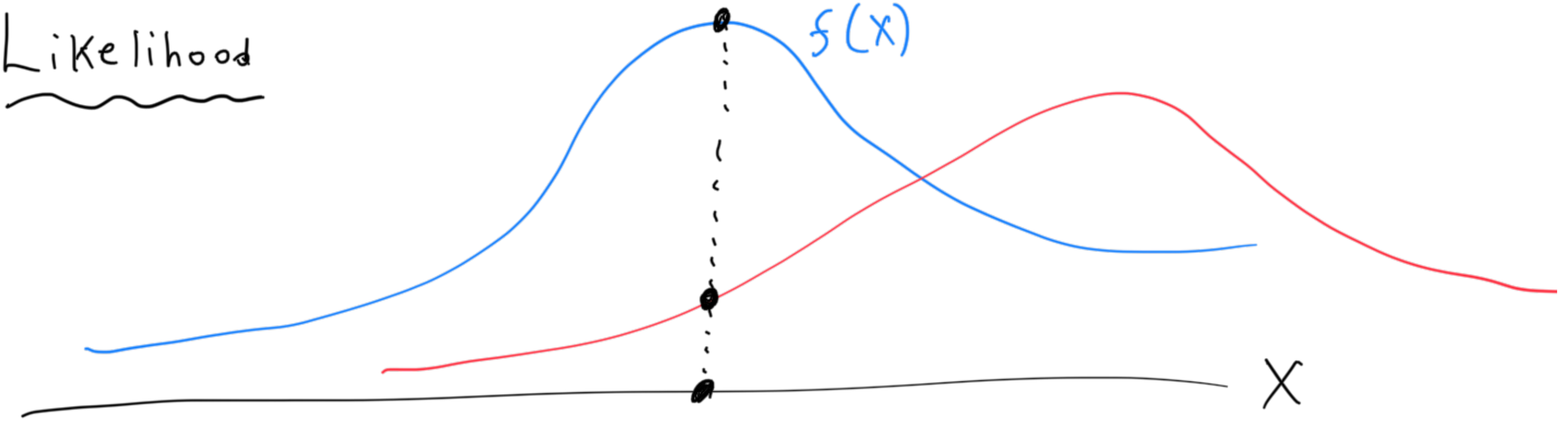
Day 10: Maximum Likelihood Estimation and Non-Parametric Learning Intro

Likelihood



$$L(\theta|X) = f(x_1|\theta) \times \dots \times f(x_n|\theta)$$

$$\log L(\theta|X) = \log f(x_1|\theta) + \dots + \log f(x_n|\theta)$$

$\log(f(x))$ increases/decreases same as $f(x)$

Maximum Likelihood Estimation (MLE)

$$\frac{\partial L(\theta|X)}{\partial \theta} = 0$$

Solve for θ

} regular optimization problem

In practice, people solve:

$$\frac{\partial \log L(\theta|X)}{\partial \theta} = 0 \quad \text{for} \quad \theta$$

Because:

* many fractions multiplied will underflow
* $\underset{\theta}{\operatorname{argmax}} f(x) = \underset{\theta}{\operatorname{argmax}} \log f(x)$

