Lecture 9

17.11.2025

Today's plan and announcements

- Hour 1:
  - Recurrent neural networks training
  - Probability review
- Hour 2: Naive Bayes classification

- Announcements
  - Problem set 4 , Problem set 5
  - You can also ask questions on Problem set 3 and the python homework

    previous
  - Quiz 2 ... next Wed 26.11.2025

Recurrent neural network for time-series prediction

Goal: given $\{x_\tau\}_{\tau=0}^{t}$, predict x_{t+1} .

- Step 1: encode the input data $a_\tau = c(x_\tau), y_\tau = c(x_{\tau+1})$, where $c(\,.\,)$ is an encoding of the signal
- Step 2: fix the state dimension and the activation functions g_s, g_o.

$s_{t+1} = g_s(W_{ss}s_t + W_{sa}a_t + b_s)$, hidden state of the recurrent neural network

$\hat{y}_t = g_o(W_{os}s_t + b_o)$, output of the recurrent neural network

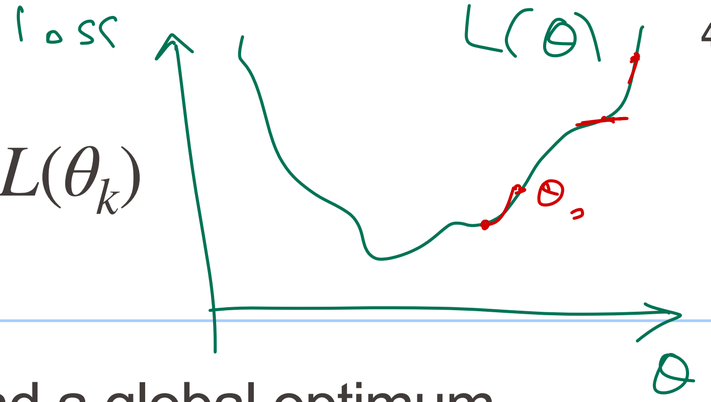
- Step 3: train the parameters $\theta = \{W_{ss}, W_{sa}, W_{os}, b_s, b_o\}$ by minimizing a loss. $L(\theta) = \frac{1}{t}\sum_{\tau=1}^{t} l(y_\tau, \hat{y}_\tau)$

training data

our model

- Step 4: given input $a_t = x_t$, predict the next signal value $\hat{y}_t \approx x_{t+1}$.

Training challenges



- Training approach: gradient descent on the loss: $\theta_{k+1} = \theta_k - \alpha \nabla_\theta L(\theta_k)$
- The loss function is generally non-convex in θ: we don't expect to find a global optimum
- Training RNNs suffers from gradient vanishing/exploding. Let's see through a simple example
  - Same issues observed for more complex examples - the growth/decay dependent on eigenvalues of W_{ss}
- Consider a simple linear RNN (scalar discrete-time linear dynamical system)

$s_{t+1} = ws_t + a_t,\ w \in \mathbb{R},$ $s_t \in \mathbb{R}$, $w \in \mathbb{R}$ → parameter

$\hat{y}_t = s_t.$ predicted output

this is a recurrent neural net with 1 hidden state, & identity activation function

- Let $L_T(\theta) = \frac{1}{T}\sum_{t=1}^{T} \frac{1}{2}(y_t - \hat{y}_t)^2$, with $\theta = w$. (consider loss over T time steps)

Simple example - back propagation

$s_{t+1} = w s_t + a_t$

$\hat{y}_t = s_t$

Define $\delta_t = \dfrac{\partial L}{\partial s_t}$. Observe that s_t affects the loss through $(y_t - \hat{y}_t)^2$ term (local) and also through $(y_\tau - \hat{y}_\tau)^2$, for $\tau > t$.

