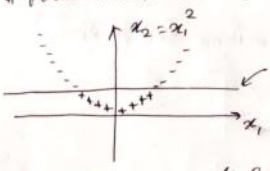
FEATURE ENGINEERING and SELECTION

30/9/24



linear classifier in 1D → single threshold

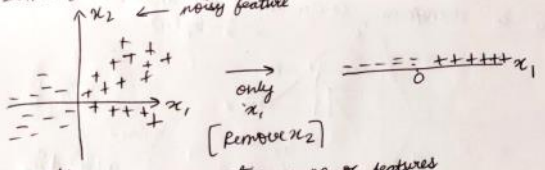
feature selection and engineering

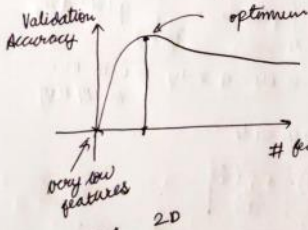


No L2 Regularized ; can lead to instability

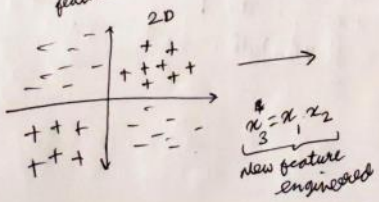
[in grouping effect

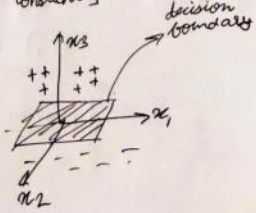
Some feature Reduction is also useful

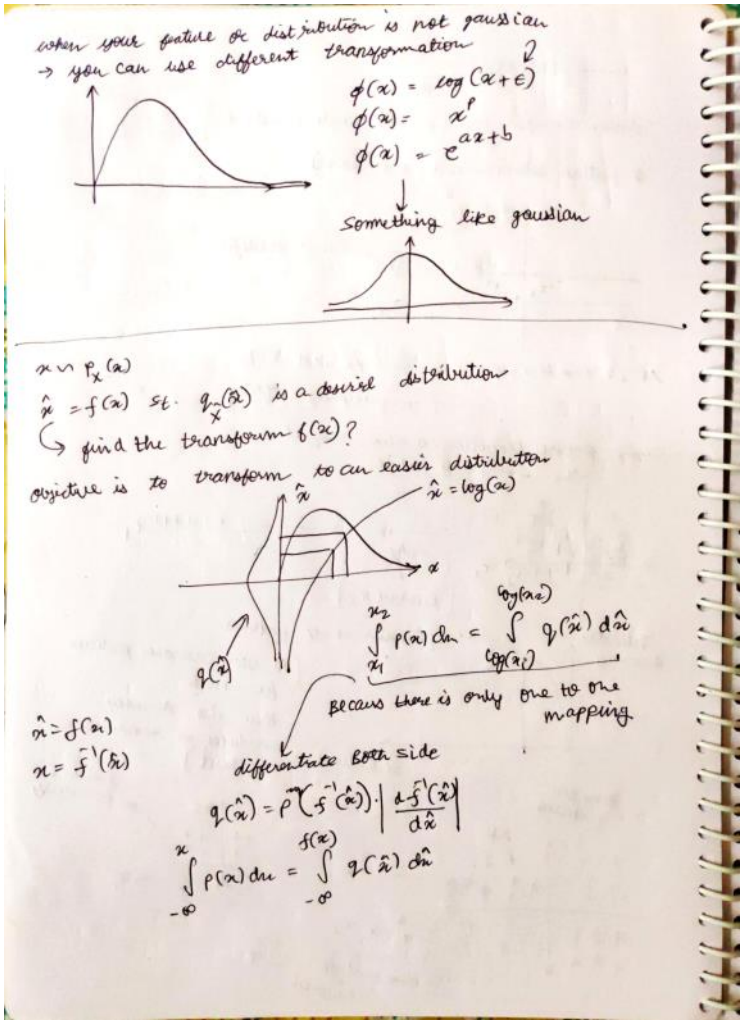


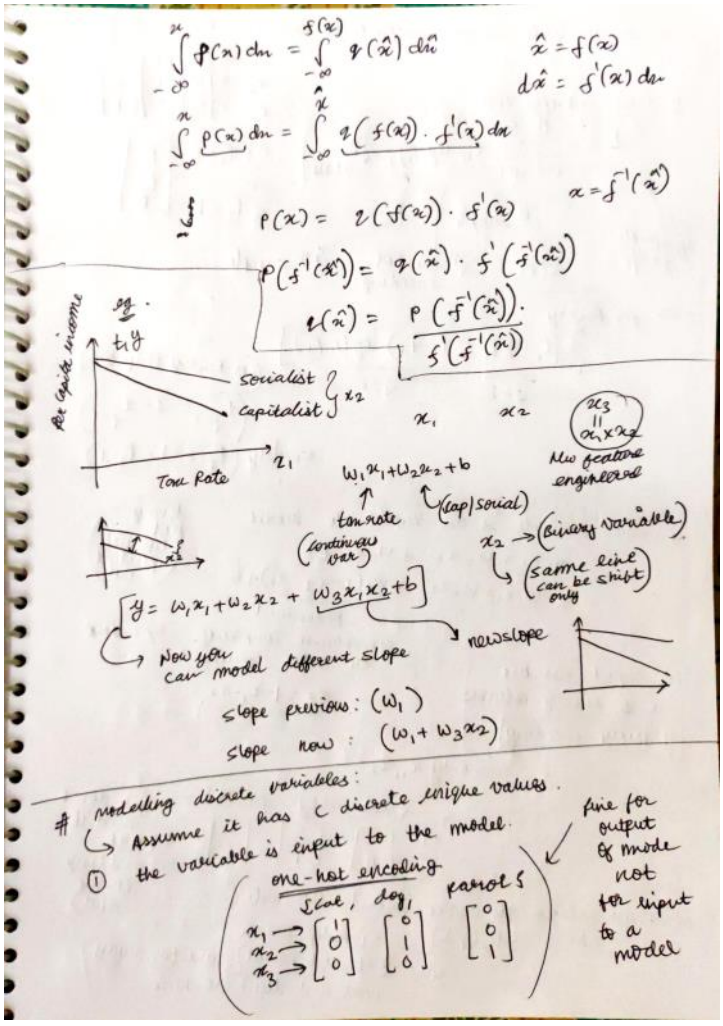


[As you increases features too much then also accuracy reduces or remains constant]

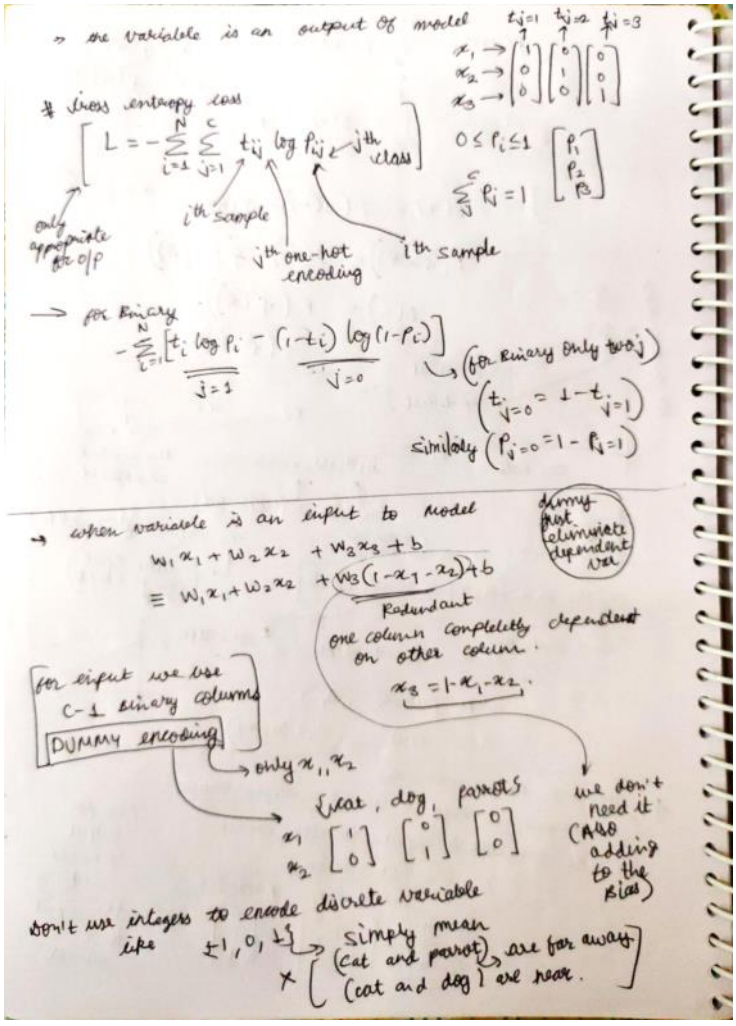





when your feature or distribution is not gaussian
→ you can use different transformation
$\phi(x) = \log(x+\epsilon)$
$\phi(x) = x^{\rho}$
$\phi(x) = e^{ax+b}$
something like gaussian
$x \sim p_x(x)$
$\hat{x} = f(x)$ s.t. $q_{\hat{x}}(\hat{x})$ is a desired distribution
↳ find the transform $f(x)$?
objective is to transform to an easier distribution
$\hat{x} = \log(x)$
$q(\hat{x})$
$\int_{x_1}^{x_2} p(x)\,dx = \int_{\log(x_1)}^{\log(x_2)} q(\hat{x})\,d\hat{x}$
Because there is only one to one mapping
$\hat{x} = f(x)$
$x = f^{-1}(\hat{x})$
differentiate both side
$q(\hat{x}) = p(f^{-1}(\hat{x})) \cdot \left| \frac{d f^{-1}(\hat{x})}{d\hat{x}} \right|$
$\int_{-\infty}^{x} p(x)\,dx = \int_{-\infty}^{f(x)} q(\hat{x})\,d\hat{x}$

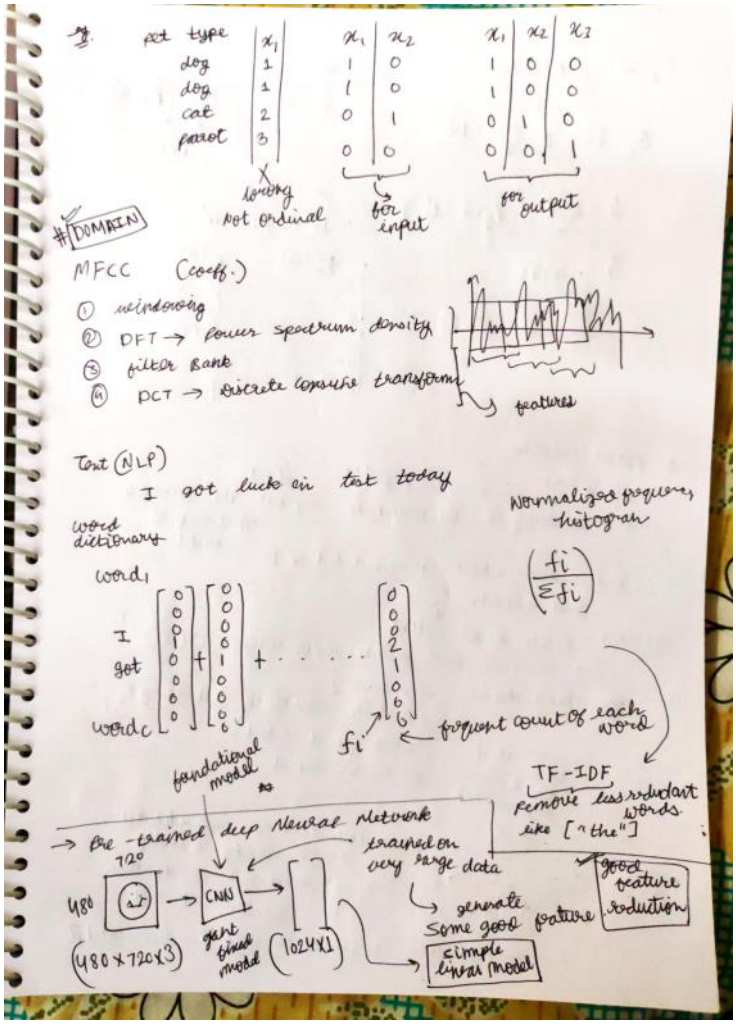


DUMMY ENCODING



DOMAIN SPECIFIC FEATURE SELECTION

MFCC-Mel-frequency Cepstral Coefficients



FEATURE REDUCTION

$P_x(x) = \int_{-\infty}^{x} p_x(x)\,dx$

$Q_{\hat{x}}(\hat{x}) = \int_{-\infty}^{\hat{x}} q_{\hat{x}}(\hat{x})\,d\hat{x}$

$\hat{x} = Q_{\hat{x}}^{-1}\left(\int_{-\infty}^{x} p_x(x)\,dx\right)$

$\hat{x} = f(x)$

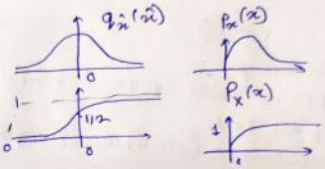
Result

$\hat{x} = g(x) = Q_{\hat{x}}^{-1}(P_x(x))$

CDFs

$\hat{x} = f(x)$

$d\hat{x} = f'(x)\,dx$



feature Reduction

$x \in \mathbb{R}^{D\times 1} \longrightarrow \hat{x} \in \mathbb{R}^{d\times 1}$ $\quad d < D$

↳ while doing so, we want to improve the accuracy, efficiency of model.

↳ Best subset selection of size d out of D.

total subsets ${}^{D}C_{d}$

Method ① · Filter Based Method

↳ examine each feature individually $O(D)$

Eg. Classification: loop over j

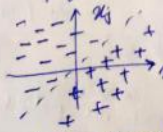
{evaluate utility of just using x_j}

Looking at its own validation performance

select top d subset.

retrain on top d.