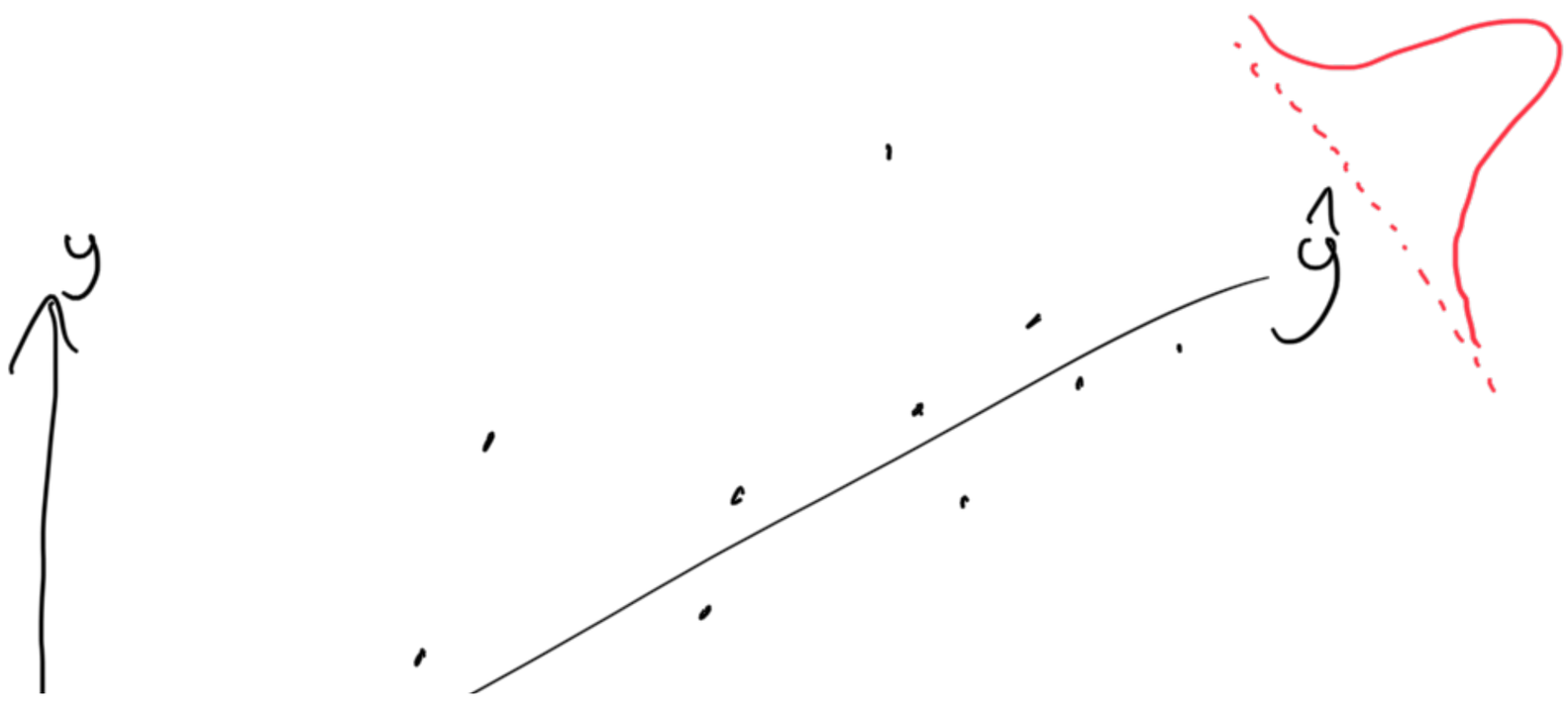
$$\underset{\theta}{\operatorname{argmin}}\, f(x) = \underset{\theta}{\operatorname{argmin}}\, \log f(x)$$

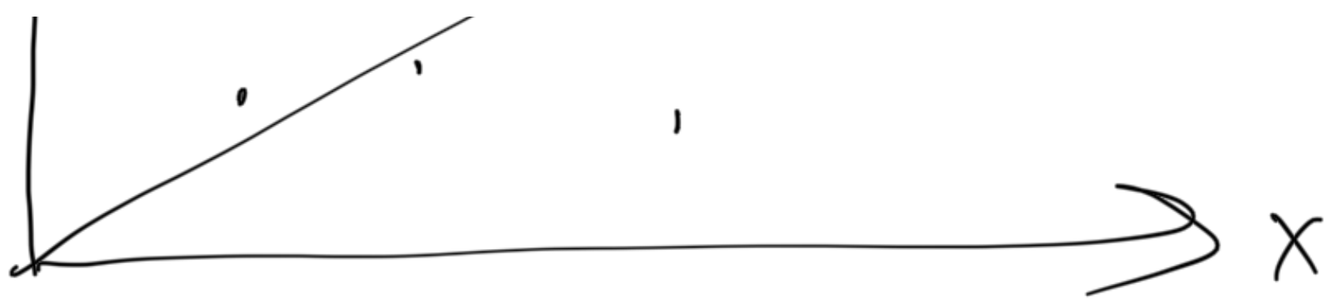
($\underset{\theta}{\operatorname{argmin}}\, f(x)$ = "value of θ which minimizes $f(x)$")

