CS217: Artificial Intelligence and Machine Learning (associated lab: CS240)

Nihar Ranjan Sahoo
PhD scholar under Prof. Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

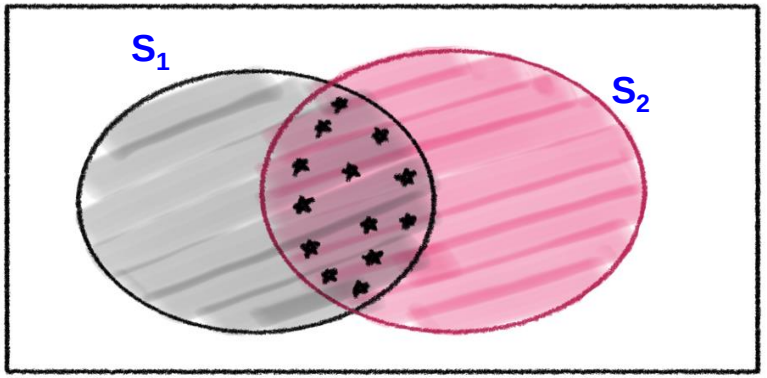
Week10 of 17mar25, Linear, logistic regression, Decision Trees

Main points covered: week9 of 10mar25

SVM using SKLEARN

- Dataset from the UCI Machine Learning Repository
  - https://archive.ics.uci.edu/ml/datasets.php
- Number of instances = 748
- Number of attributes = 4
- **Attributes / Features**:
  - Recency - months since last donation
  - Frequency - total number of donation
  - Monetary - total blood donated in c.c.
  - Time - months since first donation
- **Class label**:
  - A binary variable representing whether he/she donated blood in March 2007
  - 1 stands for donating blood; 0 stands for not donating blood

False Positives, False Negatives, Precision, Recall, F-score



SET S_1 ... OBTAINED
SET S_2 ... ACTUAL
$S_1 \cap S_2$ ... TRUE POSITIVES
$S_1 - (S_1 \cap S_2)$ ... FALSE POSITIVES
$S_2 - (S_1 \cap S_2)$ ... FALSE NEGATIVES
$(S_1 \cup S_2)^c$ ... TRUE NEGATIVES

$$Precision = \frac{|S_1 \bigcap S_2|}{|S_1|}$$

$$Recall = \frac{|S_1 \bigcap S_2|}{|S_2|}$$

Generalized F-score

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R} = \frac{1}{\frac{\beta^2}{(1+\beta^2)R} + \frac{1}{(1+\beta^2)P}}$$

As $\beta \rightarrow 0$, $F_\beta \rightarrow P$ and as $\beta \rightarrow \infty$, $F_\beta \rightarrow R$

Formal definition of HMM (1/2)

- N= #states
- M= #distinct observation symbols
- State transition probability distribution: $A=\{a_{ij}\}$
- The observation symbol probability distribution in state j, $B=b_j(k)$
- The initial state distribution, $\pi=\{\pi_i\}$

Formal definition of HMM (2/2)

- $A=\{a_{ij}\}$
  - $a_{ij}= P(q_{t+1}=S_j \mid q_t=S_i),\ 1<= i,j <=N$
- $B=b_j(k)$
  - $b_j(k)= P(V_k \text{ at } t \mid q_t= S_j),\ 1<= j <=N;$ $1<= k <= M$
- $\Pi=\{\Pi_i\}$
  - $\Pi_i = P(q_1=S_i),\ 1<= i<=N$

Classic problems with respect to HMM

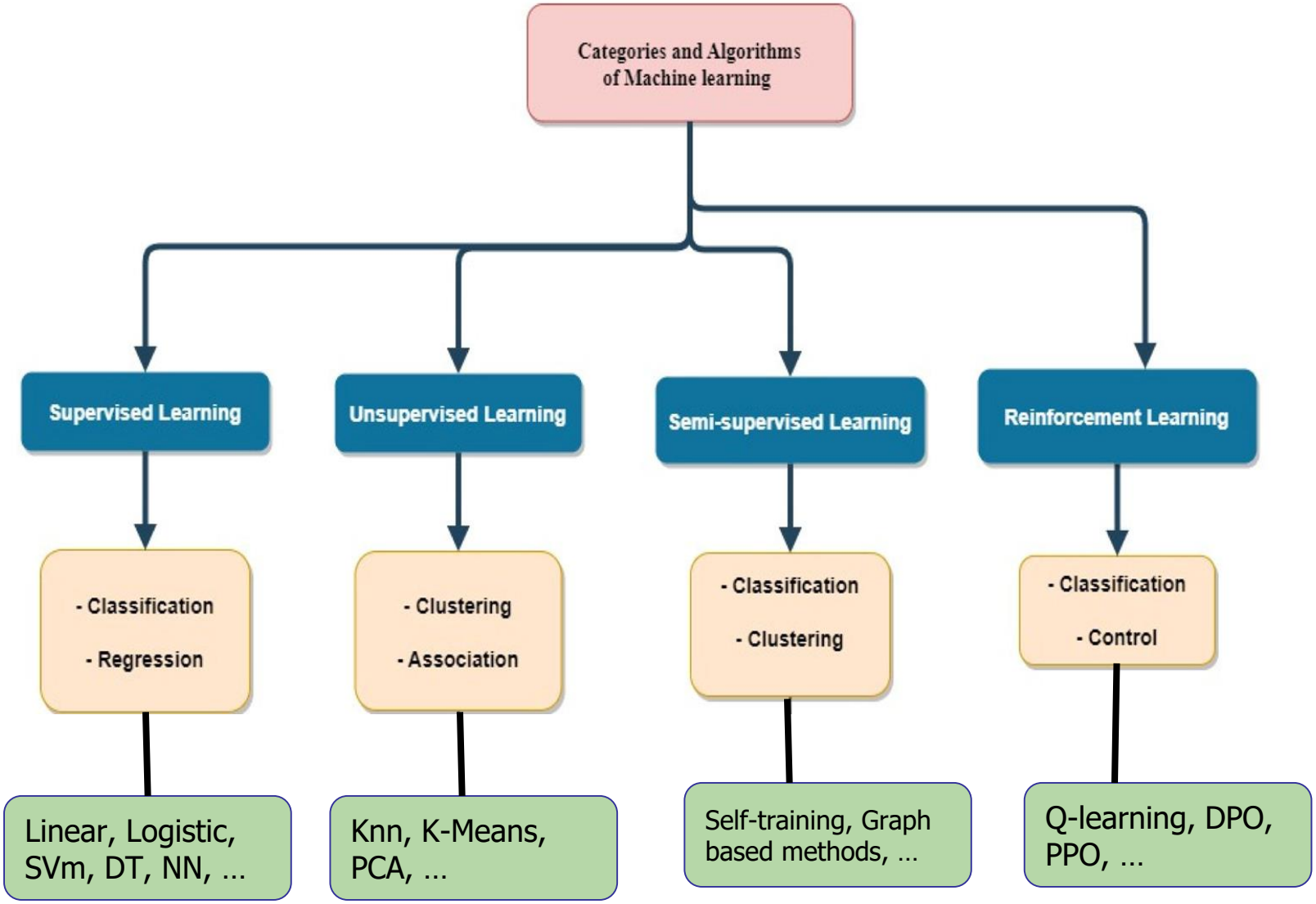
1. Given the observation sequence, find the possible state sequences- **Viterbi**