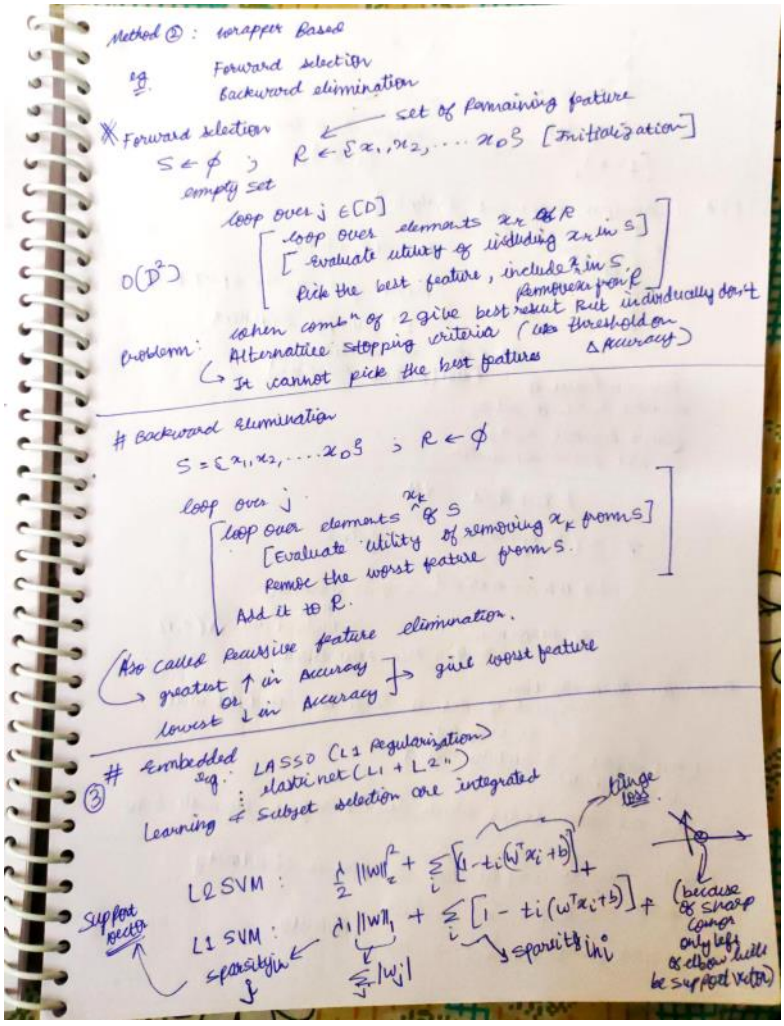
Problem



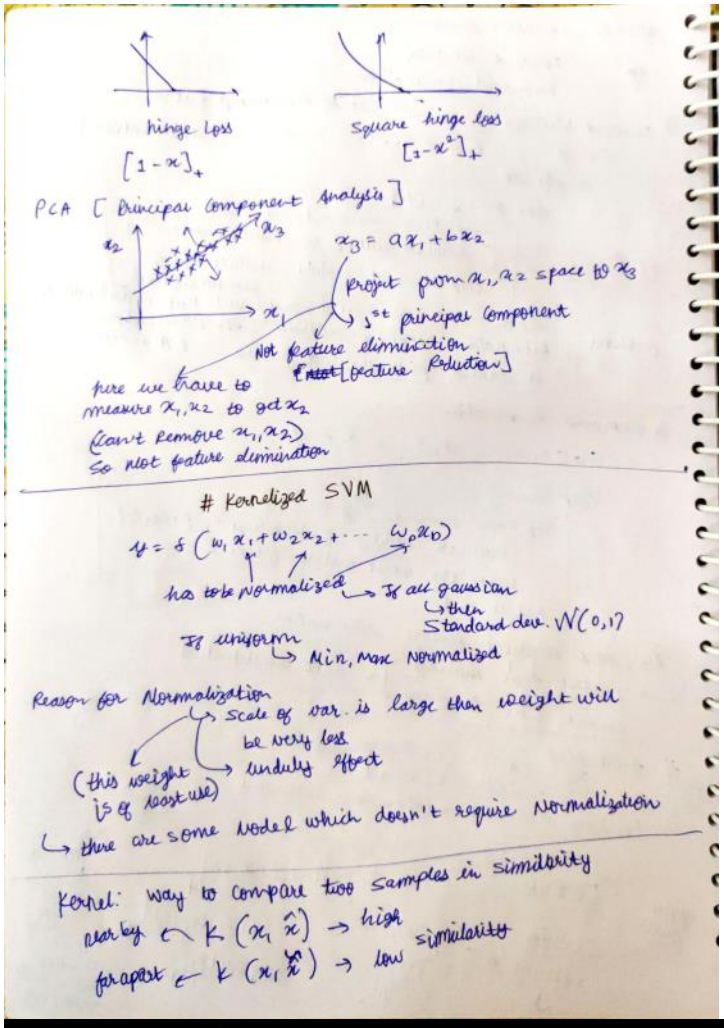
→ Both $x_i \notin x_j$ individually can't classify correctly

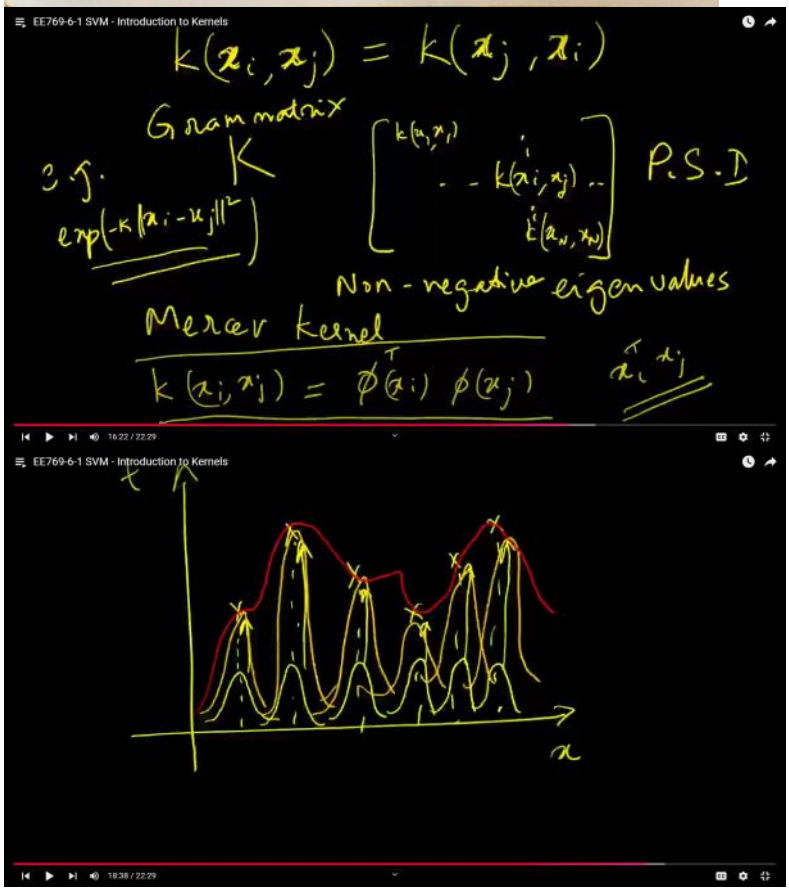
→ so misses this combination if [student's t test]

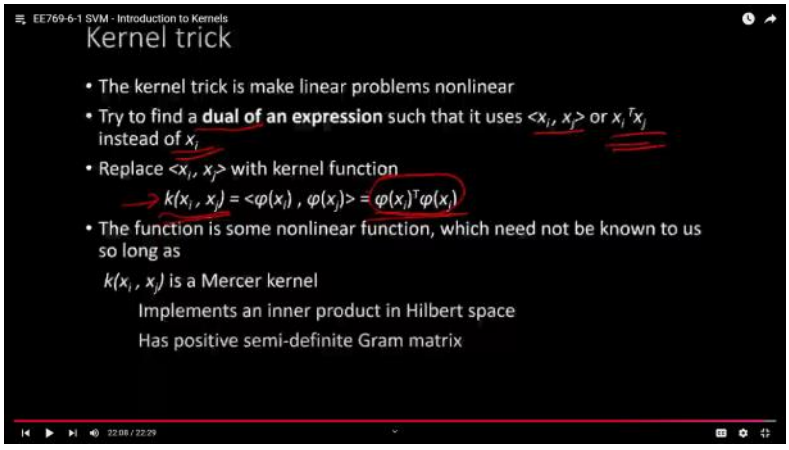
↳ In Regression: → Correlation with $\hat{x}$ and predicted $\hat{x}$

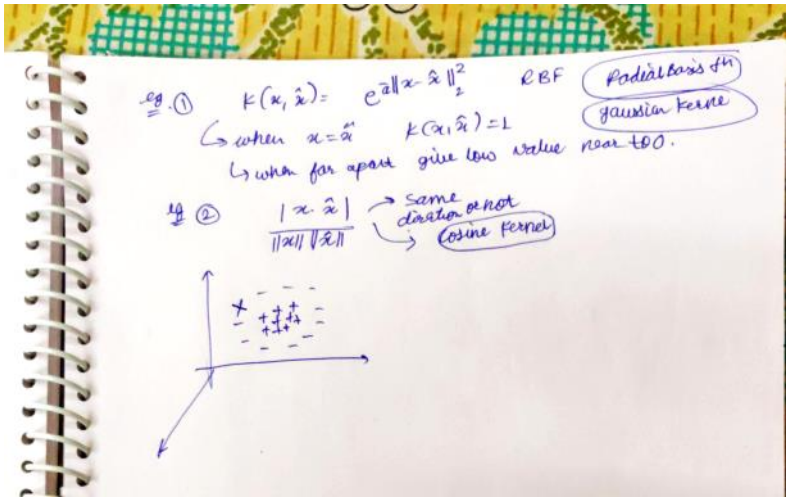


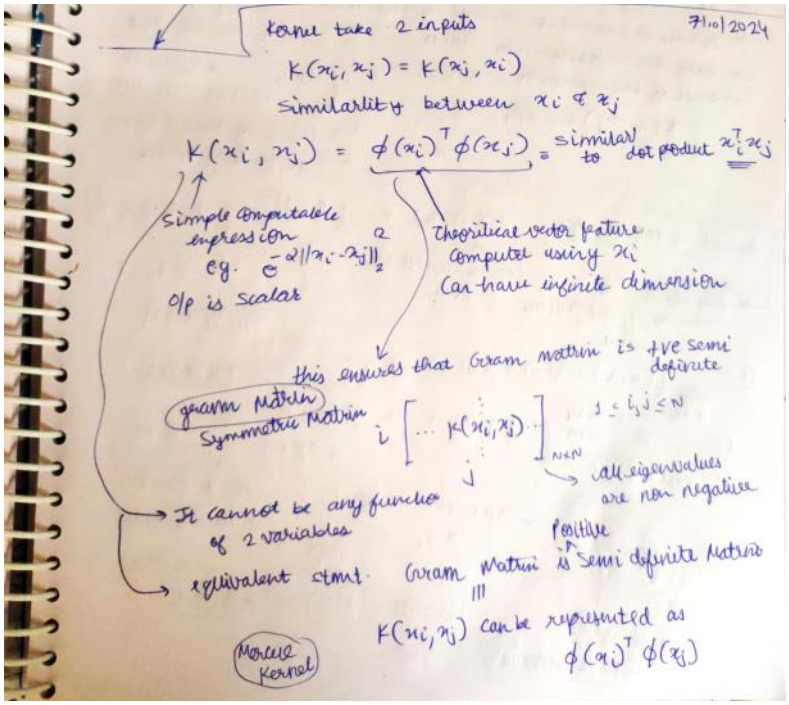
SUPPORT VECTOR REGRESSION (KERNALIZED SVM)


hinge loss
$[1-x]_+$
square hinge loss
$[1-x^2]_+$
PCA [Principal Component Analysis]
$x_3 = ax_1 + bx_2$
project from x_1, x_2 space to x_3
1st principal component
Not feature elimination
~~not~~ [feature Reduction]
pure we have to measure x_1, x_2 to get x_3
(can't remove x_1, x_2)
So not feature elimination
Kernelized SVM
$y = \delta(w_1 x_1 + w_2 x_2 + \cdots w_0 x_0)$
has to be normalized
If all gaussian → then Standard dev. $N(0,1)$
If uniform → Min, Max Normalized
Reason for Normalization
Scale of var. is large then weight will be very less
(this weight is of least use)
unduly effect
there are some model which doesn't require Normalization
Kernel: way to compare two samples in similarity
nearby $k(x_i, \hat{x})$ → high
faraway $k(x_i, \hat{x})$ → low
similarity


EE769-6-1 SVM - Introduction to Kernels
$k(x_i, x_j) = k(x_j, x_i)$
Gram matrix
K
s.t. $\exp(-\kappa \|x_i - x_j\|^2)$
$\begin{bmatrix} k(x_1,x_1) & \cdots & k(x_i,x_j) & \cdots \\ & & & k(x_N,x_N) \end{bmatrix}$
P.S.D
Non-negative eigenvalues
Mercer Kernel
$k(x_i, x_j) = \phi^T(x_i)\,\phi(x_j)$
$x_i^T x_j$
16:22 / 22:29
EE769-6-1 SVM - Introduction to Kernels
x
16:38 / 22:29


EE769-6-1 SVM - Introduction to Kernels
Kernel trick
• The kernel trick is make linear problems nonlinear
• Try to find a dual of an expression such that it uses <xi, xj> or xi^T xj instead of xi
• Replace <xi, xj> with kernel function
k(xi, xj) = <φ(xi), φ(xj)> = φ(xi)^T φ(xj)
• The function is some nonlinear function, which need not be known to us so long as
k(xi, xj) is a Mercer kernel
Implements an inner product in Hilbert space
Has positive semi-definite Gram matrix
22:08 / 22:29


eg ① K(x, x̂) = e^{-α‖x - x̂‖₂²}
RBF
Radial basis fn
gaussian kernel
when x = x̂ K(x, x̂) = 1
when far apart give low value near to 0.
eg ② |x · x̂| / (‖x‖ ‖x̂‖)
same direction or not
cosine kernel


7/10/2024
Kernel take 2 inputs
K(xi, xj) = K(xj, xi)
similarity between xi & xj
K(xi, xj) = φ(xi)^T φ(xj) = similar to dot product xi^T xj
simple computable expression
eg. e^{-α‖xi - xj‖₂²}
o/p is scalar
theoritical vector feature computed using xi
Can have infinite dimension
this ensures that Gram matrix is +ve semi definite
gram Matrix
Symmetric Matrix
1 ≤ i, j ≤ N
N×N
all eigenvalues are non negative
It cannot be any function of 2 variables
equivalent stmt
Positive
Gram Matrix is semi definite Matrix
K(xi, xj) can be represented as φ(xi)^T φ(xj)
Mercer Kernel

LAGRANGIAN

→ we are interested in defining K not ϕ
→ computing kernel is much easier than $\phi(x_i)^T \phi(x_j)$ then dot product
→ you cannot just take any function of two vector, then say it as Mercer kernel, you have to prove that gram matrix is PSD.