Now, δ_t satisfies the following backward recursion

$\delta_T = y_T - \hat{y}_T$

$L(\theta) = \frac{1}{T}\sum_{t=0}^{T} \frac{1}{2}(y_t - \hat{y}_t)^2 \Rightarrow \frac{\partial L}{\partial s_T} = -2 \cdot \frac{1}{2}(y_T - \hat{y}_T)$

$\delta_t = (y_t - s_t) + w\delta_{t+1}, \quad \text{for } t = 0,1,\ldots,T-1$

$\frac{\partial L}{\partial s_{t+1}} =: \delta_{t+1}$

Now, we apply the chain rule $\dfrac{\partial L}{\partial w} = \sum_{t=0}^{T-1} \delta_{t+1} \dfrac{\partial s_{t+1}}{\partial w} = \sum_{t=0}^{T-1} \delta_{t+1} s_t$.

Observe that by recursion we get $\delta_t = (y_t - s_t) + w(y_{t+1} - s_{t+1}) + w^2(y_{t+2} - s_{t+2}) + \cdots + w^{T-t}(y_T - s_T)$.

- for $|w| < 1$ the gradient terms above vanish and for $|w| > 1$ they explode as $T - t$ increases.

vanishing gradient

exploding gradient

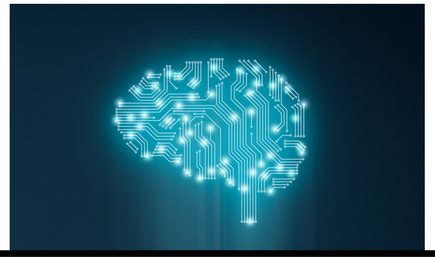
Recurrent neural network conclusions

- Powerful and early models for time-series/sequence prediction
  - Language models, forecasting
- Back propoagation generally suffers from gradient vanishing/exploding as the time horizon increases -> this has led to engineering hacks of the same mathematical idea (dynamical system prediction) that are more successful in trying and inference

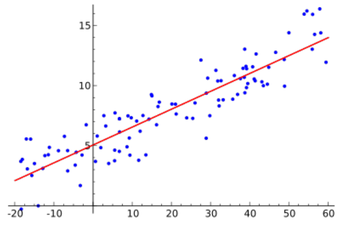
  - Long Short-Term Memory (LSTM)
  - Transformers: backbone of large language models such as ChatGPT

- Summary: learn the math fundamentals. You can then apply it to understand increasingly complex problems and come up with new approaches.