2. Given the observation sequence, find its probability- **forward/backward** algorithm

3. Given the observation sequence find the HMM prameters.- **Baum-Welch** algorithm

End of Main points

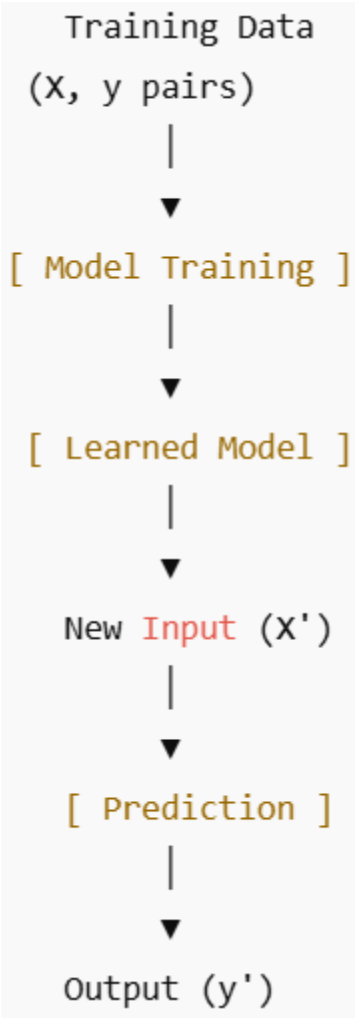
General Machine Learning Tasks (1/2)



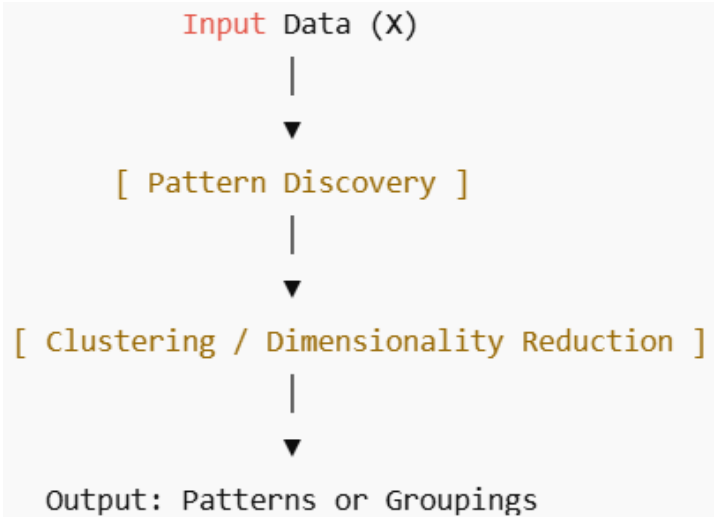
Others: Self-supervised learning, Active learning, Curriculum learning, Life-long learning

General Machine Learning Tasks (2/2)

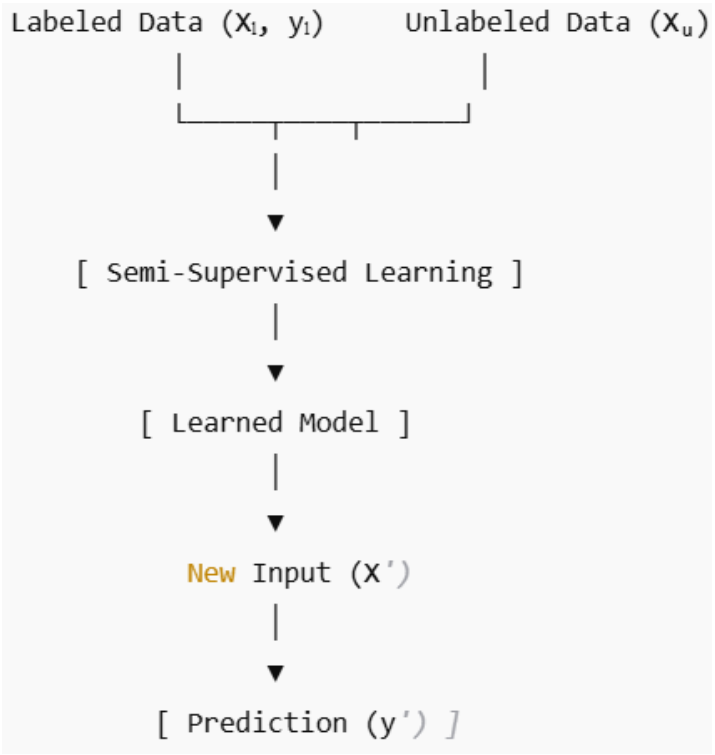
Supervised



Unsupervised



Semi-supervised



Reinforcement

Classification vs. Regression

Classification: a supervised learning task where the objective is to **predict discrete labels or categories** to input data. The model learns to map input features to one of a limited number of classes.

Examples: image recognition, text classification, etc.

Regression: a supervised learning task where the goal is to **predict continuous numeric values**. The model learns the relationship between input features and a continuous output.

Examples: stock market prediction, weather forecasting, etc.

Linear Regression: Intro

Assumes *Linear relationship* between dependent (y) and independent (x) variables.

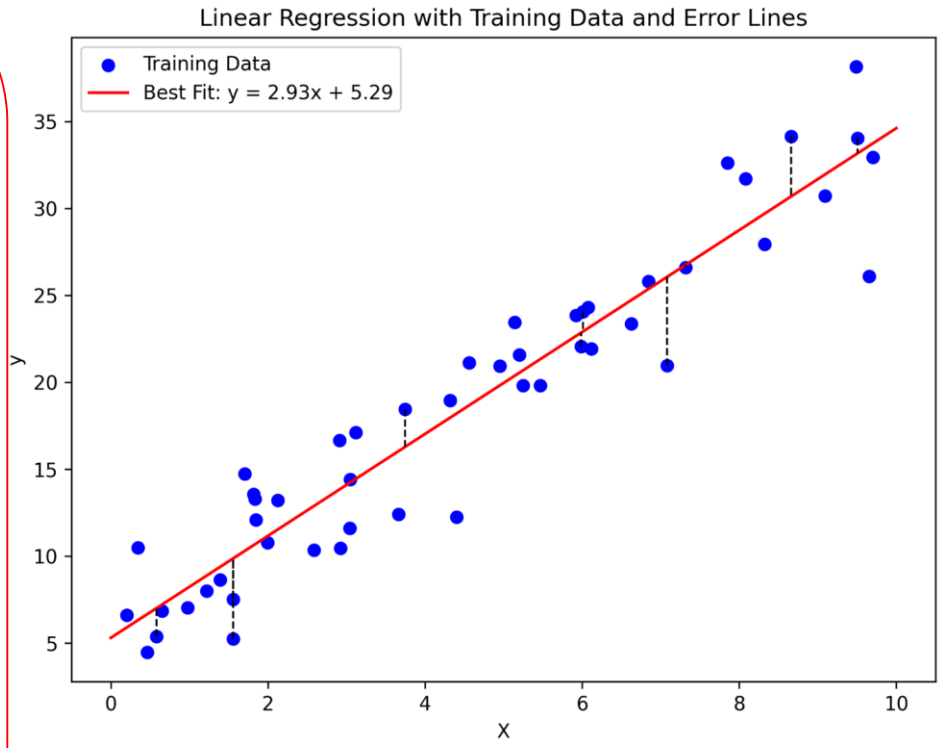
$$\hat{y} = f_\theta(x)$$

Here, *f* is the model with parameter θ that captures the linear relation between input *x* to output *y*.

$$y = mx + C + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$

1. Here, ε is the error (random noise) caused due to experimental design.
2. b is the intercept.
3. m is the slope (indicating the change in *y* for a one-unit change in *x*)
4. y is ground truth
5. $\hat{y}$ is predicted output



Assumes the dependent variable (y) is continuous in nature.