Kernel trick

① Take an objective function of x_i and try to change so that it becomes a function of $x_i^T x_j$ manipulate
PRIMAL form → DUAL form

② Replace $x_i^T x_j$ with $\phi(x_i)^T \phi(x_j)$ or $K(x_j, x_i)$

→ It can be applied to several optimization
→ Make linear problem non-linear
→ Dual of the expression
$k(x_i, x_j) = <\phi(x$

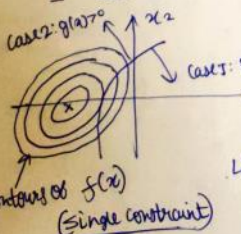
apply it to L_1, L_2, LR, PCA

(Now, in x space it will be non-linear)
↳ linear in Kernel space
↳ in feature space

Primal SVM objective: $L(w,b) = \frac{1}{2}\|w_i\|_2^2 - \sum_i a_i[z_i \cdot (w^T q_i + b) - 1]$
↳ this objective is basically a lagrangian
Loss in weights & Bias
s.t. $\forall a_i \geq 0$
Hard SVM

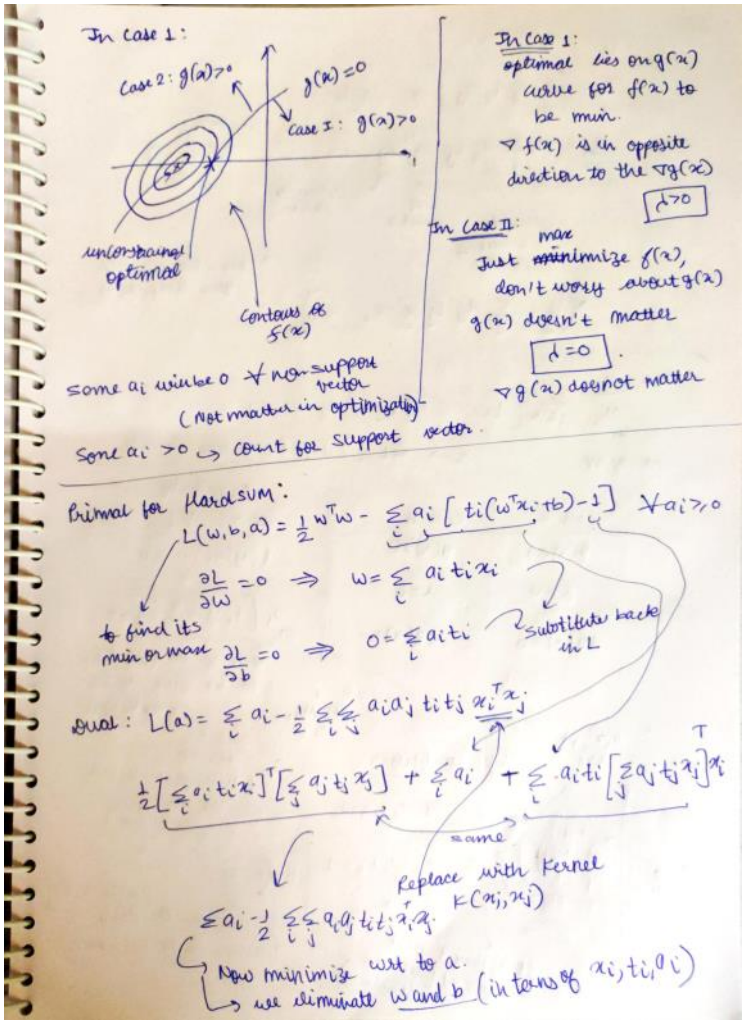
Background
① $L(x, \lambda) = f(x) + \lambda g(x)$; $\lambda \geq 0$
min max
② ≡ max min $f(x)$, subject to $g(x) \geq 0$
equivalent

x-space
$\lambda \neq 0$
for $g(x) = 0$
we already know.



(single constraint)

max min
n constraints
$L(w,b) = f(w) + \sum g(w,a)$


In case 1:
case 2: g(x) > 0
g(x) = 0
case 1: g(x) > 0
unconstrained optimal
Contours of f(x)
In case 1:
optimal lies on g(x) curve for f(x) to be min.
∇f(x) is in opposite direction to the ∇g(x)
λ > 0
In case II:
Just minimize f(x), don't worry about g(x)
g(x) doesn't matter
λ = 0
∇g(x) doesnot matter
some a_i will be 0 ∀ non support vector (Not matter in optimization)
some $a_i > 0$ → count for support vector.
Primal for Hardsvm:
$L(w,b,a) = \frac{1}{2} w^T w - \sum_i a_i [t_i(w^T x_i + b) - 1] \quad \forall a_i \geq 0$
$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_i a_i t_i x_i$
to find its min or max
$\frac{\partial L}{\partial b} = 0 \Rightarrow 0 = \sum_i a_i t_i$
Substitute back in L
Dual: $L(a) = \sum_i a_i - \frac{1}{2} \sum_i \sum_j a_i a_j t_i t_j x_i^T x_j$
$\frac{1}{2}\left[\sum_i a_i t_i x_i\right]^T \left[\sum_j a_j t_j x_j\right] + \sum_i a_i + \sum_i a_i t_i \left[\sum_j a_j t_j x_j\right]^T x_i$
same
Replace with Kernel $K(x_i, x_j)$
$\sum a_i - \frac{1}{2} \sum_i \sum_j a_i a_j t_i t_j x_i x_j$
Now minimize wrt to a.
→ we eliminate w and b (in terms of x_i, t_i, a_i)

$y_i = w^T x_i + b$

$y_i = \left[\sum_j a_j t_j x_j\right]^T x_i + b$

$y_i = \sum_j a_j t_j x_i^T x_j + b$

Replace $K(x_i, x_j)$

$x_i^T x_j$: test points, training points

a_j: weights of training points

only SV will help you compute this.

$y = \sum_j a_j t_j x^T x_j + b$

$b = \frac{1}{N_S} \sum_{i \in S} \left[t_i - \sum_{j \in S} a_j t_j x_i^T x_j \right]$

N_S → no. of SV

S → set of support vectors $(a_i > 0)$

similarly

$a_i \geq 0 \ \forall i$

$t_i y(x_i) - 1 \geq 0$

$a_i [t_i y(x_i) - 1] = 0$

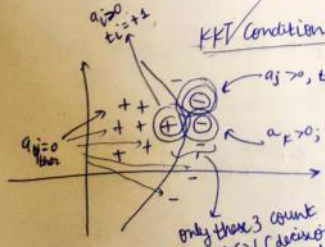
one of the two has to be zero to be in margin.

$= 0$ for SV. [these point defined the decision boundary]

$= 0$ for non SV

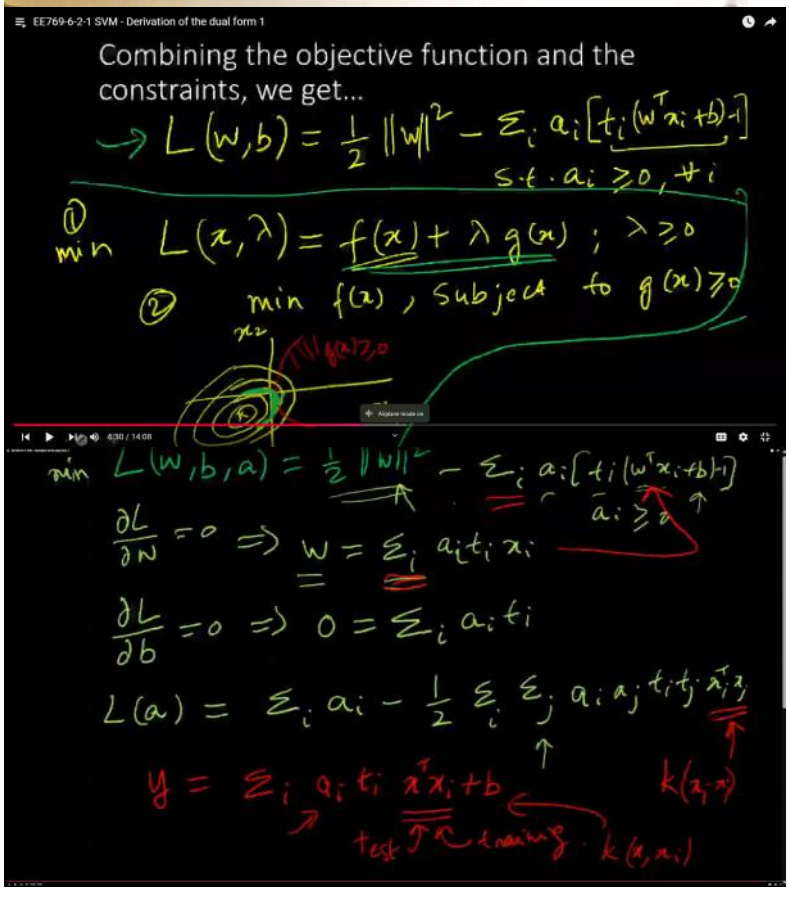
Case I	Case II
$g(x) = 0$	$g(x) \geq 0$
$\lambda > 0$	$\lambda = 0$
$dg(x) = 0$	$dg(x) = 0$

KKT Conditions:



$0 = [a_i x_i^T x - a_j x_j^T x - a_k x_k^T x + b]$

(decision boundary linear in Kernel space)



Solving the dual Lagrangian *L(a)*

$$\min L(x) = f(x) + \lambda g(x)$$

Case1	Case2	$g(x) \geq 0$
$g(x) = 0$	$g(x) > 0$	$\lambda \geq 0$
$\lambda > 0$	$\lambda = 0$	$\lambda g(x) = 0$
$\lambda g(x) = 0$	$\lambda g(x) = 0$	

$$L(a) = \sum_i a_i - \frac{1}{2} \sum_i \sum_j a_i a_j t_i t_j x_i^T x_j$$

$$L(w, b, a) = \frac{1}{2} \|w\|^2 - \sum_i a_i [t_i (w^T x_i + b) - 1]$$

$a_i > 0 \Rightarrow$ S.V. $\Rightarrow x_i \in S$

$a_i \geq 0$

$t_i \, y(x_i) - 1 \geq 0$

$a_i [t_i \, y(x_i) - 1] = 0$

$$t_i \left[\sum_{j \in S} a_j t_j x_i^T x_j + b \right] = 1$$

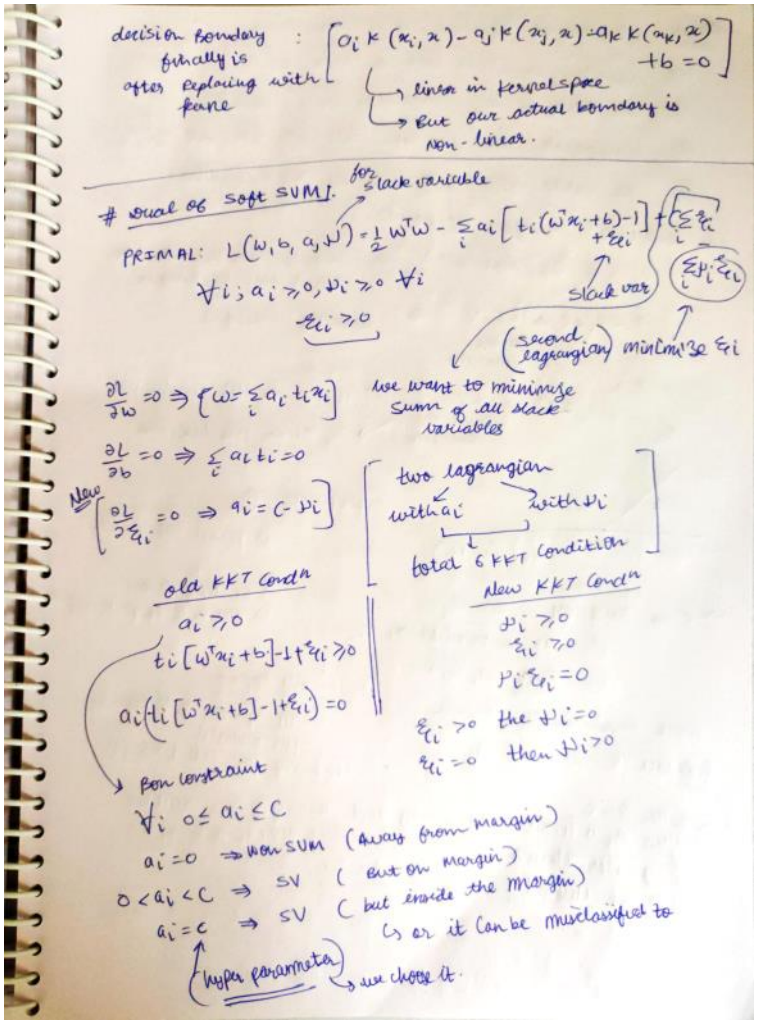
$$b = \frac{1}{N_S} \sum_{i \in S} \left[t_i - \sum_{j \in S} a_j t_j x_i^T x_j \right]$$

Analysing the KKT conditions

Not. S.V. $\notin S$

- Interpreting the points where $a_n = 0$
  - For these points, $t_n\{y(x_n)-1\}$ does not need to be zero
  - These points do not define the decision boundary
  - These points can be moved around within most of the decision region without changing the boundary
- Interpreting the points where $a_n > 0$
  - For these points, $t_n\{y(x_n)-1\} = 0$
  - These points define the decision boundary
  - These points need to be retained after training for classification of test data, while rest of the points can be discarded
  - These points are called SUPPORT VECTORS

S.V.


decision boundary finally is after replacing with kernel : $\left[\alpha_i K(x_i, x) - a_j K(x_j, x) - a_k K(x_k, x) + b = 0\right]$
linear in kernel space
But our actual boundary is non-linear.
Dual of soft SVM:
for slack variable
PRIMAL: $L(w, b, a, \mu) = \frac{1}{2} w^T w - \sum_i a_i \left[t_i (w^T x_i + b) - 1 + \xi_i \right] + \left[\sum_i \xi_i \right]$
$\sum_i \mu_i \xi_i$
$\forall i, a_i \geq 0, \mu_i \geq 0 \; \forall i$
$\xi_i \geq 0$
slack var
(second lagrangian) minimize ξ_i
$\frac{\partial L}{\partial w} = 0 \Rightarrow \left[w = \sum_i a_i t_i x_i \right]$
we want to minimize sum of all slack variables
$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_i a_i t_i = 0$
New
$\left[\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow a_i = C - \mu_i \right]$
two lagrangian
with a_i
with μ_i
total 6 KKT condition
old KKT condn
$a_i \geq 0$
$t_i [w^T x_i + b] - 1 + \xi_i \geq 0$
$a_i (t_i [w^T x_i + b] - 1 + \xi_i) = 0$
New KKT condn
$\mu_i \geq 0$
$\xi_i \geq 0$
$\mu_i \xi_i = 0$
$\xi_i > 0$ the $\mu_i = 0$
$\xi_i = 0$ then $\mu_i > 0$
Box constraint
$\forall i \; 0 \leq a_i \leq C$
$a_i = 0 \Rightarrow$ Non SVM (Away from margin)
$0 < a_i < C \Rightarrow$ SV (But on margin)
$a_i = C \Rightarrow$ SV (but inside the margin)
Or it can be misclassified to
Hyper parameter
we choose it.