Big Takeaway for Linear Regression:

Maximizing likelihood = Minimizing MSE

Proof:





$$\hat{y} = \theta_0 + \theta_1 X_1 + \dots + \theta_n X_n$$

$$y = \hat{y} + \epsilon$$

the true data is the predicted line plus some error

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

assume the error is normally distributed with mean 0 and variance σ^2

So:

$$y \sim N(\hat{y}, \sigma^2)$$

$L(\hat{y}, \sigma^2 \mid y)$

log-likelihood =

$$\left(\frac{-(y-\hat{y})^2}{} \right)$$

$$\sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\hat{y})^2}{2\sigma^2}}\right)$$

$$= n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{(y-\hat{y})^2}{2\sigma^2}$$

Constant in terms of θ

$\operatorname{argmin} f(x) = \operatorname{argmin} C \cdot f(x)$

Solving $\dfrac{\partial(\text{log likelihood})}{\partial\theta} = 0$ for θ

is equivalent to

$$= \left(\sum_{i=1}^{n} (y-\hat{y})^2\right)$$

Solving $\frac{\partial(-\sum_{i=1}^{n}(\cdots))}{\partial\theta} = 0$ for θ

"maximum likelihood estimate for θ"

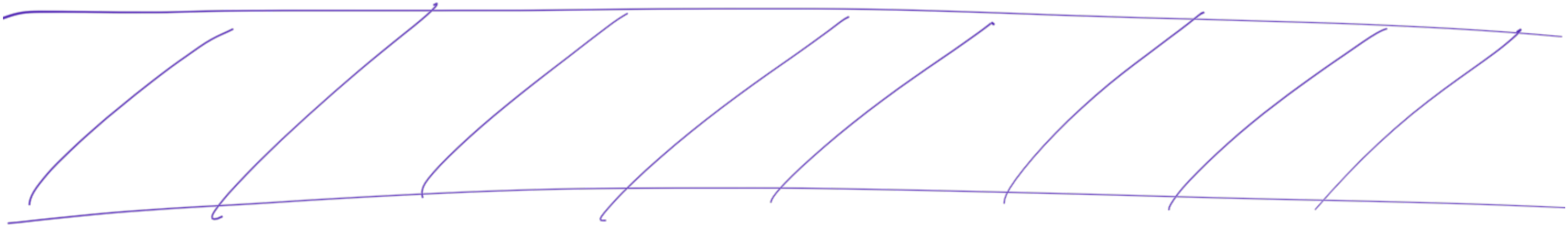
$$\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}} \left[-\sum_{i=1}^{n} (y - \hat{y})^2 \right]$$

$$= \underset{\theta}{\operatorname{argmin}} \left[\sum_{i=1}^{n} (y - \hat{y})^2 \right]$$

Sum of Square errors (SSE)

$$MSE = \underbrace{\frac{1}{n}}_{\text{constant}} SSE$$

$$\underset{\theta}{\operatorname{argmin}}\ MSE = \underset{\theta}{\operatorname{argmin}}\ SSE$$



$$L(p) = \prod_{i=1}^{n} \underbrace{p^{y_i}(1-p)^{(1-y_i)}}_{\text{Bernoulli PMF}}$$

$$\log L(p) = \log\left(\prod_{i=1}^{n} p^{y_i}(1-p)^{(1-y_i)}\right)$$

$$= \sum^{n} \log\left(p^{y_i}(1-p)^{(1-y_i)}\right)$$

$i=1$

$$= \sum_{i=1}^{n} \left(y_i \log p_i + (1 - y_i) \log (1 - p_i) \right)$$

$$\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}} \left(\dots \right)$$

$$= \underset{\theta}{\operatorname{argmin}} \left[- \sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log (1 - p_i) \right]$$