Simple linear regression: $y = b + w_1x + \epsilon$

Multiple linear regression: $y = b + w_1x_1 + w_2x_2 + \cdots + w_px_p + \epsilon$

Linear Regression: **Setup and Error**

Given a dataset of input-output pairs, $\{(x_i, y_i)\}_{i=1}^{N}$.

We train a linear model $f(x; w, b) \Rightarrow w^T x + b$, where w and b are model parameters.

Loss function: Sum Squared Error (SSE) => L(w, b) = $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - (b + wx_i))^2$

Goal: To find the model parameters that minimizes the loss

$$\arg\min_{w,b} L(w,b) = \arg\min_{w,b} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \arg\min_{w,b} \sum_{i=1}^{n} (y_i - (b + wx_i))^2$$

Compactly we car write:

$$\hat{y} = w^\top x, \text{ where } w = \begin{pmatrix} w_1 \\ \vdots \\ w_p \\ b \end{pmatrix} \text{ and } x = \begin{pmatrix} x_1 \\ \vdots \\ x_p \\ 1 \end{pmatrix}$$

Assumptions

1. **Linearity**: The relationship between predictors and response is linear
2. **Independence**: Observations are independent of each other
3. **Homoscedasticity**: Error variance is constant across all levels of predictors
4. **Normality**: Errors are normally distributed
5. **No multicollinearity**: Predictor variables are not highly correlated

Least Square via **Calculus**

We know, L(w, b) = $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - (b + wx_i))^2$

Compute partial derivatives with respect to *b* and *w* and set them to zero:

For *b*: $\frac{\partial L}{\partial b} = -2\sum_{i=1}^{n} (y_i - (b + wx_i)) = 0$ $\frac{\partial L}{\partial w} = -2\sum_{i=1}^{n} x_i (y_i - (b + wx_i)) = 0$

Solving the above equations will give us:

$$w = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$b = \bar{y} - w\bar{x}$$

This is the solution for simple linear regression.

Multivariate Least Square via **Calculus (1/2)**

Considering the loss function in compact form

$$L(w) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - w^\top x_i\right)^2 = \|y - Xw\|_2^2$$

Let $X = \begin{pmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{pmatrix}, y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$. We can rewrite it in terms of matrices:

$$\begin{aligned} L(w) &= \|y - Xw\|_2^2 \\ &= (y - Xw)^\top (y - Xw) \\ &= y^\top y - (Xw)^\top y - y^\top (Xw) + (Xw)^\top (Xw) \\ &= y^\top y - 2y^\top (Xw) + \left(w^\top X^\top\right)(Xw) \\ &= y^\top y - \left(2y^\top X\right) w + w^\top \left(X^\top X\right) w \end{aligned}$$

In this step coalescing is possible due to the scalar thing there. Each term here is a scalar.

Multivariate Least Square via **Calculus (2/2)**

$$\frac{\partial L}{\partial w} = \frac{\partial \left(y^\top y\right)}{\partial w} - \frac{\partial \left(\left(2X^\top y\right)^\top w\right)}{\partial w} + \frac{\partial \left(w^\top \left(X^\top X\right) w\right)}{\partial w} = 0$$

$$0 - 2X^\top y + \left(X^\top X + \left(X^\top X\right)^\top\right) w = 0$$

$$-2X^\top y + 2\left(X^\top X\right) w = 0$$

$$2\left(X^\top X\right) w = 2X^\top y \quad \text{(Normal equations)}$$

If X is full-rank,

$$w = \left(X^\top X\right)^{-1} X^\top y$$

This also called the **Closed-form solution** to Linear Regression or Ordinary Least Square regression.

If this is the closed-form solution then $w = \left(X^\top X\right)^{-1} X^\top y$ should be the global minimum. To prove this, we need to show that ***L(w)*** **is convex**.

Convexity of Sum Square Error

Theorem: Hessian of a convex function is Positive Semi-definite.

Recall: $\frac{\partial L}{\partial w} = -2X^\top y + 2\left(X^\top X\right) w$

$$\frac{\partial^2 L}{\partial w \partial w^\top} = \frac{\partial}{\partial w}\left(\frac{\partial L}{\partial w}\right)$$

$$= \frac{\partial}{\partial w}\left(-2X^\top y + 2\left(X^\top X\right) w\right)$$

$$= \frac{\partial}{\partial w}\left(-2X^\top y\right) + \frac{\partial}{\partial w}\left(2\left(X^\top X\right) w\right)$$

$$= 0 + 2\left(X^\top X\right)^\top$$

$$= 2X^\top X$$

$$\succcurlyeq 0$$

Since the Hessian is always positive semi-definite, the loss function is convex.
So any critical point is a local and global minimum, so the critical point $w = \left(X^\top X\right)^{-1} X^\top y$ minimizes the loss function.

Scenarios of OLS (based on Pseudoinverse)

The matrix $(X^TX)^{-1}X^T$ is known as the **pseudoinverse** of X, and is usually denoted as $X^\dagger$.

Case 1:

- When # of data points (N) = # of input dimensions (p), can fit data perfectly.
  - In other words, $w = min_w \|y - Xw\|_2^2$, so finding the optimal w amounts to solving the linear system of equations $Xw = y$.
  - If X is full-rank, $w = X^{-1}y$.

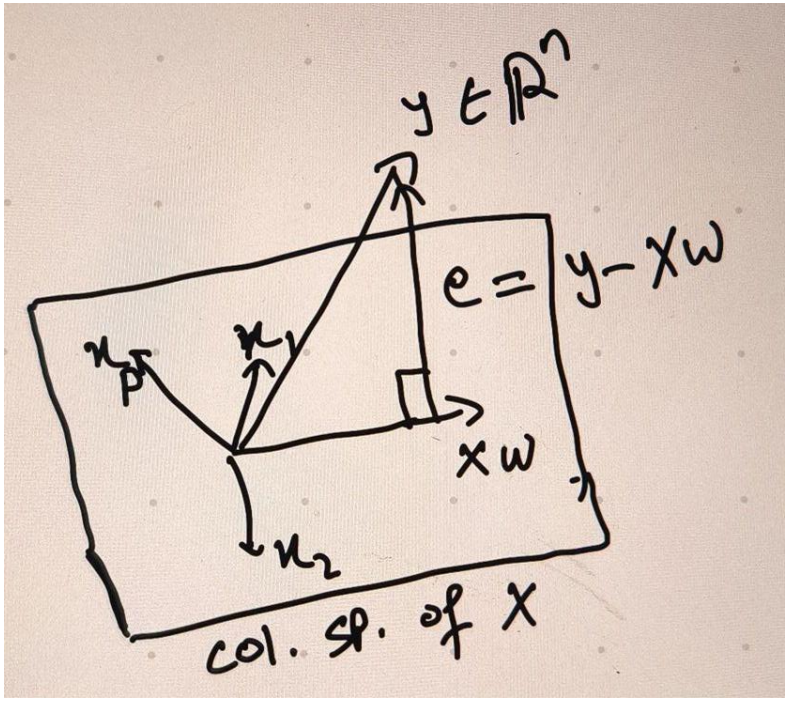
Case 2:

- When # of data points (N) > # of input dimensions (p), in general cannot fit data perfectly. (In real world scenarios, this is more practical)
  - As shown earlier, $w = (X^TX)^{-1}X^Ty = X^\dagger y$. If we compare these formulas $w = X^{-1}y$ vs. $w = X^\dagger y$, $X^\dagger$ can be seen as a generalization of X^{-1} to the case where data cannot be fit perfectly.
  - $X^\dagger$ can be interpreted as an operator that finds the best possible solution to a linear system of equations that has no solution

If full rank then directly calculate the weight otherwise calculate first the pseudo inverse then calculate the weight

Least Square via **Linear Algebra**

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_p \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$



In real world scenarios, n >> p

=> no solution generally

$y = Xw \Rightarrow$ (linear comb. of col of X)

col. sp (X) $\subseteq R^n$, dim(col(X)) $\leq p$

$x_1^T e = 0, x_2^T e = 0, \cdots, x_p^T e = 0 \Rightarrow X^T e = 0$

$\Rightarrow x^T(y - Xw) = 0 \Rightarrow X^T X w = X^T y$

$\Rightarrow$ ***$w = (X^T X)^{-1} X^T y$*** (Same closed-form solution)

If Rank(X) = p.

$\Rightarrow$ Rank($X^T X$) = Rank(X) = p

$\Rightarrow$ Then Pseudo inverse can give solution

Least Square (Closed Form): **Concerns**

1. High computational cost

   Computing $(X^T X)^{-1}$ requires matrix inversion, which generally has a computational cost on the order of $O(p^3)$, where p is the number of features.

1. Numerical instability

   If the features are highly correlated (i.e., multicollinearity), $X^T X$ can be nearly singular. This makes the inversion numerically unstable, leading to unreliable coefficient estimates.

1. May require large memory

   When dealing with large-scale data, storing the entire X matrix and computing $X^T X$ can exceed available memory.

Least Square via **Gradient Descent**

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

This set of parameters minimizes the ¨sum-squared-error" or "mean-squared-error"

Remember, $SSE = \|Y - Xw\|_2^2$

$$\nabla L = \begin{bmatrix} \frac{\partial}{\partial w_1} L \\ \frac{\partial}{\partial w_2} L \\ \vdots \\ \frac{\partial}{\partial w_{p+1}} L \end{bmatrix} = -2X^\top y + 2(X^\top X)w$$

The gradient descent algorithm: Set w^0 to a random vector in $\mathbb{R}^{p+1}$ for an initial guess and choose a learning rate parameter γ. Compute $X^\top y$ (an element of $\mathbb{R}^{p+1}$ and $X^\top X$ (a $(k+1) \times (k+1)$ matrix). Shouldn't it be a (p+1) * (p+1) matrix.

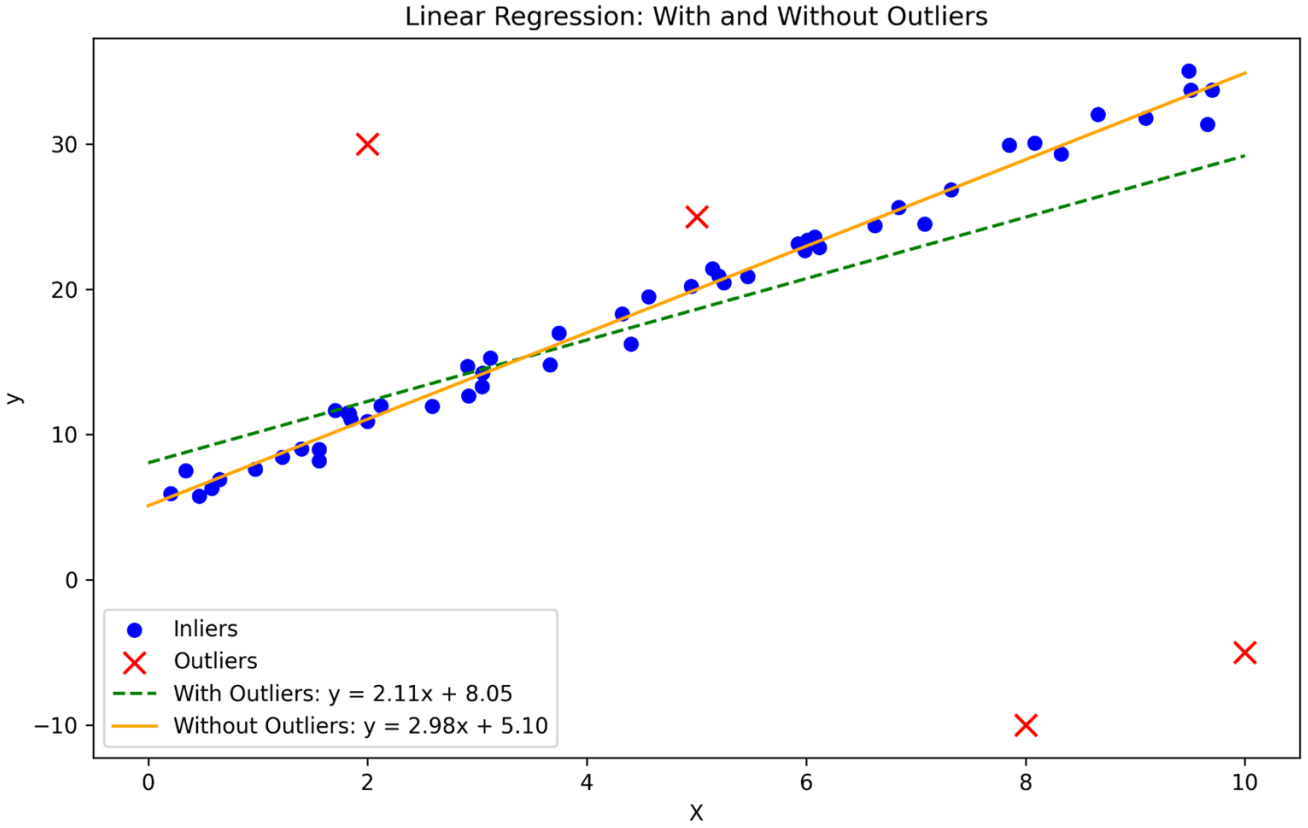
Iteratively compute

$$w^{(t+1)} = w^{(t)} - \gamma \left(-2X^\top y + 2(X^\top X)w^{(t)}\right)$$

until the entries a stopping condition is met. For example, stop if the mean squared error $\left\|y - Xw^{(k)}\right\|^2$ changes by less than some tolerance on each iteration, or the entries of $w^{(k)}$ change by less than some tolerance.

Notice that this algorithm does not need computation of $(X^\top X)^{-1}$.