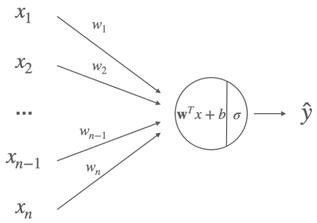
Introduction



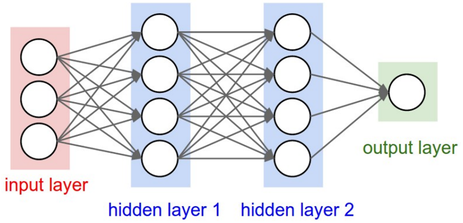
Linear regression



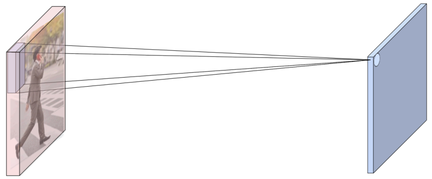
Logistic regression



Neural networks

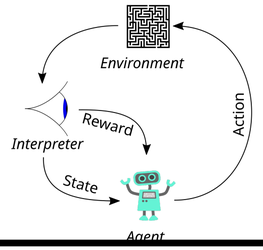


Convolutional neural networks

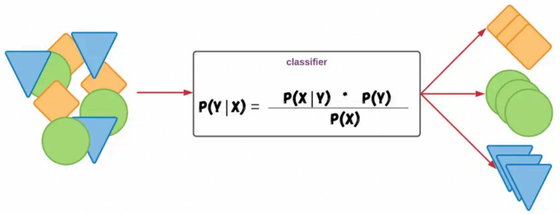


AI & sustainability

Reinforcement learning

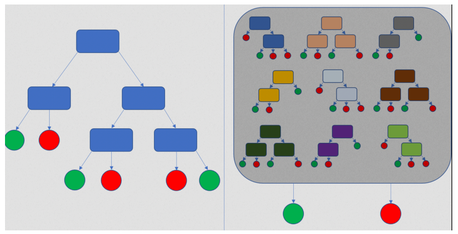


Naive Bayes

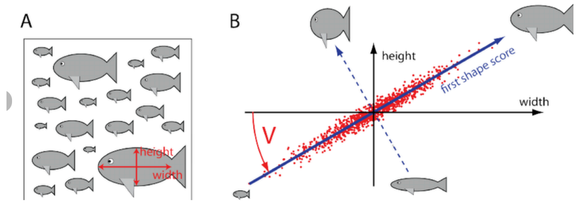


KNN

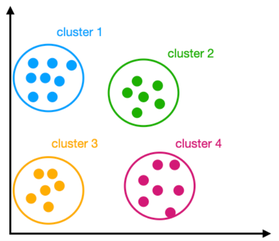
Decision-trees



Dimensionality reduction



Clustering



+ Recurrent Neural Net

Approaches we look at in this course

- **Supervised** learning: $\{x^i, y^i\}_{i=1}^N$ (features → labels)
  - Regression $y^i \in \mathbb{R}^m$: linear regression, neural network,
  - Classification $y^i \in \{1, 2, \dots, K\}$: logistic regression, neural networks, CNN
  - Time-series prediction : recurrent neural network
- **Reinforcement** learning data : $\{(s^i_{0:H+1}, a^i_{0:H}), R^i(\text{reward})\}_{i=1}^N$
  - Policy gradient : learning optimal control policy using data
- **Unsupervised** learning $\{x^i\}_{i=1}^N$
  - Dimensionality reduction
  - Clustering

Naive Bayes Classifier

classification approach $\{x^i, y^i\}_{i=1}^N$

$y^i \in \{1, 2, \dots, K\}$

$\subset$ supervised learning

Supervised learning - probabilistic classifier

Recall probabilistic interpretation of logistic regression

- Data $\{x^i, y^i\}_{i=1}^N$, $x^i \in \mathbb{R}^d$, $y^i \in \{1, 2, \dots, K\}$

- Goal: a probabilistic classifier gives probability of class of a data point x^{test} $\Rightarrow$ probability on $\{1, 2, \dots, K\}$

- Example: multinomial logistic regression

$$p(y = c \mid x) = \frac{e^{z_c}}{\sum_{j=1}^{K} e^{z_j}}, \quad z_c = w_c^T x + b_c, \quad c \in \{1, 2, \dots, K\}$$

$$z_j = w_j^T x + b_j, \quad j \in \{1, 2, \dots, K\}$$

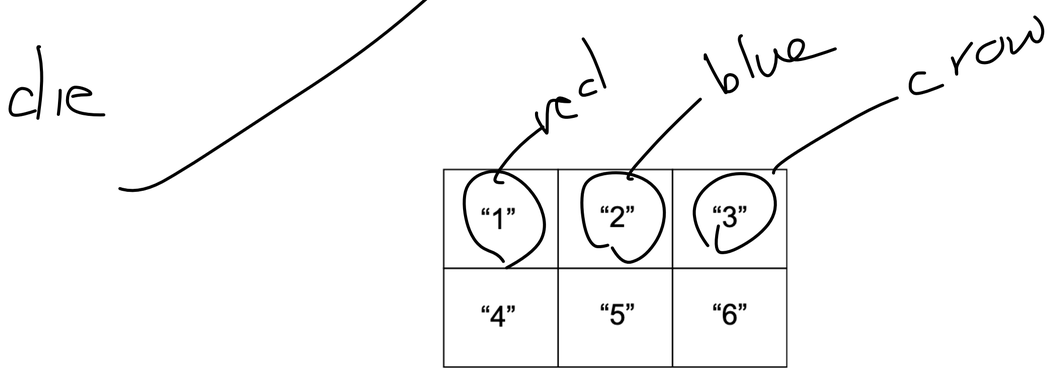
We will now discuss another approach to probabilistic classification