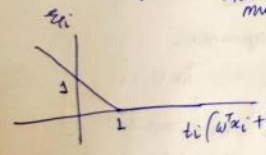
Steps: ① change objective to a lagrangian (a, w)
② equate derivatives wrt to w, b to 0.
③ substitute of w, expression in a's and t's.
④ Manipulate the loss eqn. till you get all $x_i^T x_j$.
⑤ Re write the constraints as KKT conditions

Box constraints: 4 types of points

$a_i = 0 \Rightarrow$ Not a S.V. $\Rightarrow$ correctly classified and on correct side of margin

$0 < a_i < C \Rightarrow$ S.V. on the margin $\Rightarrow$ correctly classified

$a_i = C \Rightarrow$ S.V. inside the margin
- $\xi_i \leq 1 \Rightarrow$ Not misclassified
- $\xi_i > 1 \Rightarrow$ Misclassified



Hyper params: ① C
② kernel-hyperparam
eg. scaling factors
↳ power of poly^n kernel

→ C in Hard SVM. $C = \infty$ → no constraints on a_i
Just $a_i \geq 0$

More regularization → less C
[reduce C more S.V.s]

forcing $\xi_i = 0$ all $\xi_i = 0$; nothing cross the margin looking for fixed margin

→ with less C your a_i is limited to const and many of a_is will be greater than 0.
↳ force more a_i to be $a_i > 0$.
↳ So Number of S.V.s increases.

Analysing the KKT conditions

← Not. S.V. ∉ S

- Interpreting the points where $a_n = 0$
  - For these points, $t_n\{y(x_n)-1\}$ does not need to be zero
  - These points do not define the decision boundary
  - These points can be moved around within most of the decision region without changing the boundary
- Interpreting the points where $a_n > 0$
  - For these points, $t_n\{y(x_n)-1\} = 0$
  - These points define the decision boundary
  - These points need to be retained after training for classification of test data, while rest of the points can be discarded
  - These points are called SUPPORT VECTORS

S.V.

What if the classes are not separable?

- We need to allow some misclassification
- An error function that gives infinite error upon misclassification needs to be modified
- We introduce a slack variable $\xi_n \geq 0$ for each point n, such that ξ_n is:
  - 0, if the point is on the correct side of margin boundary
  - $|t_n - y(x_n)|$ for points outside the margin

$$L(w,b,a,\mu) = \frac{1}{2}\|w\|^2 - \sum_i a_i [t_i y(x_i) - 1 + \xi_i] + C\sum_i \xi_i - \sum_i \mu_i \xi_i$$

Lagrange

$a_i \geq 0$

$t_i y(x_i) - 1 + \xi_i \geq 0$

$a_i [t_i y(x_i) - 1 + \xi_i] = 0$

$\mu_i \geq 0$

$\xi_i \geq 0$

$\mu_i \xi_i = 0$

$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_i a_i t_i x_i$

$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_i a_i t_i = 0$

$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow a_i = C - \mu_i$

$$\tilde{L}(a) = \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i a_j t_i t_j x_i^T x_j$$

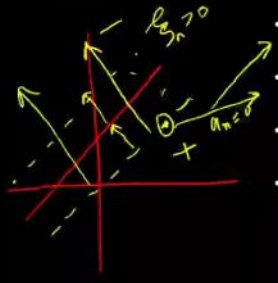
$0 \leq a_i \leq C$; $\sum_i a_i t_i = 0$

$a_i = 0$; not s.v.

$C > a_i > 0$; $\xi_i \leq 1$ correct, but inside margin

$a_i = C$; $\xi_i \geq 1$ $\Rightarrow$ incorrectly classified.

Box constraints: Four types of points



- Inside the margin (and correctly classified)
  - $a_n=0$
- On the margin (and correctly classified)
  - $a_n<C$, $\xi_n=0$
- Across the margin and correctly classified
  - $a_n=C$, $\xi_n \leq 1$
- Across the margin and misclassified
  - $a_n=C$, $\xi_n>1$

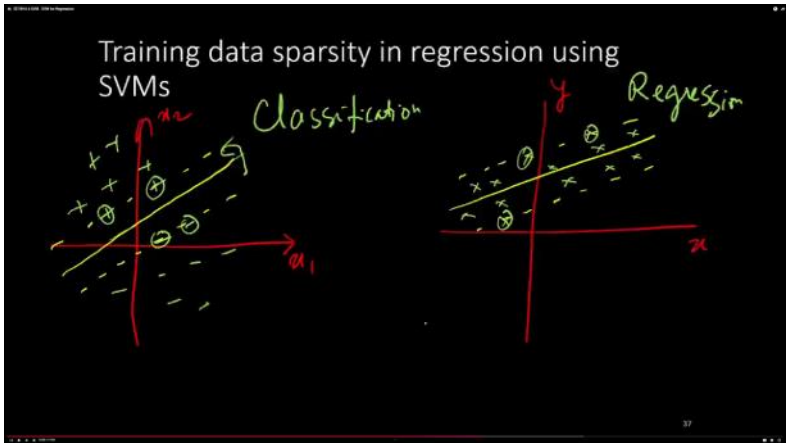
11

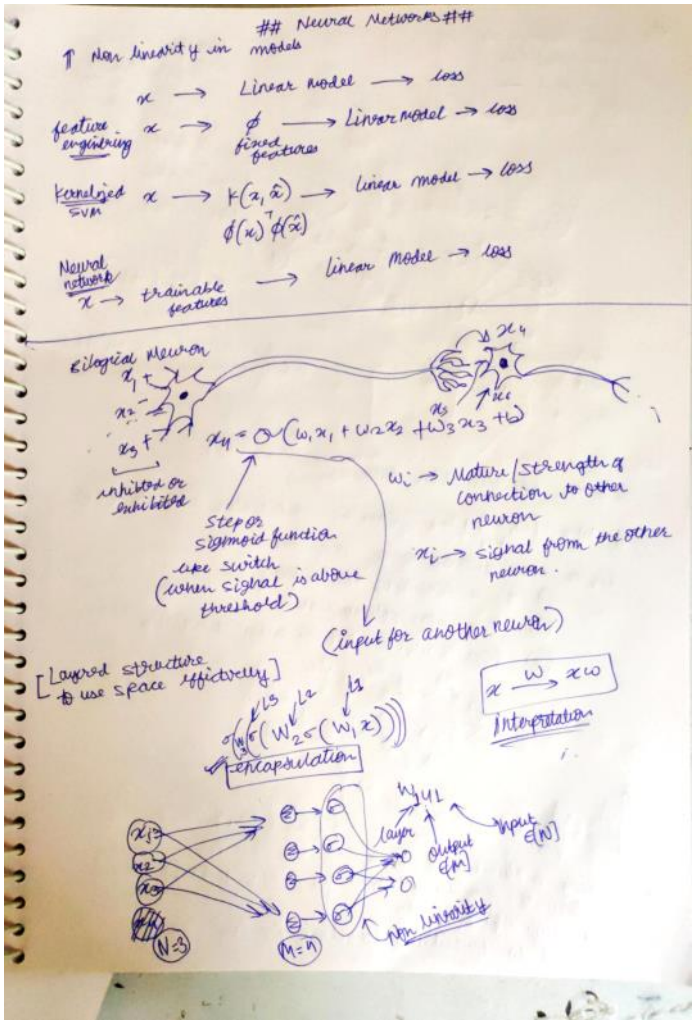
Source: PRML book by C. Bishop

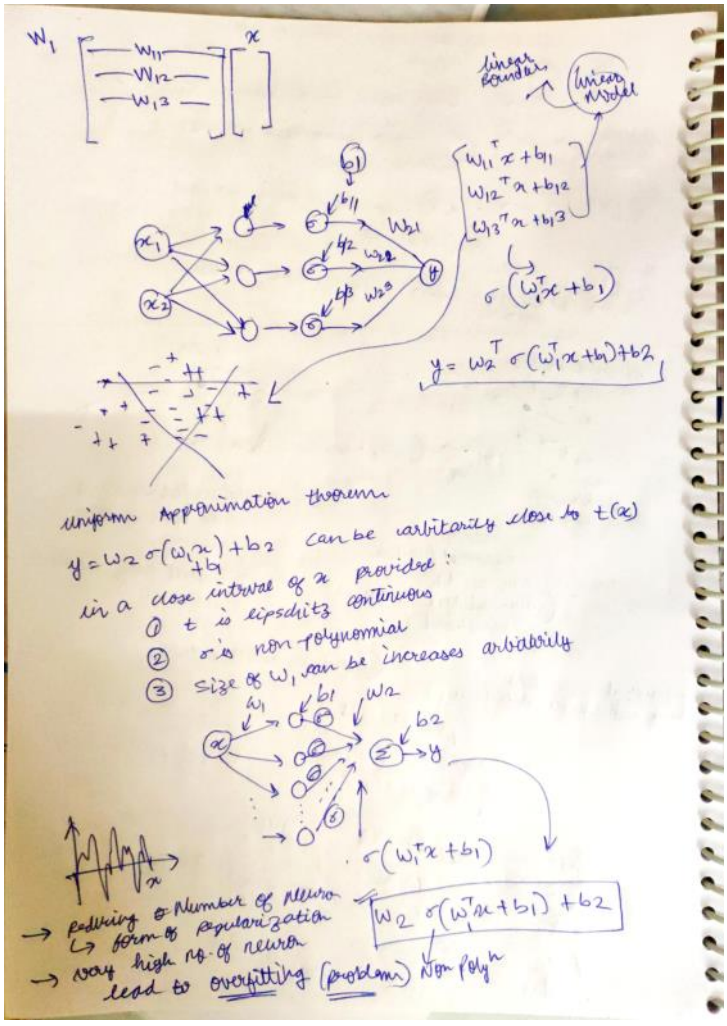
What defines the boundary?

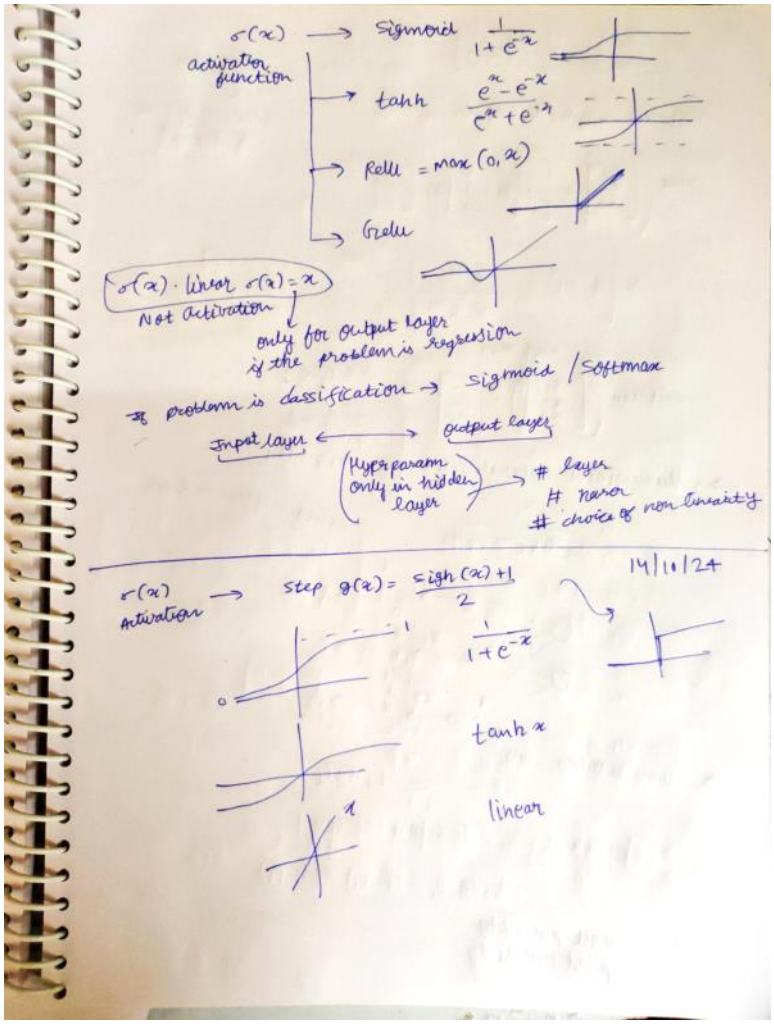
- The following points define the boundary:
  - $\xi_n=0$, on the margin (same as the separable case)
  - $0<\xi_n \leq 1$, across the margin, but not misclassified
  - $\xi_n \geq 1$, misclassified points
- For these points: because $a_n>0$
- Again, $y(x_n)$ calculation remains the same

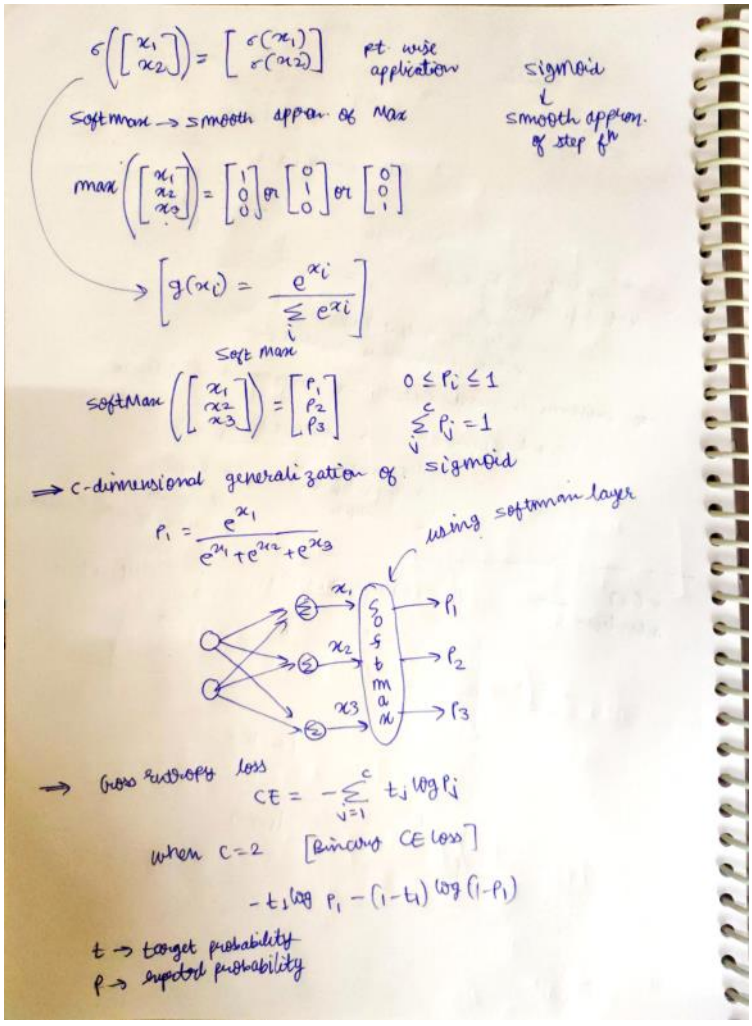
33


Training data sparsity in regression using SVMs
Classification
Regression
Loss
E_ε
Regression
$-\varepsilon$
$+\varepsilon$
$y_i - t_i$
M.S.E. : $\frac{1}{2}\sum_i (y_i - t_i)^2 + \frac{\lambda}{2}\|w\|^2$
S.U.R. : $C\sum_i E_\varepsilon(y_i - t_i) + \frac{1}{2}\|w\|^2$
$E_\varepsilon(y_i - t_i) = \max(0, |y_i - t_i| - \varepsilon)$
Two slack var-s. $t_i \le y_i + \varepsilon + \xi_i$; $t_i \ge y_i - \varepsilon - \hat{\xi}_i$
$C\sum_i (\xi_i + \hat{\xi}_i) + \frac{1}{2}\|w\|^2$