Cross-Entropy Loss:

$$\frac{1}{n} \sum_{i=1}^{n} \left(y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \right)$$

Loss Function for Logistic Regression

Multi-Class Logistic Regression

Generalization of cross-entropy loss to multi class:

$$-\sum_{i=1}^{C} y_i \log \hat{y}_i \qquad C = \#\text{ of classes}$$

For multi-class logistic regression:

$$y = \text{Softmax}\left(\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n\right)$$

$$\frac{e^{x_i}}{\left(\right)}$$

$$\text{Softmax}(X)_i = \frac{e^{X_i}}{\sum_{j=1}^{C} e^{X_j}}$$

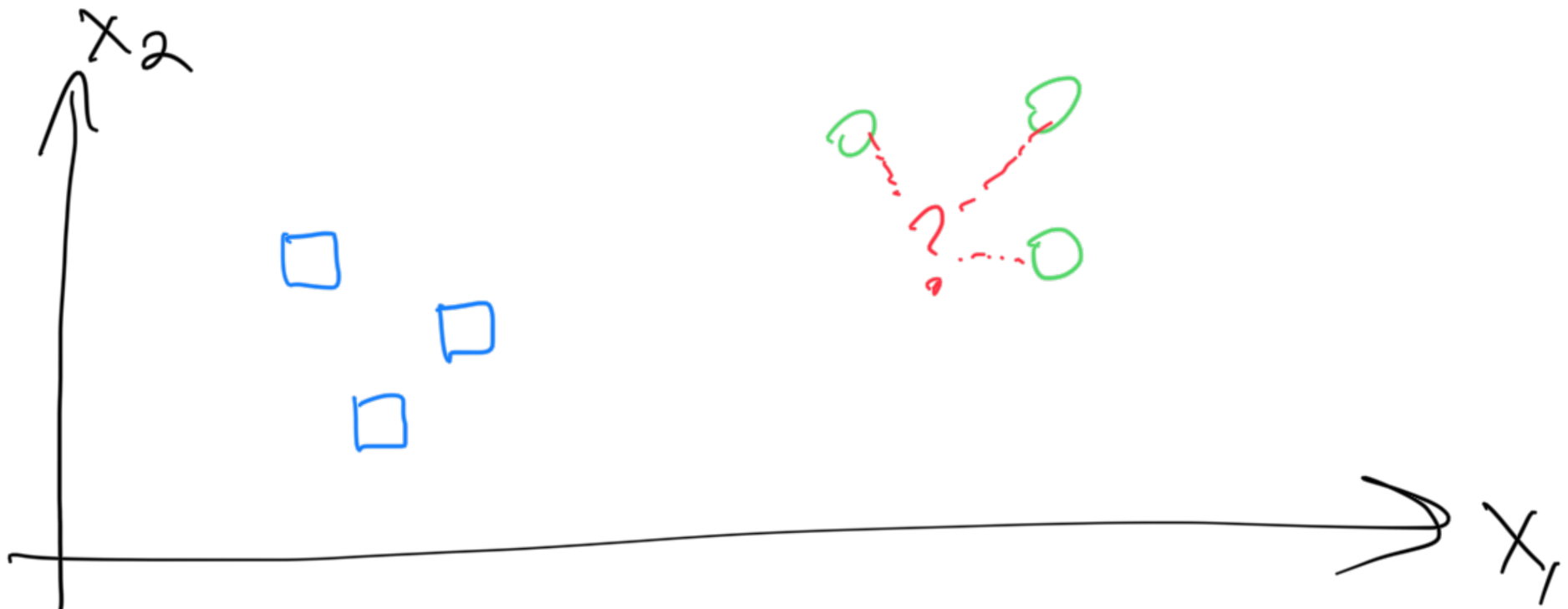
Example:

$$[-3, 4, 9]$$

$$\text{Softmax}(-3) = \frac{e^{-3}}{(e^{-3} + e^{4} + e^{9})}$$

$$\text{Softmax}(4) = \frac{e^{4}}{(e^{-3} + e^{4} + e^{9})}$$

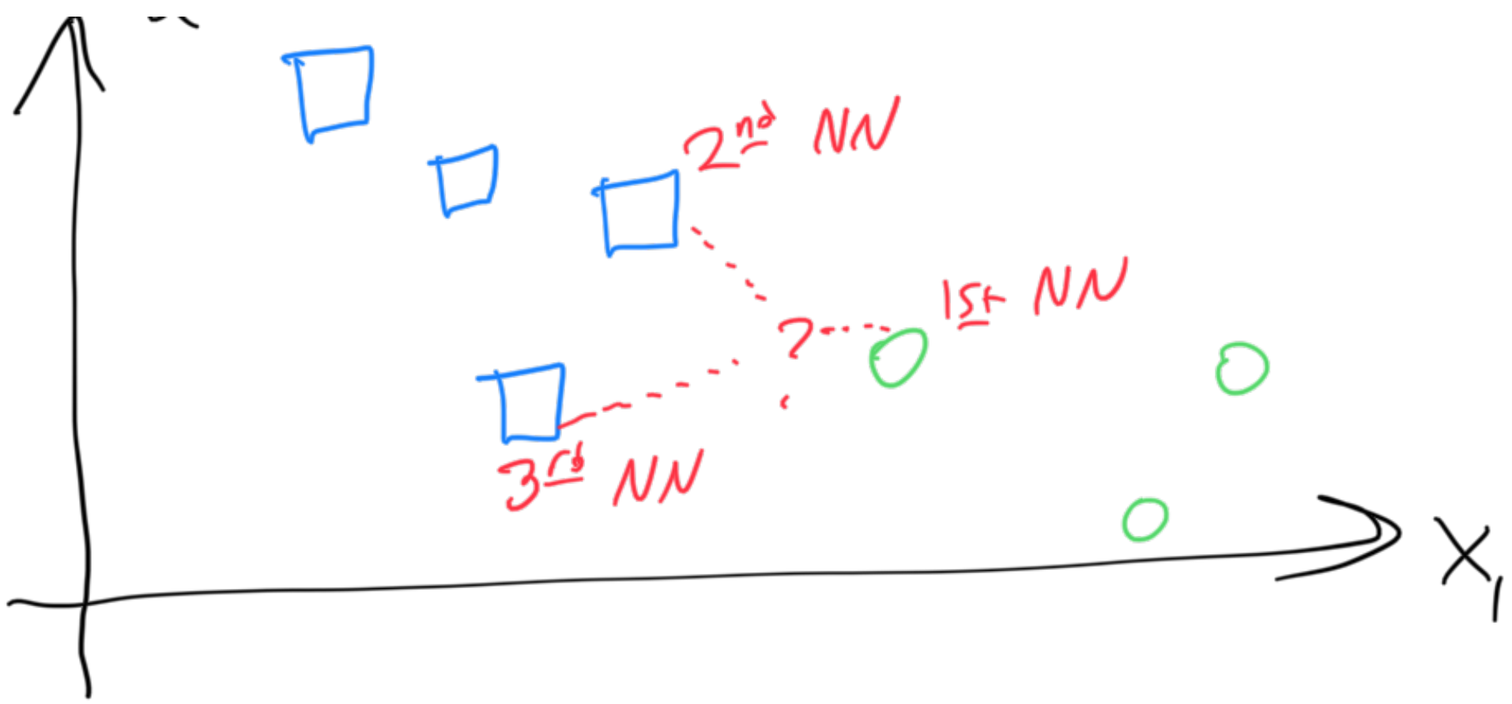
K - Nearest Neighbors (KNN)



KNN algorithm: Find K nearest neighbors of the point in question and classify as the majority