Least Square: **Handling Outliers (1/3)**



$$\text{Linear regression: } L(w) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - w^\top x_i\right)^2 = \|y - Xw\|_2^2$$

$$\text{Ridge regression: } L(w) = \sum_{i=1}^{n} \left(y_i - w^\top x_i\right)^2 + \lambda \|w\|_2^2 = \|y - Xw\|_2^2 + \lambda \|w\|_2^2$$

Least Square: **Handling Outliers (2/3)**

$$L(w) = \|y - Xw\|_2^2 + \lambda \|w\|_2^2$$
$$= (y - Xw)^\top (y - Xw' + \lambda w^\top w$$
$$= y^\top y - (Xw)^\top y + y^\top (Xw) + (Xw)^\top (Xw) + \lambda w^\top w$$
$$= y^\top y - \left(2y^\top X\right) w + w^\top \left(X^\top X\right) w + \lambda w^\top w$$

Derivative transposing the matrix.

$$\frac{\partial L}{\partial w} = \frac{\partial \left(y^\top y\right)}{\partial w} - \frac{\partial \left(\left(2X^\top y\right)^\top w\right)}{\partial w} + \frac{\partial \left(w^\top \left(X^\top X\right) w\right)}{\partial w} + \frac{\partial \left(\lambda w^\top w\right)}{\partial w} = 0$$

$$0 - 2X^\top y + \left(X^\top X + \left(X^\top X\right)^\top\right) w + \lambda \left(I + I^\top\right) w = 0$$

$$-2X^\top y + 2\left(X^\top X\right) w + 2\lambda I w = 0$$

$$-2X^\top y + 2\left(X^\top X + \lambda I\right) w = 0$$

$$2\left(X^\top X + \lambda I\right) w = 2X^\top y$$

$$\left(X^\top X + \lambda I\right) w = X^\top y$$

$$w = \left(X^\top X + \lambda I\right)^{-1} X^\top y$$

Least Square: **Handling Outliers (3/3)**

$$\text{Recall: } \frac{\partial L}{\partial w} = -2X^\top y + 2\left(X^\top X + \lambda I\right) w$$

$$\frac{\partial^2 L}{\partial w \partial w^\top} = \frac{\partial}{\partial w}\left(\frac{\partial L}{\partial w}\right)$$

$$= \frac{\partial}{\partial w}\left(-2X^\top y + 2\left(X^\top X + \lambda I\right) w\right)$$

$$= \frac{\partial}{\partial w}\left(-2X^\top y\right) + \frac{\partial}{\partial w}\left(2\left(X^\top X + \lambda I\right) w\right)$$

$$= 0 + 2\left(X^\top X + \lambda I\right)^\top$$

$$= 2\left(X^\top X + \lambda I\right)$$

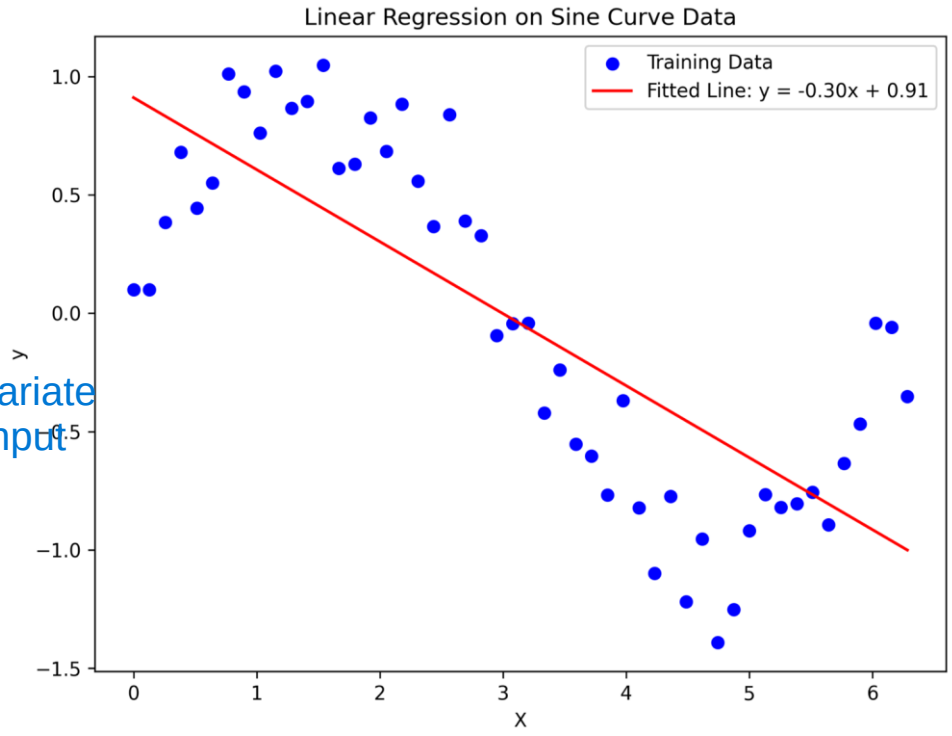
Claim: $X^\top X + \lambda I \succ 0$

Proof: $w^\top \left(X^\top X + \lambda I\right) w = w^\top \left(X^\top X\right) w + w^\top (\lambda I) w = (Xw)^\top (Xw) + \lambda w^\top w = \|Xw\|_2^2 + \lambda \|w\|_2^2$

For any $w \neq \vec{0}$, $\|w\|_2^2 > 0$

Since $\|Xw\|_2^2 \geq 0$ and $\lambda > 0$, $\|Xw\|_2^2 + \lambda \|w\|_2^2 > 0 \quad \forall w \neq \vec{0}$

Least Square: **Concerns**



1. Linearity assumption

   What about non-linear relation between *X* and *y*?

- Kernel Trick Same as making "x" a multivariate random variable with many input features.

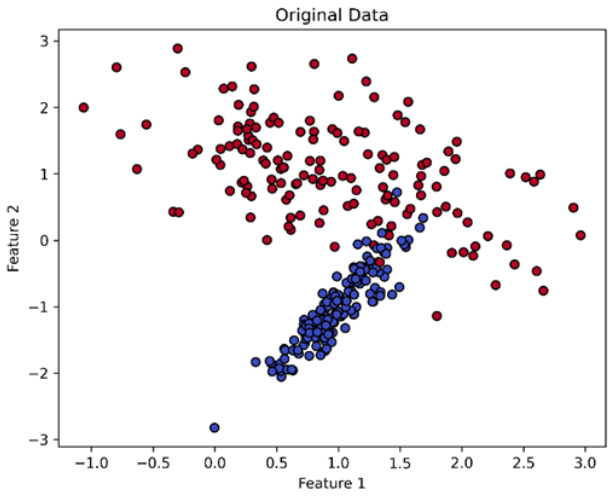
1. Sensitive to large outliers

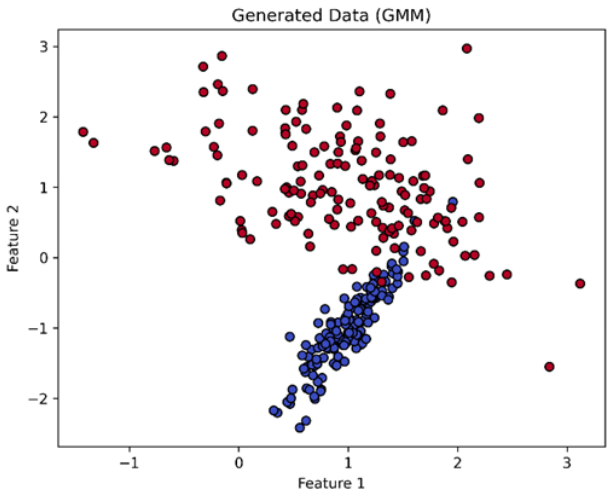
   It does not detect outliers, rather try to reduce the effect of outliers through regularization.