Neural Networks
Non linearity in models
Linear model → loss
feature engineering
fixed features
Linear model → loss
kernelized SVM
Linear model → loss
Neural network
trainable features
Linear model → loss
Biological Neuron
inhibited or exhibited
Step or sigmoid function
like switch
(when signal is above threshold)
w_i → Nature/strength of connection to other neuron
x_i → signal from the other neuron.
(input for another neuron)
[Layered structure to use space effectively]
Encapsulation
$x \xrightarrow{W} xw$
Interpretation
Layer
Input
Output
Non linearity


linear boundary
linear model
uniform Approximation theorem
$y = w_2 \sigma(w_1 x + b_1) + b_2$ can be arbitarily close to $t(x)$
in a close interval of x provided:
① t is lipschitz continuous
② σ is non-polynomial
③ Size of w_1 can be increases arbitarily
→ reducing # Number of neuro
↳ form of regularization
→ very high no. of neuron
lead to overfitting (problem)
Non Poly


σ(x) → Sigmoid $\frac{1}{1+e^{-x}}$
activation function
→ tanh $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
→ Relu = max(0, x)
→ Gelu
σ(x) · linear σ(x) = x
Not Activation
only for output layer
if the problem is regression
if problem is classification → sigmoid / Softmax
Input layer ⟷ output layer
Hyperparam only in hidden layer → # layer
neron
choice of non linearity
σ(x) Activation → step g(x) = sign(x) + 1 / 2
14/10/24
tanh x
linear



BACK PROPAGATION

Multi-class SVM

choice 1 — class i vs class j — $\binom{C}{2}$ SVMs

choice 2 — class i vs rest SVMs — C-SVMs

you have to trained this much SVMs.

$\arg\max_j \left[W_j^T x_i + b_j \right]$ → whichever j it is largest

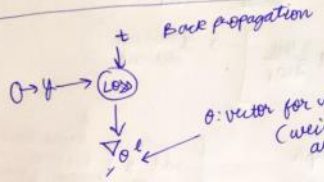
for choice 2 [$j \rightarrow$ class]

decided class for any point x in the space

$\Rightarrow$ we usually use choice 2.

$j = 1, \ldots C$ — total no. of classes.

Back propagation → uses chain rule of derivatives

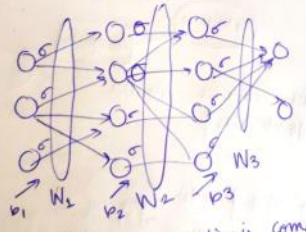


θ: vector for all params (weight, biases) and I

eg.

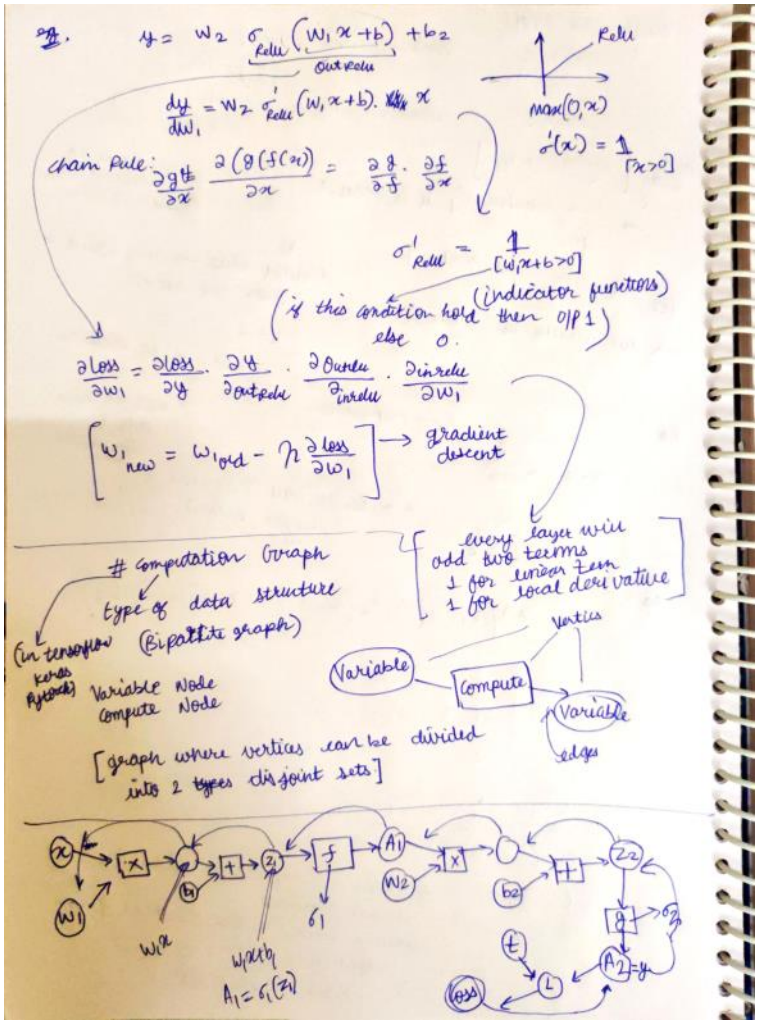
$$l = \text{loss}\left(t, \sigma_3\left(W_3 \sigma_2\left(W_2 \sigma_1\left(W_1 x + b_1\right) + b_2\right) + b_3\right)\right)$$

y



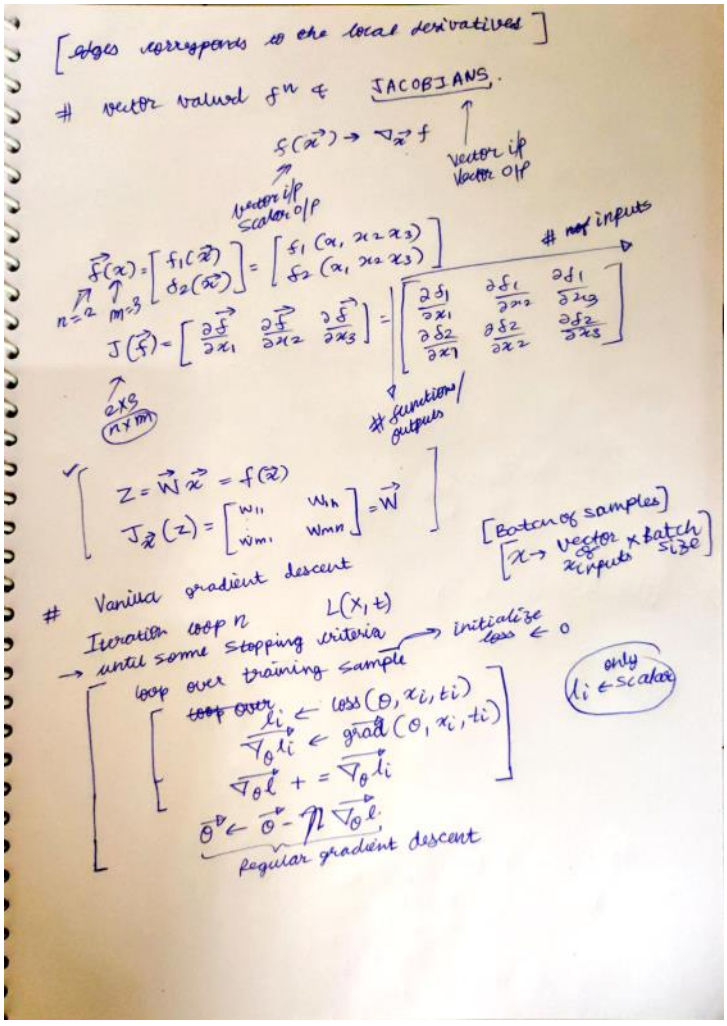
$\frac{\partial l}{\partial y} = \frac{\partial \text{loss}}{\partial y}$ → derivative is computed to train the neural network coming back to the signal from output (y) (x)

→ BACK PROPAGATION

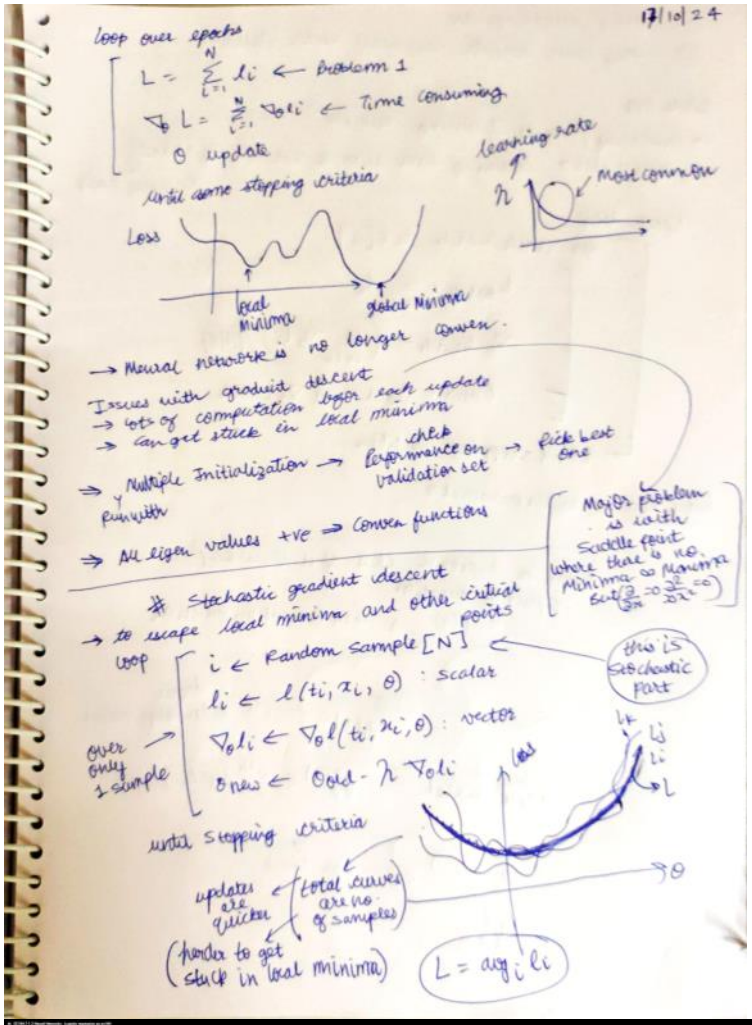


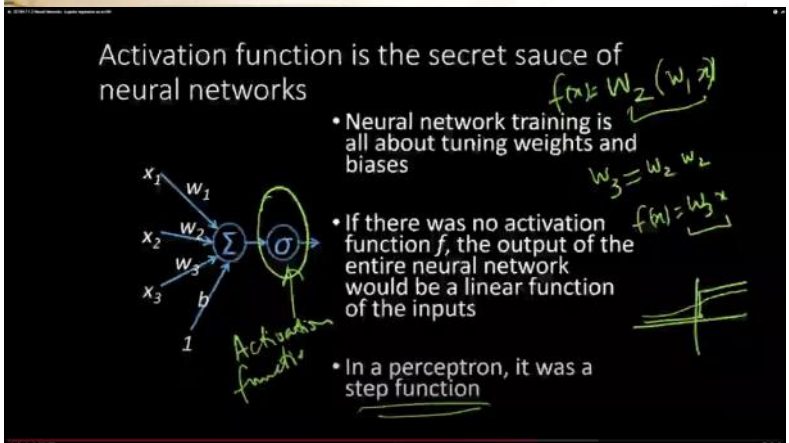
JACOBIAN

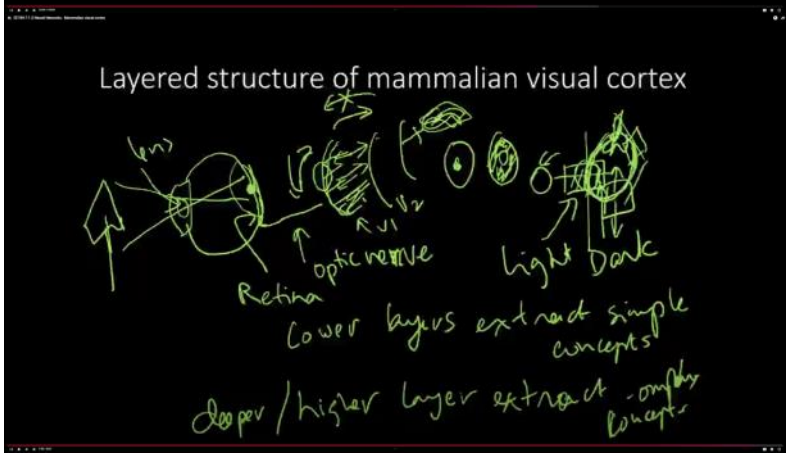
VANILLA GRADIENT DESCENT



STOCHASTIC GRADIENT DESCENT

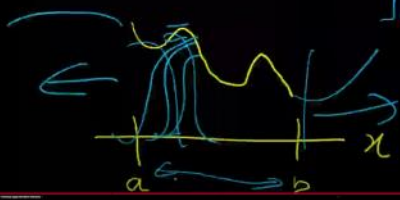

17/10/24
loop over epochs
$L = \sum_{i=1}^{N} l_i$ ← Problem 1
$\nabla_\theta L = \sum_{i=1}^{N} \nabla_\theta l_i$ ← Time consuming
θ update
until some stopping criteria
learning rate
Most common
Loss
local Minima
global Minima
→ Neural network is no longer convex.
Issues with gradient descent
→ lots of computation bcoz each update
→ can get stuck in local minima
→ Multiple initialization → check performance on validation set → pick best one
run with
→ All eigen values +ve ⇒ convex functions
Major problem is with Saddle point where there is no Minima as Slope $\frac{\partial}{\partial x} = 0 \frac{\partial^2}{\partial x^2} = 0$
* Stochastic gradient descent
→ to escape local minima and other critical points
loop
i ← Random sample [N]
$l_i \leftarrow l(t_i, x_i, \theta)$: scalar
$\nabla_\theta l_i \leftarrow \nabla_\theta l(t_i, x_i, \theta)$: vector
$\theta_{new} \leftarrow \theta_{old} - \lambda \nabla_\theta l_i$
over only 1 sample
this is Stochastic part
until stopping criteria
updates are quicker
total curves are no. of samples
harder to get stuck in local minima
$L = \text{avg}_i\, l_i$
θ