Probability review
Probability, random variable

- Event: outcome of a random trial, example: rolling a die

- Probability of an event : how likely an event will happen..
  ex: how likely the dice shows a "crow"

- Random variable : X assign a real value to each event



Probability review

- Random variable example $X_{all} = \{1, 2, \dots, 6\}$ (fair dice)

$$P(X = x) = \frac{1}{6}$$

$$\sum_{x \in X_{all}} P(X = x) = 1,$$

$P(X = 6)$

"1"	"2"	"3"
"4"	"5"	"6"

$1 - P(X = 6)$

$= P(X \neq 6)$

"1"	"2"	"3"
"4"	"5"	"6"

$P(X_{all}) = 1$

"1"	"2"	"3"
"4"	"5"	"6"

- Note: In this course we use probability notation informally: $P(A)$ denotes probability of event A, and $P(X = x), f(x)$ denotes distribution of a random variable, in discrete and continuous spaces, respectively.

in continuous space

$$\int_{X_{all}} f(x)\, dx = 1$$

see the notes here: https://www.cs.ubc.ca/~schmidtm/Courses/Notes/probability.pdf

Probability review

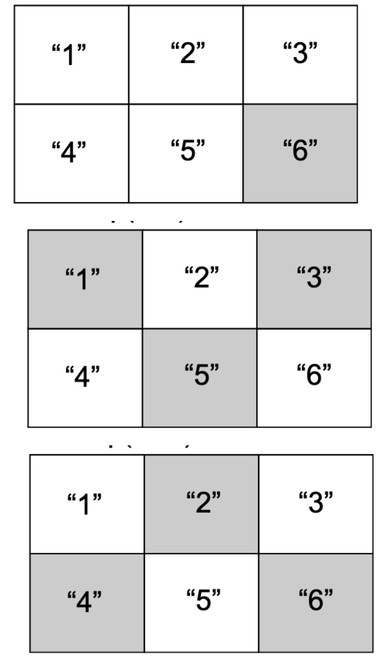
Joint probability, independence

- Joint probability of two events $P(A \cap B)$ probability of both A & B happeng

$P(\text{"odd"} \cap 6) = 0$,

$P(\text{"even"} \cap 6) = P(X=6) = \frac{1}{6}$

$\{6\} \subset \{2,4,6\}$



- Independence of two events A and B are independent:

$P(A \cap B) = P(A)P(B)$. Similarly, two random variables X, Y are independent. if $\forall x, y$ they can take, $P(X=x, Y=y) = P(X=x)P(Y=y)$

Probability review
conditional distribution

"1"	"2"	"3"
"4"	"5"	"6"

"1"	"2"	"3"
"4"	"5"	"6"

- Conditional distribution : probability distribution & is defined as,

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \text{ assume } P(B) \neq 0.$$

$$P(X=6 \mid \text{"even"}) = 1/3$$

- Conditional independence : two events A_1, A_2 are conditionally independent given event B if $P(A_1 \cap A_2 \mid B) = P(A_1|B)P(A_2|B)$

- Product rule comes from definition of conditional distribution

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

Probability review
Bayes rule

- Recall product rule $P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$

- Bayes rule $$P(B|A) = \frac{P(A|B)P(B)}{P(A)}, \quad \text{whenever } P(A) \neq 0$$

more generally

$$P(B|A_1, A_2, \dots, A_d) = \frac{P(A_1 \cap A_2 \cdots \cap A_d | B)\, P(B)}{P(A_1 \cap A_2 \cap \cdots A_d)}$$