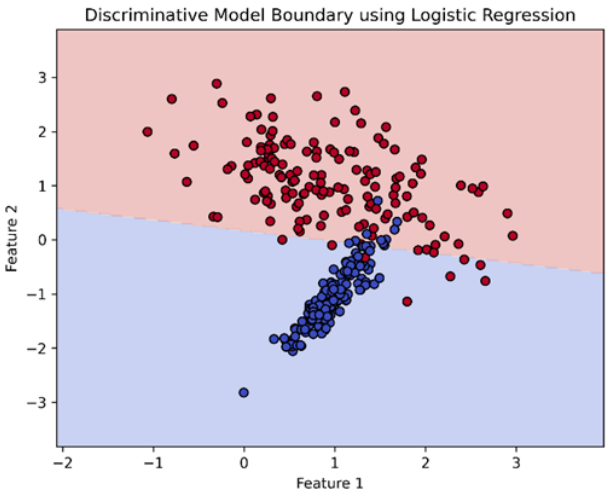
1. Less interpretable

Least Square: **Code Walk through**

Discriminative *vs.* **Generative** Model (1/2)







Generative Methods (***G***):

Generative models aim to capture how the data is generated by modeling the joint probability distribution p(x,y). In doing so, they "explain" both the features and the labels, which allows them not only to classify but also to generate new data samples.

A generative model is defined by: $p(x,y|\vartheta)=p(x|y,\vartheta)\,p(y|\pi)$

- p(y) is the prior distribution over the labels.
- p(x|y) is the likelihood, describing how the features x are generated given y.
- θ: the parameters of ***G***
- π: the class prior

Discriminative *vs.* **Generative** Model (2/2)

Discriminative Methods (***D***):

Discriminative models focus directly on modeling the decision boundary between classes. Instead of modeling the joint distribution, they learn the conditional probability p(y|x) or a direct mapping from x to y. This often leads to better classification performance as the model is optimized solely for the prediction task.

- P(X,Y): the true joint distribution of inputs and outputs
- θ: the parameters of discriminative model
- ***D***: directly model the conditional probability distribution P(Y|X; θ).

Aspect	**Generative Methods**	**Discriminative Methods**
Modeling Objective	Models the joint distribution $p(x, y)$	Models the conditional distribution $p(y\|x)$ or a decision function
Data Generation	Can generate new samples x given a label y	Focus solely on predicting y given x
Assumptions	Often require assumptions about data generation (e.g., conditional independence in Naive Bayes)	Fewer assumptions about how data is generated
Classification	Uses Bayes' rule: $p(y\|x) \propto p(x\|y)p(y)$	Directly estimates $p(y\|x)$ or a decision function
Flexibility vs. Complexity	Can be more flexible (e.g., handling missing data) but may need more data to estimate the full joint distribution	Often simpler for classification and may generalize better for prediction tasks

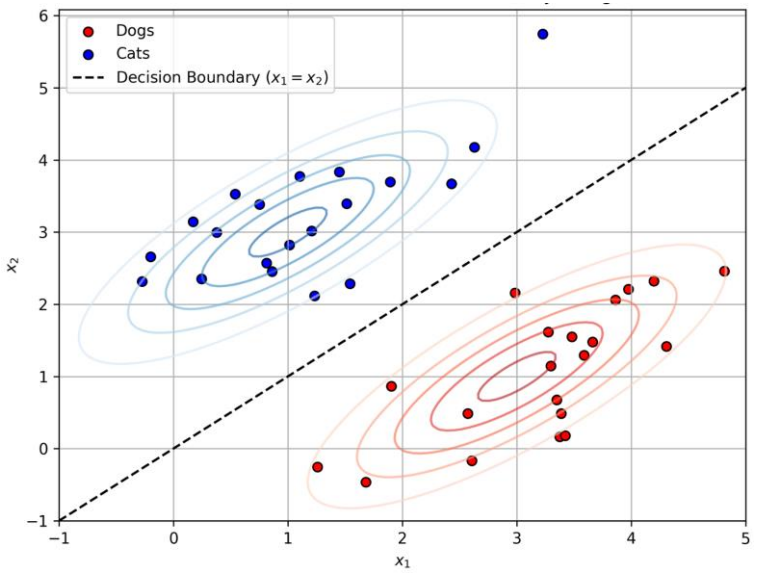
Formal representation of ***D***

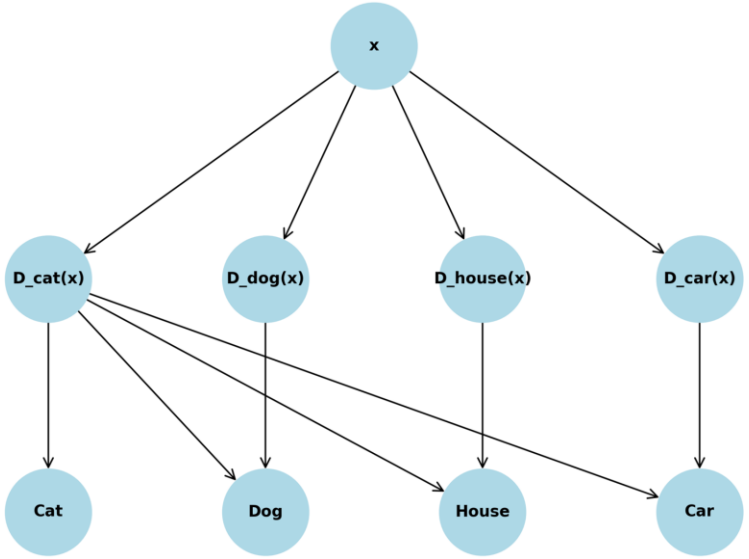
Discriminative method labelling or classifying into classes.

$Y = \{y_1, y_2, \ldots, y_k\}$

$D_i(X) = P(y_i|X)$

$D_i(X) > D_j(X)$, for all j != i => predict y_i





Parameter Estimation Methods

These are different methods used to understand how models learn from data. In other words, how to learn model parameters given the input data.

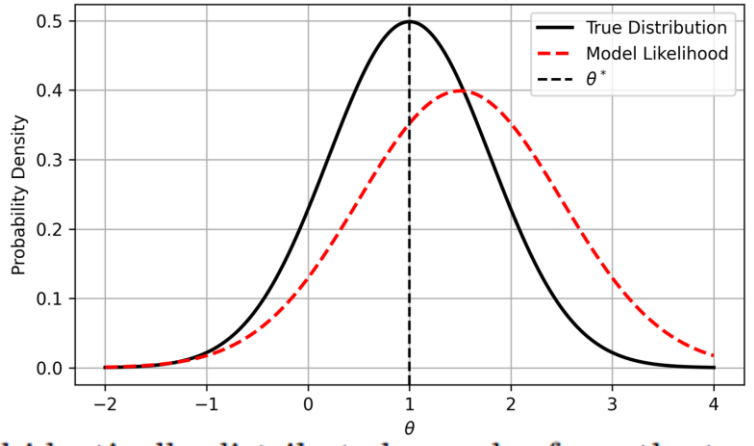
Criterion	MLE	MAP	LS	MoM	Bayesian	EM
Objective	Maximize $p(x\|\theta)$	Maximize $p(\theta\|x)$	Minimize squared error	Match sample and theoretical moments	Compute full posterior $p(\theta\|x)$	Maximize expected complete log-likelihood
Incorporation of Prior	No	Yes	No	No	Yes	(Yes, via MAP-EM variant)
Computational Complexity	Moderate to high	Similar to MLE + prior handling	Low (for linear models)	Low	Often high (integration or sampling)	Moderate, iterative
Applicability	Well-specified likelihood models	When prior knowledge is available	Regression and related tasks	When moments are easily computed	Full uncertainty quantification needed	Models with latent variables/missing data
Concerns	Sensitive to model misspecification	Dependent on chosen prior	Sensitive to outliers	May be biased in small samples	Computationally intensive	May converge to local optima
Examples	Logistic Regression, HMM (via MLE)	Regularized Regression, MAP estimates in Bayesian networks	Linear Regression	Moment matching for Gaussian or Beta distributions	Bayesian Linear Regression, MCMC, Variational Inference	Gaussian Mixture Models, Hidden Markov Models