Activation function is the secret sauce of neural networks
$f(x) = W_2(W_1 x)$
• Neural network training is all about tuning weights and biases
$W_3 = W_2 W_1$
$f(x) = W_3 x$
• If there was no activation function f, the output of the entire neural network would be a linear function of the inputs
• In a perceptron, it was a step function
Activation function

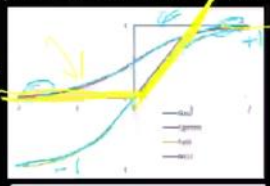

Layered structure of mammalian visual cortex
Retina
optic nerve
Light
Dark
Lower layers extract simple concepts
deeper / higher layer extract complex concepts

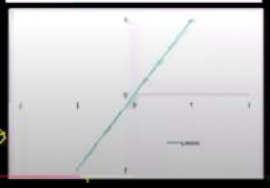
Universal approximation theorem

- $f_\epsilon(W_2\,\sigma(W_1\,x)$ can approximate within error ϵ in a compact interval, any smooth function $f(x)$, provided
  - Size of W's is arbitrary
  - σ is also smooth but not a polynomial



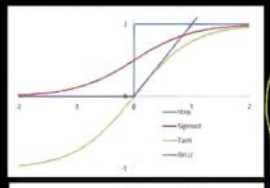
Types of activation functions

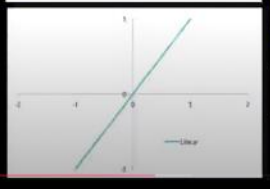




- **Step**: original concept behind classification and region bifurcation. Not used anymore
- **Sigmoid** and tanh: trainable approximations of the step-function
- **ReLU**: currently preferred due to fast convergence
- **Softmax**: currently preferred for output of a *classification net*. Generalized sigmoid
- **Linear**: good for modeling a range in the output of a *regression net*

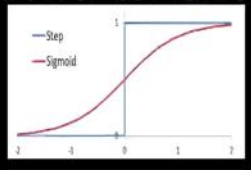
Formulas for activation functions





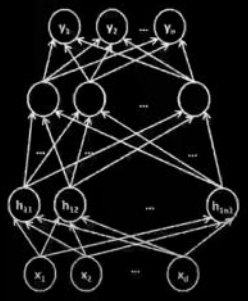
- **Step**: $g(x) = \frac{\text{sign}(x)+1}{2}$
- **Sigmoid**: $g(x) = \frac{1}{1+e^{-x}}$
- **Tanh**: $g(x) = \tanh(x)$
- **ReLU**: $g(x) = \max(0, x)$
- **Softmax**: $g(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}}$
- **Linear**: $g(x) = x$

The problem with sigmoid is (near) zero gradient on both extremes



- For both large positive and negative input values, sigmoid doesn't change much with change of input
- ReLU has a constant gradient for almost half of the inputs
- But, ReLU cannot give a meaningful final output

Basic structure of a neural network

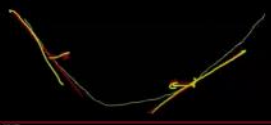


- **Output layer**
  - Represent the output of the neural network
  - For a two class problem or regression with a 1-d output, we need only one output node
- **Hidden layer(s)**
  - Represent the intermediary nodes that divide the input space into regions with (soft) boundaries
  - These usually form a *hidden layer*
  - Usually, there is only one such layer
  - Given enough hidden nodes, we can model an arbitrary input-output relation.
- **Input layer**
  - Represent dimensions of the input vector (one node for each dimension)
  - These usually form an *input layer*, and
  - Usually there is only one such layer

Gradient descent minimizes the loss function

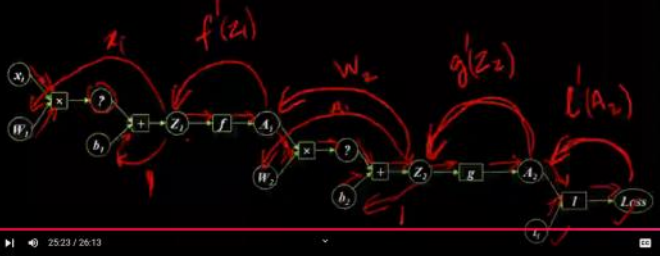
- At every point, compute
  - Loss (scalar): $l_i(\boldsymbol{w})$
  - Gradient of loss with respect to weights (vector): $\nabla_{\boldsymbol{w}} l_i(\boldsymbol{w})$
  - Take a step towards negative gradient:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \left(\frac{1}{N} \sum_{i=1}^{N} l_i(\boldsymbol{w}) \right)$$



EE769-7-2-1 Neural Networks - Chain rule and backprop

1. Make a forward pass and store partial derivatives
2. During backward pass multiply partial derivatives



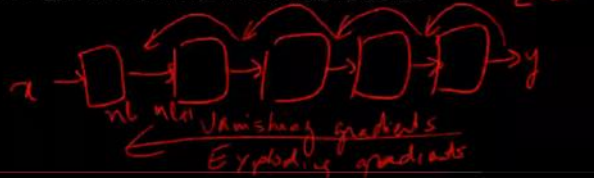
25:23 / 26:13

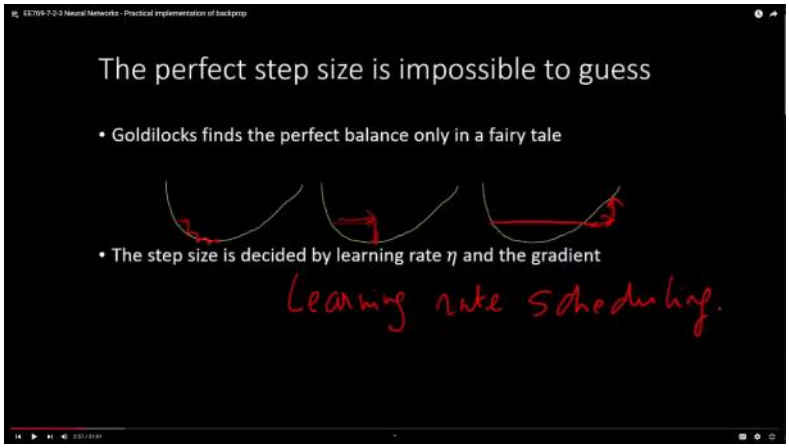
Some questions

- What if we scale all the weights and biases by a factor?

$(Wx+b)_+$ $(\alpha Wx+\alpha b)_+$

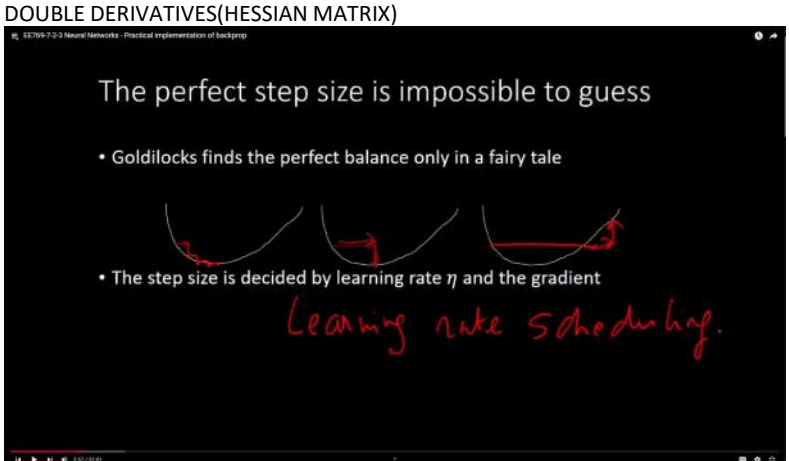
- What happens to gradients in deep neural networks?

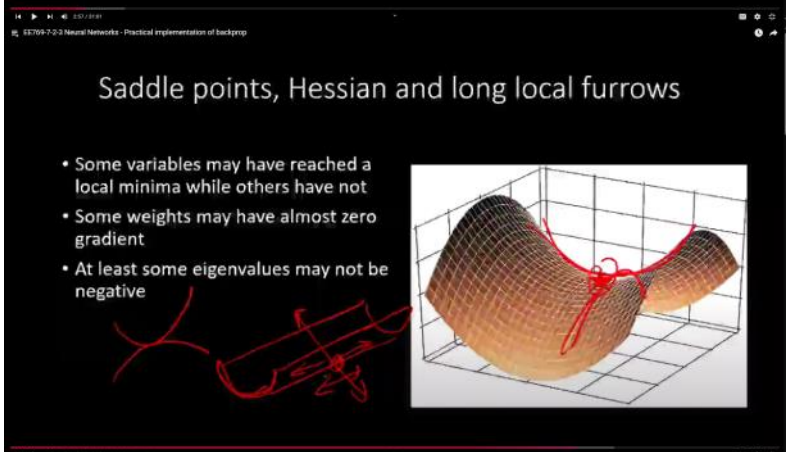


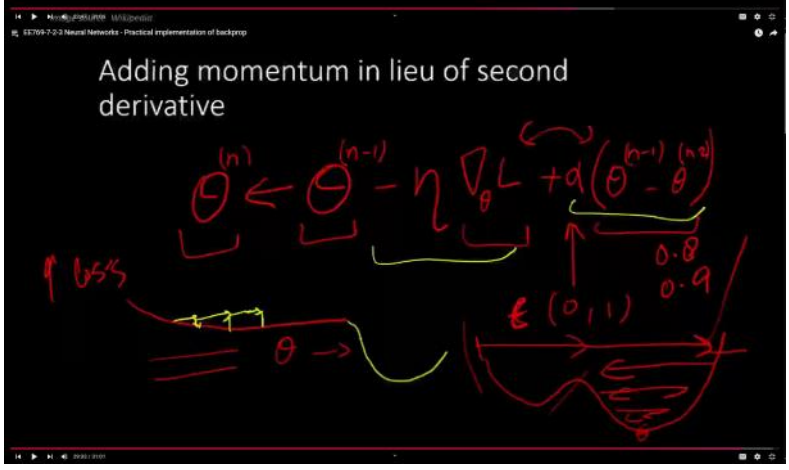


BATCH GRADIENT DESCENT

DOUBLE DERIVATIVES(HESSIAN MATRIX)







ADDING MOMENTUM IN GRADIENT DESCENT

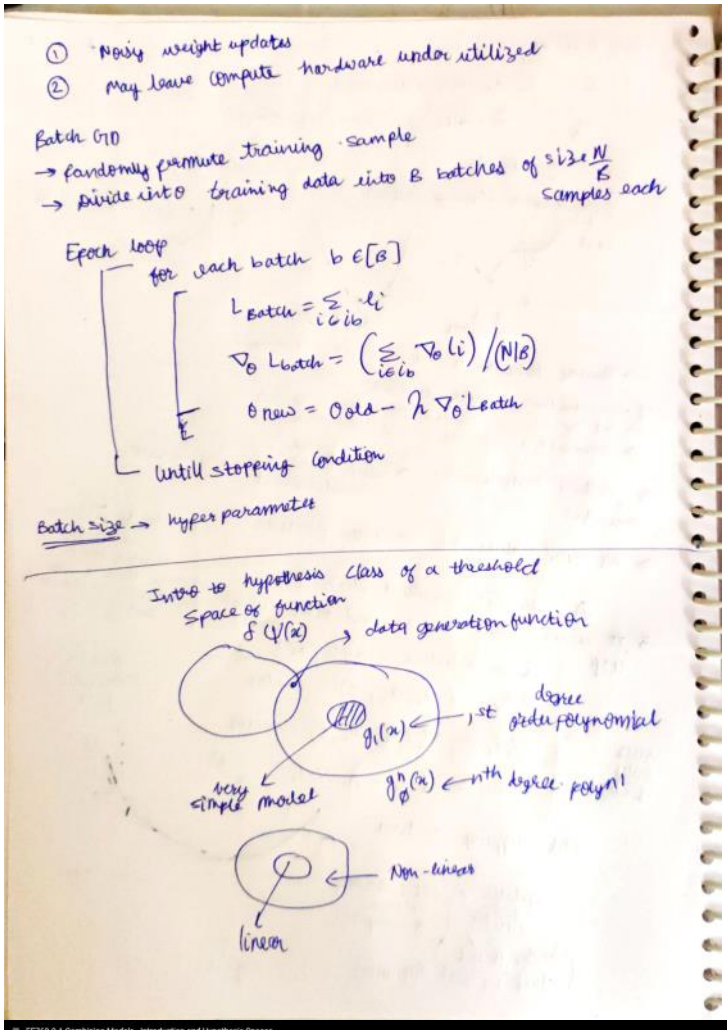
Advanced neural networks

EE769-8-1-1 Advanced Neural Networks - Regression revisited

Learning objectives

- Write NN loss functions for some intermediate ML problems
- Write some properties of natural signals
- List advantages of convolution and pooling layers
- Explain the reason behind exploding and vanishing gradients
- Write how LSTM units solve the exploding gradient problem