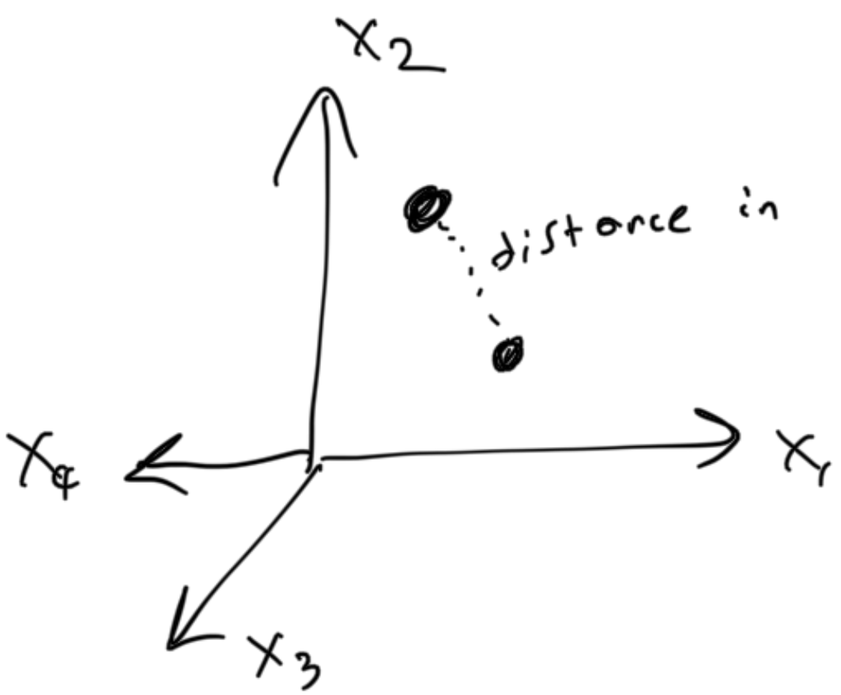
Choice of K matters: (K is a hyperometer)

x_2



K = 1 : returns green circle

K = 3 : returns blue box



(N-dimensional space, where N is the number of features)

Naive Bayes Classification

Bayes' Rule:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)} = \frac{P(a \text{ and } b)}{P(B)}$$

$$P(y|X) = \frac{P(x|y)P(y)}{P(x)}$$

(Spam classification)

To Classify:

if $P(y_i = \text{"spam"} \mid X_i) > P(y_i = \text{"not spam"} \mid X_i)$:

    return "Spam"

else:

    return "not spam"

To "Train":

$$P(y_i = \text{"spam"} \mid X_i) = \frac{P(X_i \mid y_i = \text{"spam"})\, P(y_i = \text{"spam"})}{P(X_i)}$$

$$P(y_i = \text{"spam"}) = \frac{\#\text{ spam messages}}{\#\text{ messages}}$$

$$P(X_i) = \frac{\#\text{ of emails with features } X_i}{\#\text{ messages}}$$

$$P(X_i \mid y_i = \text{"Spam"}) = \frac{\#\text{ Spam messages with features } X_i}{\#\text{ Spam messages}}$$

The "naive" in Naive Bayes:
assume all features X_i are
conditionally independent given label y_i

Example:

$$P(\text{"hello"} = 1, \text{"Vicodin"} = 0, \text{"340"} = 1 \mid \text{Spam})$$

$$\approx P(\text{"hello"} = 1 \mid \text{Spam}) \cdot P(\text{"Vicodin"} = 0 \mid \text{Spam}) \cdot P(\text{"340"} = 1 \mid \text{Spam})$$

Naive Bayes Example

$$X = \begin{bmatrix} x_1 & x_2 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

x_1 = presence of word x1

x_2 = presence of word x2

Classify $X_2 = [1 \quad 1]$

$$P(y=0 \mid X=[1\ 1]) \propto P(X_1=1 \mid y=0)\,P(X_2=1 \mid y=0)\,P(y=0)$$

$$= 1 \cdot 0.25 \cdot 0.4$$

$$= 0.1$$

$$P(y=1 \mid X=[1\ 1]) \propto P(X_1=1 \mid y=1)\,P(X_2=1 \mid y=1)\,P(y=1)$$

$$= 0.5 \cdot 0.666 \cdot 0.6$$

$= 0.2$

Because $0.2 > 0.1$,

Classify $[1\ 1]$ as "$y = 1$"