we know somehow which model it is just want to suit better parameter values

Prior knowledge or assumptions about the data before doing any analysis

MLE - Maximum Likelihood Estimation

$x = \{x_1, x_2, \ldots, x_n\}$ are i.i.ds from true distribution Px(x)
We want to learn a model with parameter θ such that the difference between Px(x) and P(x|θ) is minimized.
P(x|θ) => likelihood of the model



Given a training dataset $\mathcal{D} = (x_i, y_i)_{i=1}^n$ of n independent and identically distributed samples from the true distribution $P(X, Y)$, the likelihood function for a discriminative model is:

$$L(\theta \mid \mathcal{D}) = \prod_{i=1}^{n} P(y_i \mid x_i; \theta)$$

Taking the logarithm for computational convenience:

$$\log L(\theta \mid \mathcal{D}) = \sum_{i=1}^{n} \log P(y_i \mid x_i; \theta)$$

The maximum likelihood estimator is then:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y_i \mid x_i; \theta)$$

MAP - Example 1 MLE example it is.

$$X_1, \ldots, X_n \sim \text{i. i. d Bernoulli}(p)$$

- Samples: $x_1, x_2, \ldots, x_n$

  $x_i = 0$ or 1
- Likelihood: $L(x_1, \ldots, x_n) = \prod_{i=1}^{n} f_X(x_i)$
  - $f_X(x_i) = p$ if $x_i = 1$, or $f_X(x_i) = 1 - p$ if $x_i = 0$
  - Let w denote the number of 1 s in the sample

$$L(x_1, \ldots, x_n) = p^w (1-p)^{n-w}$$

Maximise function: "Differentiate and equate to zero"

$$\rightarrow \text{ Maximizing } L \leftrightarrow \text{ Maximizing } \log L$$

$$h(p) = \log L = w \log p + (n - \omega) \log(1 - p)$$

$$\frac{dh(p)}{dp} = \omega \times 1/p + (n - w)\frac{1}{1-p}(-1) = 0$$

$$w(1-p) = (n - \omega)p$$

$$p = \omega/n$$

MAP - Maximum A Posteriori Estimate

MAP estimation incorporates prior knowledge about parameters through Bayes' rule:

$$P(\theta|X,Y) \propto P(Y|X,\theta)\cdot P(\theta)$$

In MAP (Maximum A Posteriori) estimation, you assume that the model is known, and the goal is to choose the parameter values that best align with both the observed data and your prior beliefs.

Till now, $P(X \mid \theta) \leftarrow$ any likelihood. [not included prior yet — non-bayesian]

$$P(\theta \mid x) = \frac{P(x \mid \theta)P(\theta)}{P(x)}$$

$$L_{\text{map}} = \log P(\theta \mid x) = \log P(x \mid \theta) + \log P(\theta) + C$$

$$\hat{\theta}_{\text{map}} = \underset{\theta}{\text{argmax}}\ L_{\text{MLE}}\ (\theta) + \log P(\theta)$$

MLE tries to fit the data as best as possible.

MAP balances the data fit with prior knowledge

Linear : MLE :: Ridge : MAP

$$y = \omega^\top x + \epsilon \sim N\left(0, \sigma^2\right)$$

for MLE,

$$P(y \mid x, \omega) = N\left(y; \omega x, \sigma^2\right)$$
$$= \frac{1}{\sigma\sqrt{2\pi}} 2^{-(y-\omega^\top x)^2/2\sigma^2}$$

for MAP,

Terms not depending on the 'w' we effectively can ignore them because they obviously not pop up in the optimization process because they aren't the function of w.

$$P(\omega \mid x, y) = \frac{P(x, y, \omega)}{P(x, y)} \propto P(y \mid x, \omega) P(x, \omega)$$
$$\propto P(y \mid x, \omega) P(\omega) P(x \mid \omega)$$
$$\log p(\omega \mid x, y) = \log P(y \mid x, \omega) + \log P(\omega)$$

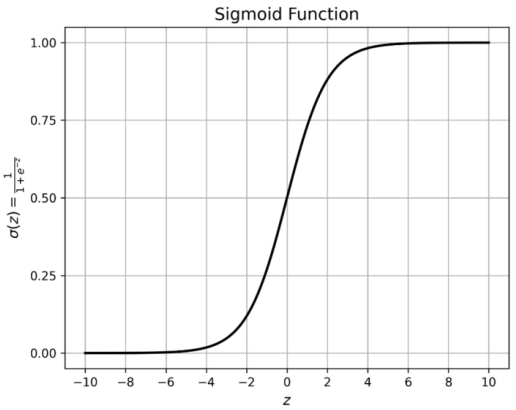
Given, $P(\omega) = N\left(\omega; 0, \frac{1}{\lambda} I\right) \rightarrow$ (uniform prior, condition for linear regression)

$$\omega^* = \underset{\omega}{\operatorname{argmax}} \log p(y \mid x, \omega) + \log p(\omega)$$

$$= \underset{\omega}{\operatorname{argmax}} \log \frac{1}{\sigma\sqrt{2\pi}} + \log\left(e^{-(y-\omega^\top x)^2/2\sigma^2}\right) + \log\left(e^{-(\omega-0)^\top \Sigma^{-1} \omega}\right)$$

$$L(\omega) = -\frac{y - \left(\omega^\top x\right)^2}{2\sigma^2} - \lambda\|\omega\|^2$$
$$\omega^* = \operatorname{argmax} L(\omega) \quad \text{Let, } \sigma = 1$$
$$= \arg\min_{\omega} \frac{\left(y - \omega^\top x\right)^2}{2} + \lambda\|\omega\|^2$$

Logistic Regression: **Intro**



- Supervised algorithm for classification
- Based on “odds ratio”
- Odds = P / (1-P) => that is why sigmoid is used

$$\text{Data: } \{(x_i, y_i)\}_{i=1}^{n}, \quad x_i \in \mathbb{R}^d$$

$$P\left(y_i = 1 \mid x_i, \omega\right) = \text{Benn}\left(\sigma\left(\omega^\top x\right)\right) = P_i$$