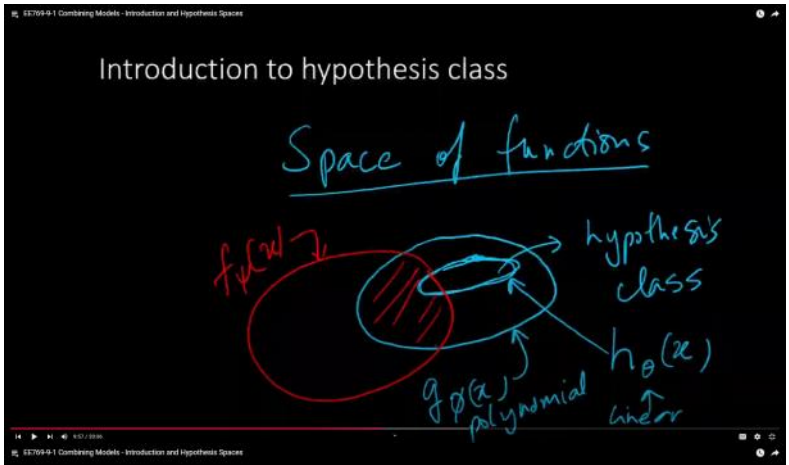
0:42 / 12:40 • Learning Objectives ›

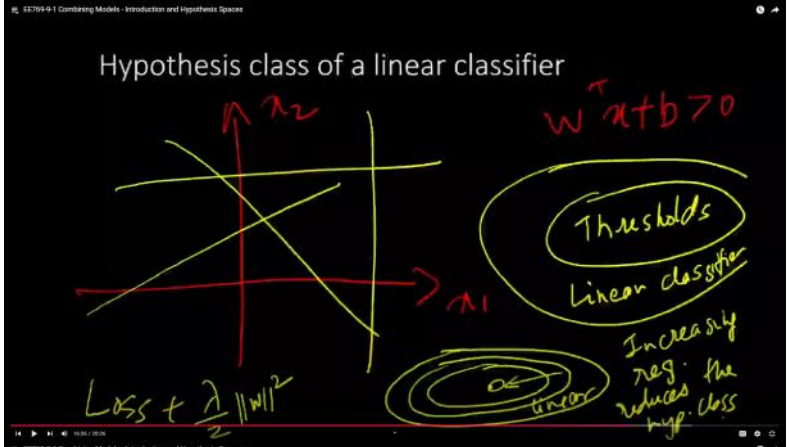


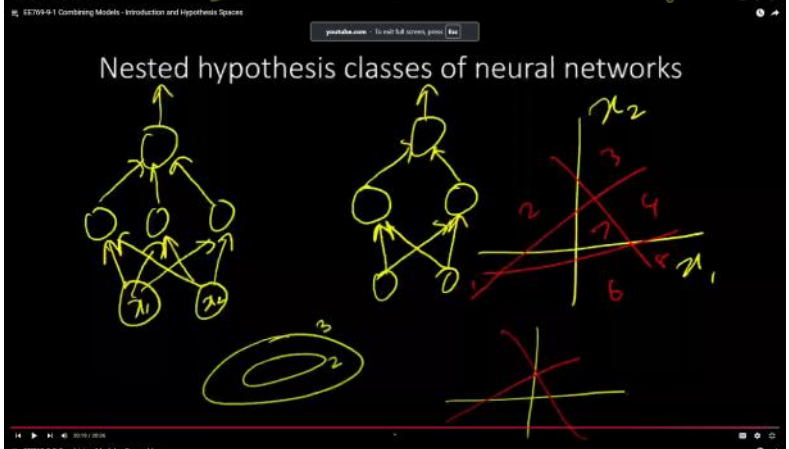
EE769-9-1 Combining Models - Introduction and Hypothesis Spaces

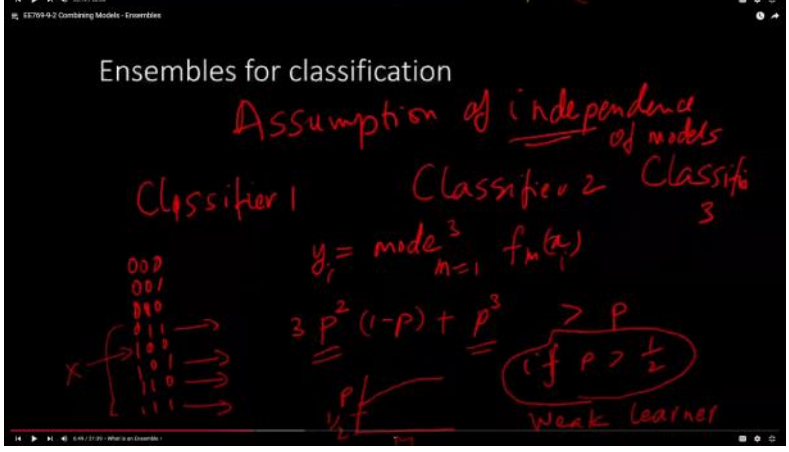
Questions for today's lecture

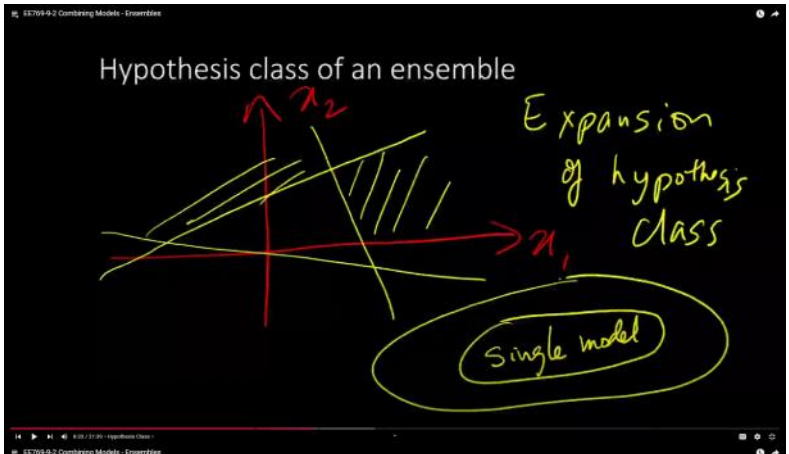
- Why combine models?
  - Multiple models are available, but none of them are perfect
  - Multiple types of features can be extracted for a given pattern
  - Certain complimentary properties exist among different models and different features.
- Issues
  - How many classifiers are needed?
  - What kind of models should be used?
  - What features to be used in each classifier?
  - How to combine results from different classifiers?

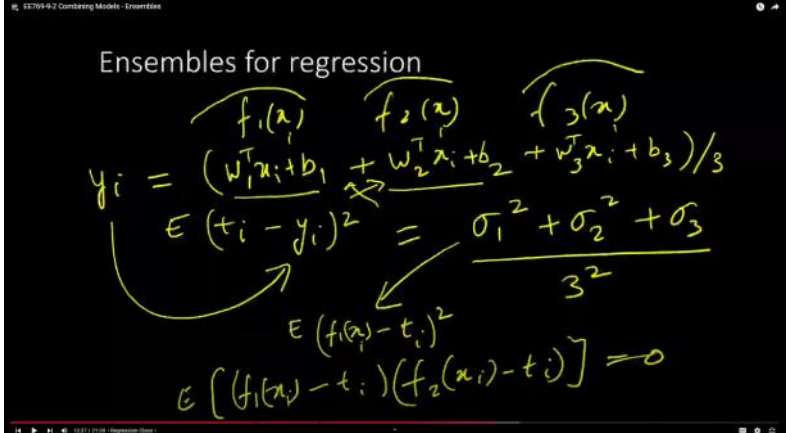

EE769-9-1 Combining Models - Introduction and Hypothesis Spaces
Introduction to hypothesis class
Space of functions
hypothesis class
polynomial
linear

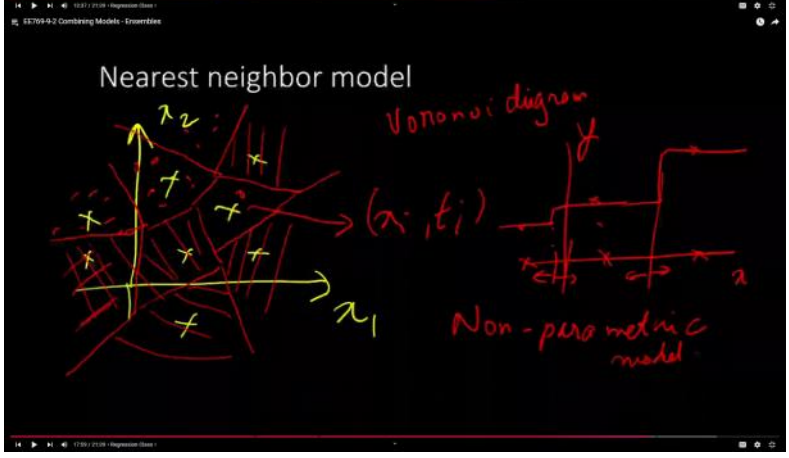

EE769-9-1 Combining Models - Introduction and Hypothesis Spaces
Hypothesis class of a linear classifier
Thresholds
Linear classifier
Increasing reg. the reduces hyp. class
linear


EE769-9-1 Combining Models - Introduction and Hypothesis Spaces
youtube.com - To exit full screen, press Esc
Nested hypothesis classes of neural networks


EE769-9-2 Combining Models - Ensembles
Ensembles for classification
Assumption of independence of models
Classifier 1
Classifier 2
Classifier 3
Weak learner


Hypothesis class of an ensemble
Expansion of hypothesis class
Single model

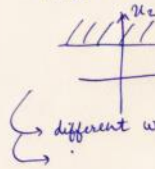

Ensembles for regression


Nearest neighbor model
Voronoi diagram
Non-parametric model

very Simple model

Randomly select input dim j

Find threshold on x_j for model



output 1 (eg. class 1, y_1)

output 0 (eg. class 0, y_0)

Hypothesis class

different ways of combining model

Ensembles for classification — probabilities

h_1	h_2	h_3	y	
0	0	0	0	P^3
0	0	1	0	$P^2(1-P)$
0	1	0	0	$P^2(1-P)$
0	1	1	1	$P(1-P)^2$
1	0	0	0	$P^2(1-P)$
1	0	1	1	$P(1-P)^2$
1	1	0	1	$P(1-P)^2$
1	1	1	1	$(1-P)^3$

Ensembles for regression

h_1 h_2 h_3

σ^2 σ^2 σ^2

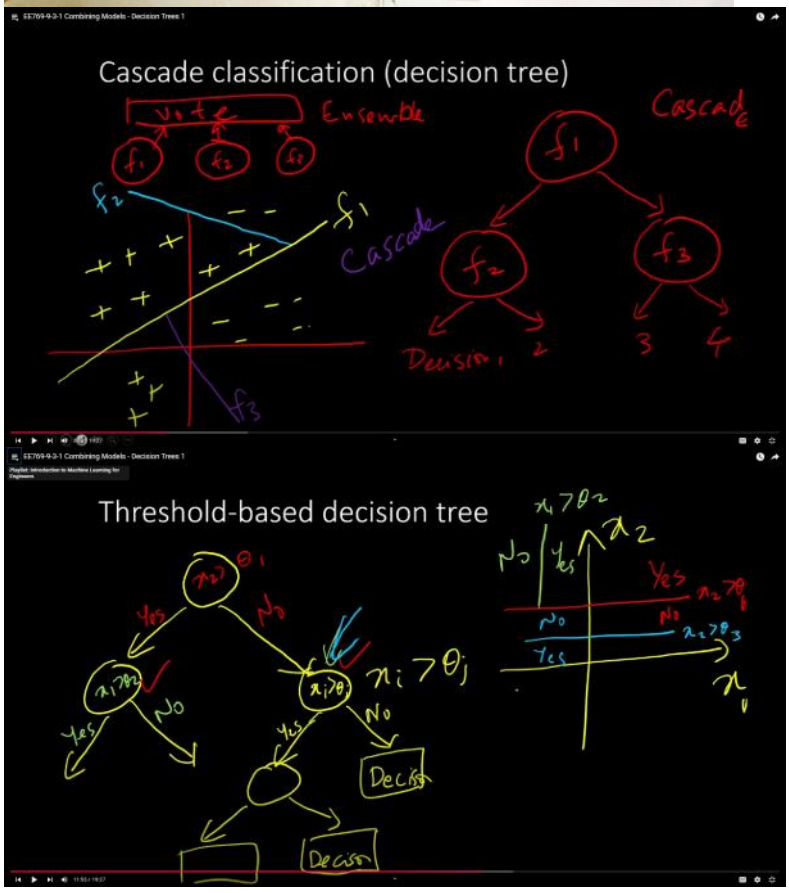
$y = \frac{h_1 + h_2 + h_3}{3}$

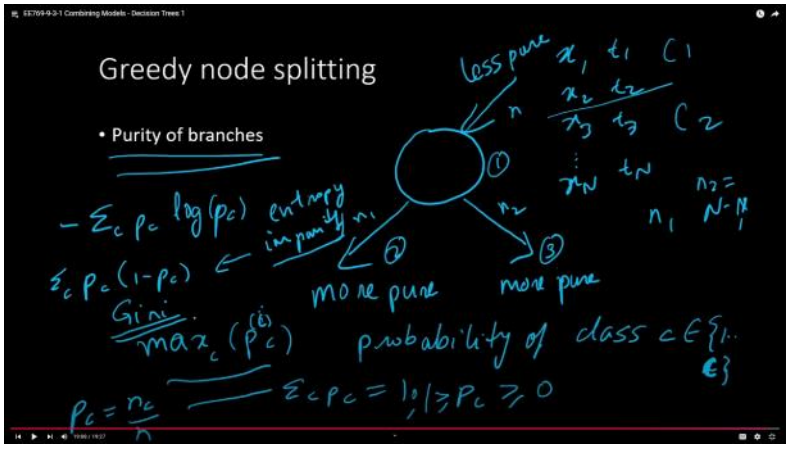
total variance

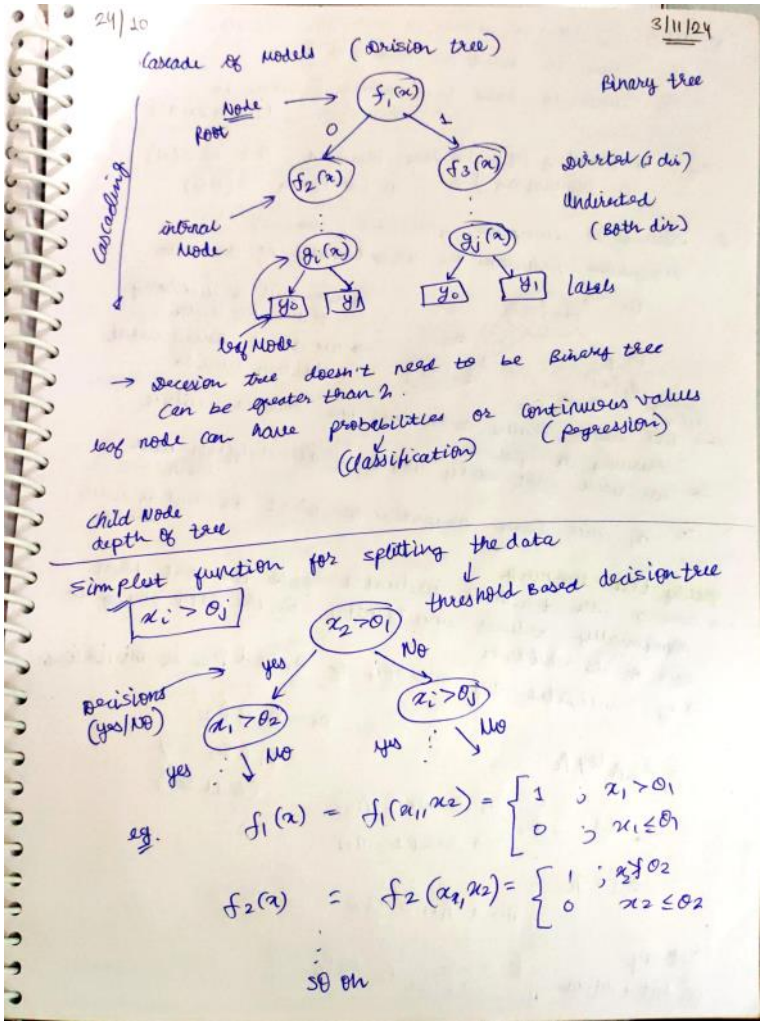
$\frac{3\sigma^2}{9}$ $\frac{\sigma^2}{3}$

$y_i = (w_1^T x_i + b_1 + w_2^T x_i + b_2 + w_3^T x_i + b_3)/3$

$E(t_i - y_i)^2 = \frac{\sigma_1^2 + \sigma_2^2 + \sigma_3^2}{3^2}$



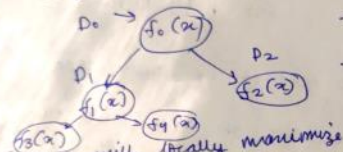

Greedy node splitting
• Purity of branches
less pure
more pure
more pure
entropy
impurity
Gini
probability of class


24/10
3/11/24
Cascade of models (Decision tree)
Binary tree
Node
Root
Directed (1 dir)
Undirected (both dir)
Cascading
Internal Node
labels
Leaf Node
→ Decision tree doesn't need to be Binary tree
Can be greater than 2.
leaf node can have probabilities or continuous values
(Classification)
(regression)
Child Node
depth of tree
simplest function for splitting the data
threshold based decision tree
Decisions (yes/No)
yes
No
eg.
so on

Questions: ① How to choose a var. for decision node?
② How to select a threshold?
③ When to have leaf nodes (when to TERMINATE?)

say N training pts, D dimensions → Max. threshold $N-1 \Rightarrow O(N)$
$D \cdot (N-1) \Rightarrow O(ND)$

Criteria to compare threshold classifier
Maximum reduction in uncertainty of the data



→ we will sequentially train the data
→ we don't know what children will do

→ we will locally minimize the criteria while learning a parent
→ we will not worry about what children will learn.
→ f_1 will learn irrespective of what f_3 and f_4 will learn

greedy tree learning
→ among all threshold evaluated pick the one that maximally reduces uncertainty in the data going to each of its children.
↳ eg. weighted avg. entropy of P_1 and P_2 is minimized.

$P_0 = P_1 \cup P_2$
$P_1 \cap P_2 = \phi$ (Null set)

$-\sum_c P_c \log P_c$

$\mathcal{D}_0$ → label 0 n_{00}
→ label 1 n_{01}

$|\mathcal{D}_0| = n_0$

$n_{00} + n_{01} = n_0$

entropy for distribution $E = -\sum_{c \in \text{labels}} P_c \log P_c$

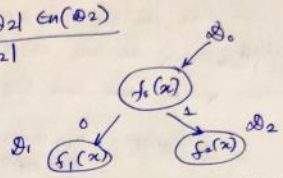
Entropy = $E = -\sum_{c \in \{1,2\}} P_c \log P_c = -\left\{ \frac{n_{00}}{n_0} \log \frac{n_{00}}{n_0} + \frac{n_{01}}{n_0} \log \frac{n_{01}}{n_0} \right\}$

→ trying to increase purity
→ want threshold that will reduce entropy

Basic criteria: weighted avg. entropy ← criteria for (f_0)

$$\frac{|D_1| \, En(D_1) + |D_2| \, En(D_2)}{|D_1| + |D_2|}$$

$|D_1|$ → size of D_1
$En\{D_1\}$ → entropy of D_1

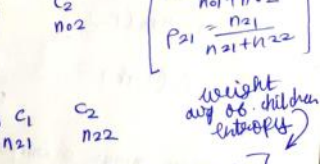


stopping criteria
↳ all the D reaching a branch has a single criteria → label

Node splitting algorithm

greedy Node splitting

Node split (D_i) ← input
If all D_i have same label then return None (stopping criteria)
For all dimensions (n)
  For all pts in D_i (m)
    [split D_i into D_j and D_k acc. to threshold ∈ Dim
    compute weighted avg. entropy of D_j, D_k
Best combo ← $\arg\min_{n,m}(x_n, \theta_n m)$
return Best combo corresponding D_j, D_k



$P_{01} = \frac{n_{01}}{n_{01} + n_{02}}$
$P_{21} = \frac{n_{21}}{n_{21} + n_{22}}$

child C_1 C_2
n_{11} n_{12}

weight avg of children entropy

entropy (P_{11}, P_{02}) $= -\left[\frac{(n_{11}+n_{12}) \, En(P_{11}, P_{12}) + (n_{21}+n_{22}) \, En(P_{21}, P_{22})}{n_{01} + n_{02}} \right]$

No. of leaf Node = n (minimum) ← each sample requires its sole decision. has its own node

* Tree training Algorithm
  recursively split D. until [None / stopping criteria] is returned and make leaf Node
  → Max. possible leaf Node (depth) = O(N)
  (including root) → No. of internal nodes = # leaves − 1
NOTE: deep trees can OVERFIT

n leaves: Min order O (log N) → well structured
Max depth O(N) → very difficult to classify

→ we don't want imbalanced tree
But we can get if our Data is like this

⇒ to reduce overfitting
Regularization : ① stop splitting below a certain entropy
② Prune tree beyond a certain depth

Tree → Combination of threshold classifiers.

Purity or impurity criteria:

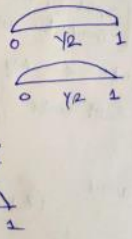
① Entropy = $-\sum_c P_c \log P_c$

② Gini Index = $1 - \sum_c P_c^2$

③ $1 - \max(P_c)$

④ $P_c(1-P_c)$

graph of matrix for different unit of P_c.



[0 entropy means 0 impurity
higher entropy → higher ||]

What are impurity criteria for regression
↳ for regression labels are of continuous values
⇒ Variance of labels (target values)
↳ weighted sum of variances of child is minimized

* Random forest
↳ ensembles of decision tree
→ need to randomise as you'll always end up with same tree.

Randomise:
① Bagging: Training on random subset of samples
- Out of Bag (OOB) samples ⇒ not used in training
② At each node consider only a random subset of features/ dim/ var.

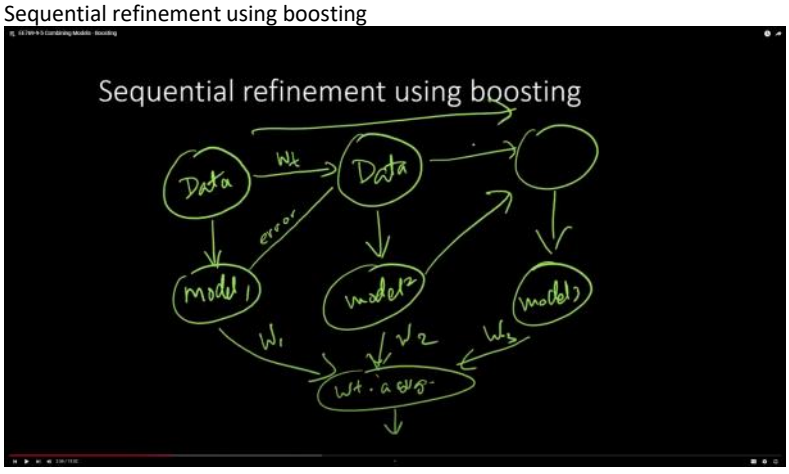
→ In this way, we train different trees
↳ for each sample you take avg. of decisions that was given by each of these trees.

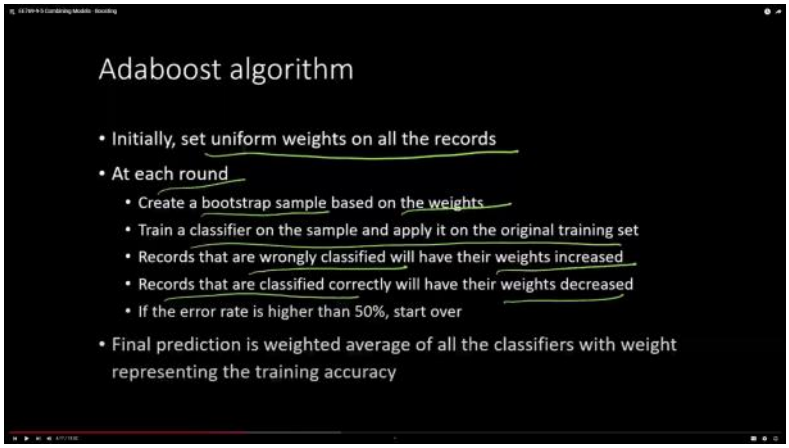
Hyperparams (to consider at each node)
Max depth (% of samples in bag)
Trees

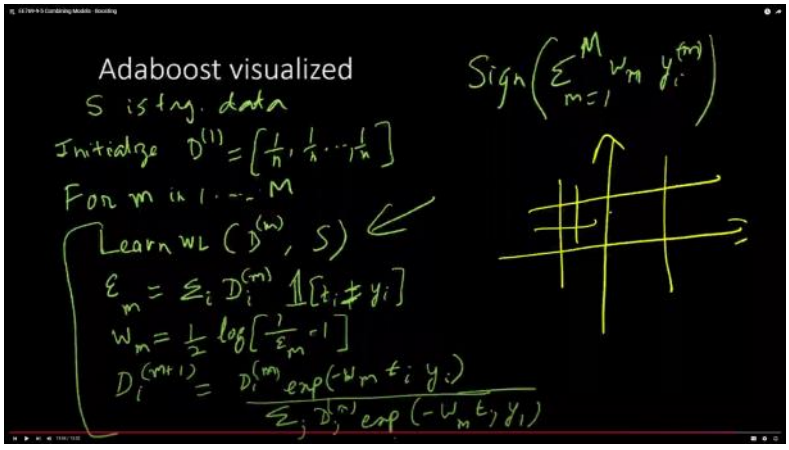
Randomizing models
① Initialization (non convex only)
② Feature subset selection (features/ training part)
③ Training subset selection (Bagging)

Random var-selection
Not going to select or use all D var.,
Instead we use randomly select $d < D$ var.
for testing optimally.

Sequential refinement using boosting







Properties of RF

→ good idea of generalization
for each tree → OOB → testing data
avg OOB label across tree
converge to their generalization error → law of large no.

→ guarantee against OVERFITTING
→ give feature importance
→ handle both discrete and continuous var

Random shuffle is a way to get feature importance

28/10/24

Cascading
Bagging
ensembling
↓
similar
↓
Same model but different subset

3/12/24

Boosting (Ada Boost) Adaptive boosting (way of combining model)
↓
sequentially combination

→ different model, same data (different initialization)

$\sum w_n^1 y_1$ $\sum w_n^2 y_2$ ..... $\sum w_n^M y_M$

n → no. of sample
M → model no.

$y_1(x)$ $y_2(x)$ $y_M(x)$

$Y_M(x) = \text{sign}\left(\sum_m^M \alpha_m y_m(x)\right)$

weights for each sample

Cross entropy loss

$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} \mathbb{1}(t_{ij} \log t_{ij})$

$L_W = -\sum_{i=1}^{N}\sum_{j=1}^{C} w_i\, t_{ij} \log t_{ij}$ ← Sample Bit

(classifying certain sample is more important than other)

At Beginning all weights have same importance
↳ are equal

→ then decrease the importance for that sample that are misclassified and vice-versa.
(second learner to focus more on that features)

Ada Boost Algo

Set all $w_n^{(1)} = \frac{1}{N}$ for $n = 1, \dots N$ (Initialization)

loop over $m = 1$ to M

Compute loss $J_m = \sum_{n=1}^{N} w_n^{(m)} \underbrace{I(y_m(x_n) \neq t_n)}_{\text{indicator function}}$

Normalized error
$\epsilon_m = \frac{J_m}{\sum_{n=1}^{N} w_n^{(m)}}$

Compute model weight
$\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$

update sample rate
$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(x_n) \neq t_n)\}$

exponential function (can use any other function that we wanna mimic)

model importance very large → weight for those where error is large

If error made → Indicator function = 1
No error → otherwise = 0

⇒ Adaboost is minimizing exponential error

$E = \sum_{n=1}^{N} \exp\{-t_n \cdot f_m(x_n)\}$ (overall loss function to minimize)

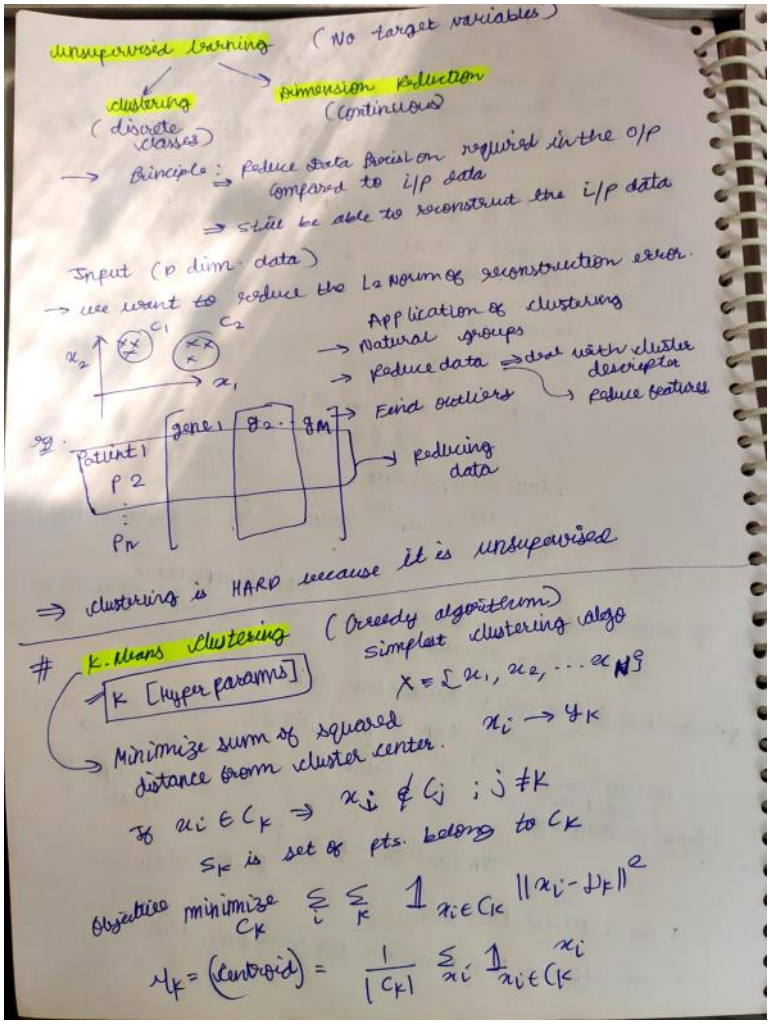
label

$f_m(x) = \frac{1}{2} \sum_{i=1}^{m} \alpha_i y_i(x)$

m^{th} composite model

$t_n y_m(x_n) = 1 - 2 \, I(y_m(x_n) \neq t_n)$

→ for classification only
↳ use different loss functions for regression also



k means clustering