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

$$\text{MLE:} \quad \theta^* = \underset{\theta}{\text{argmax}}\, \underset{x,y}{E}\left[\log(P(x,y) \mid \theta)\right]$$

$$= \underset{\theta}{\text{argmax}}\, \underset{x,y}{E}\left[\log \frac{P(x,y,\theta)P(\theta,x)}{P(\theta)P(\theta,x)}\right]$$

In logistic regression (a discriminative model), we directly model P(y|x,w) and treat x as given data, so P(x|w) does not affect the parameter estimation. It’s effectively a constant with respect to w

$$= \underset{\theta}{\text{argmax}}\, \underset{x,y}{E}\left[\log P(y \mid x,\theta), P(x \mid \theta)\right]$$

P(x|theta) doesn't affect the point where it maximizes the terms.

$$\theta^* = \underset{\theta}{\text{argmax}}\, \underset{x,y}{E}\left[\log p(y \mid x,\theta)\right]$$

$$\theta^* = \underset{\theta}{\text{argmax}} \frac{1}{N} \sum_{i=1}^{n} \log P\left(y_i \mid x_i, \theta\right)$$

Logistic Regression: **MLE**

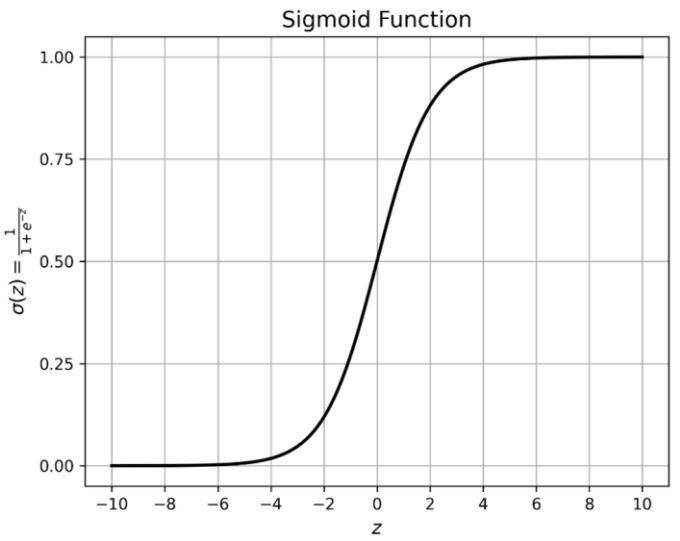
Standard Logistic Regression is designed for Binary classification.

$$\text{Bern}\left(y_i; p_i\right) = p_i^{y_i}\left(1 - p_i\right)^{1-y_i}$$

For logistic,

$$\omega^* = \underset{\omega}{\text{argmax}} \frac{1}{N} \sum_{i=1}^{N} \log P_i^{y_i}\left(1 - P_i\right)^{1-y_i}$$

$$= \underset{\omega}{\text{argmin}} - \frac{1}{N} \sum_{i=1}^{N} \left\{y_i \log P_i + \left(1 - y_i\right) \log\left(1 - P_i\right)\right\}$$



Logistic Regression: **Gradient Descent**

$$\omega^{(t+1)} = \omega^{(t)} - \eta_t \nabla_\omega L\left(\omega^{(t)}\right)$$

$$P_i = \sigma\left(z_i\right)$$

$$z_i = \omega^\top x_i$$

$$\frac{\partial L}{\partial \omega^j} = \frac{\partial L_i}{\partial P_i} \cdot \frac{\partial P_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial \omega^j}$$

$$\frac{\partial L_i}{\partial P_i} = \frac{-1}{N}\left[\frac{y_i}{P_i} - \frac{1-y_i}{1-P_i}\right]$$

$$\frac{\partial P}{\partial z} = \sigma'(z) = \sigma(z)(1-\sigma(z))$$

$$\frac{\partial z_i}{\partial \omega^j} = \frac{\partial}{\partial \omega^j}\left[\omega_0 + \omega^1 x_i^1 + \omega^2 x_2^2 + \cdots\right]$$

$$= x_i^j$$

$$\frac{\partial L}{\partial \omega^j} = (1) \times (2) \times (3)$$

$$= -\frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i}{P_i} - \frac{1-y_i}{1-P_i}\right] P_i\left(1-P_i\right) x_i^j$$

Logistic Regression: **Gradient Descent**

$$= -\frac{1}{N}\sum_{i=1}^{N}\left[y_i\left(1-p_i\right)-\left(1-y_i\right)p_i\right]x_i^j$$

$$\frac{\partial Li}{\partial \omega^j} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i-p_i\right)x_i^j$$

$$\nabla_\omega L\left(\omega^{(t)}\right) = \begin{bmatrix} \frac{\partial L}{\partial \omega^0} \\ \frac{\partial L}{\partial \omega'} \\ \vdots \\ \frac{\partial L}{\partial \omega^\alpha} \end{bmatrix} = -\frac{1}{N}x^\top\left(y-\sigma(x\omega)\right)$$

G.D: $\omega^{(t+1)} = \omega^{(t)} - \eta\frac{1}{N}x^\top\left(\sigma(\omega)-y\right)$

Inferenc:?

Logistic Regression: **Multiclass**

$$C = \{C_1, L_2, \ldots C_k\}$$

$$C_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\sum p_i = 1, \quad p_i > 0$$

$$p_i \propto e^{\omega_i^\top x}$$

$$\Rightarrow P_i = \frac{e^{\omega_i^\top x}}{\sum_{j=1}^{k} e^{\omega_j^\top x}}$$

$$P\left(y = c_i \mid x_i, \omega\right) = \text{ muttinomial } \left\{ y_i; \frac{e^{\omega_1^\top x}}{s}, \frac{e^{\omega_2^\top x}}{S}, \cdots, \frac{e^{\omega_k^+ x}}{S} \right\}$$

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \underset{x,y}{E} \left[-\log(p(y \mid x, \omega)) \right]$$

Have to do the derivative step of it.

$$= \underset{\omega}{\operatorname{argmin}} \sum_{i=1}^{w} \sum_{j=1}^{k} y_i^j \log p_i^j$$

$$\nabla L = -\frac{1}{N} x^\top (y - \sigma(x\omega))$$

Logistic Regression: **Code Walk through**

Decision Tree

Match	Pitch Type	Host	Batting First	Winner
M1	Spin-friendly	India	India	India
M2	Pace-friendly	Australia	Australia	Australia
M3	Balanced	India	Australia	India
M4	Spin-friendly	Australia	India	Australia
M5	Pace-friendly	India	Australia	Australia
M6	Spin-friendly	India	Australia	India
M7	Balanced	Australia	India	India
M8	Pace-friendly	Australia	India	Australia
M9	Spin-friendly	India	India	India
M10	Balanced	Australia	Australia	Australia

Decision Tree

$$P(x=1) = P$$

$$P(x=0) = 1 - P$$

$$H(x) = \sum_{z=1}^{k} -P(x=i)\log P(x=i)$$

$$= -\int_{-\infty}^{\infty} P(x)\log P(x)dx$$

$$= E_{x\sim P(x)}[-\log P(x)]$$

$$H(\text{Bern}\,n(P)) = -P\log P - (1-P)\log(1-P)$$

$$\frac{\partial H}{\partial P} = \frac{-P}{P} - \log P + \log(1-P)$$

$$H = 0$$

$$\log\left(\frac{1-P)}{P}\right) = 0$$

$$\Rightarrow 1 - P = P$$

$$\Rightarrow P = 1/2$$