Bayes rule for classification

$\{x^i, y^i\}_{i=1}^N \quad y^i \in \{1, 2, \dots, K\}$

Probability to observe this x knowing the label c

Probability to label with c knowing x

Probability to label with class c

$$P(y = c \mid x) = \frac{P(x \mid y = c)P(y = c)}{P(x)}$$

class c

Probability of the x

$P(y)$ is called the **prior probability** of the classes

Describes the probability to encounter the data labeled y

Bayes rule for classification

1 compute for each class $c \in \{1, 2, \dots K\}$, the following

$$P(y = c \mid x) = \frac{P(x \mid y = c) P(y = c)}{P(x)} \quad ;$$

2 take $\hat{y} = \underset{c \in \{1,2,\dots,K\}}{\operatorname{argmax}} \; P(y = c \mid x)$

example $y \in \{0, 1\}$

$$P(y = 0 \mid x) = \frac{P(x \mid y = 0) P(y = 0)}{P(x)} \quad , \text{ versus } \quad P(y = 1 \mid x) = \frac{P(x \mid y = 1) P(y = 1)}{P(x)}$$

as both probabilities above have the same denominator, we just compare the numerators. But How do we use data to compute numerators?

Spam detection example

Bag of words to encode text to features

List all words encountered in the set of texts you have (say d words)

Use a Boolean feature for the presence (1) or absence (0) of each word, $x \in \mathbb{R}^d$, with $x_i \in \{0,1\}$

In spam email detection, data: emails, labelled as: spam, or no-spam

Suppose your emails contain “Dear”, “Friend”, “Lunch”, “Money”.

Our bag of word in this simple model has 4 words only

Spam detection example

Features and label

$x^i \in \mathbb{R}^4$, i : email i

Data	"Dear" x_1	"Friend" x_2	"Lunch" x_3	"Money" x_4	class
email 1	1	1	1	0	non-spam
email 2	1	0	0	1	spam
⋮					
email N					

EPFL

Spam detection example

Dataset $D = \{x^i, y^i\}_{i=1}^N$

Use of Base's rule for spam detection

Need to compute $P(x \mid y = \text{"spam"})\, P(y = \text{"spam"})$ — class 1

$P(x \mid y = \text{"non-spam"})\, P(y = \text{"non-spam"})$ — class 0

$$P(y = \text{"spam"}) = \frac{\#\text{ spam emails in } D}{N}$$

$$P(y = \text{"non-spam"}) = \frac{\#\text{ non-spam emails in } D}{N}$$

say $x = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$. How do we compute

$P(x \mid y = 1)$, $P(x \mid y = 0)$?

$P\left(\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, y = 1\right)$, $P\left(\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, y = 0\right)$

Naive Bayes assumption

Conditional independence of different features given the label

$$P(x \mid y) = \prod_{j=1}^{d} P(x_j \mid y), \quad x \in \mathbb{R}^d, \quad y \in \{1, 2, \dots, K\}$$

$$P\left(\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, y = 1\right) = P(x_1 = 1 \mid y = 1) P(x_2 = 0 \mid y = 1) P(x_3 = 1 \mid y = 1) P(x_4 = 0 \mid y = 1)$$

Now,

$$Pr(x_j = 1 \mid y = \text{"spam"}) = \frac{\#\text{ of spam emails containg word } j}{\#\text{ of spam emails}}, \quad j = 1, \dots, d$$

$$P(x_j = 0 \mid y = \text{"spam"}) = \frac{\#\text{ of spam emails not containg word } j}{\#\text{ spam emails}}, \quad j = 1, \dots, d$$

Naive Bayes classifier

$$P(y = \text{"spam"} \mid x) = \frac{\prod_{j=1}^{d} P(x_j \mid y = \text{"spam"})\, P(y = \text{"spam"})}{P(x)}$$

$P(x)$ — no need to compute since this term is the same for all classes..

for any classification $y \in \{1, 2, \ldots, k\}$

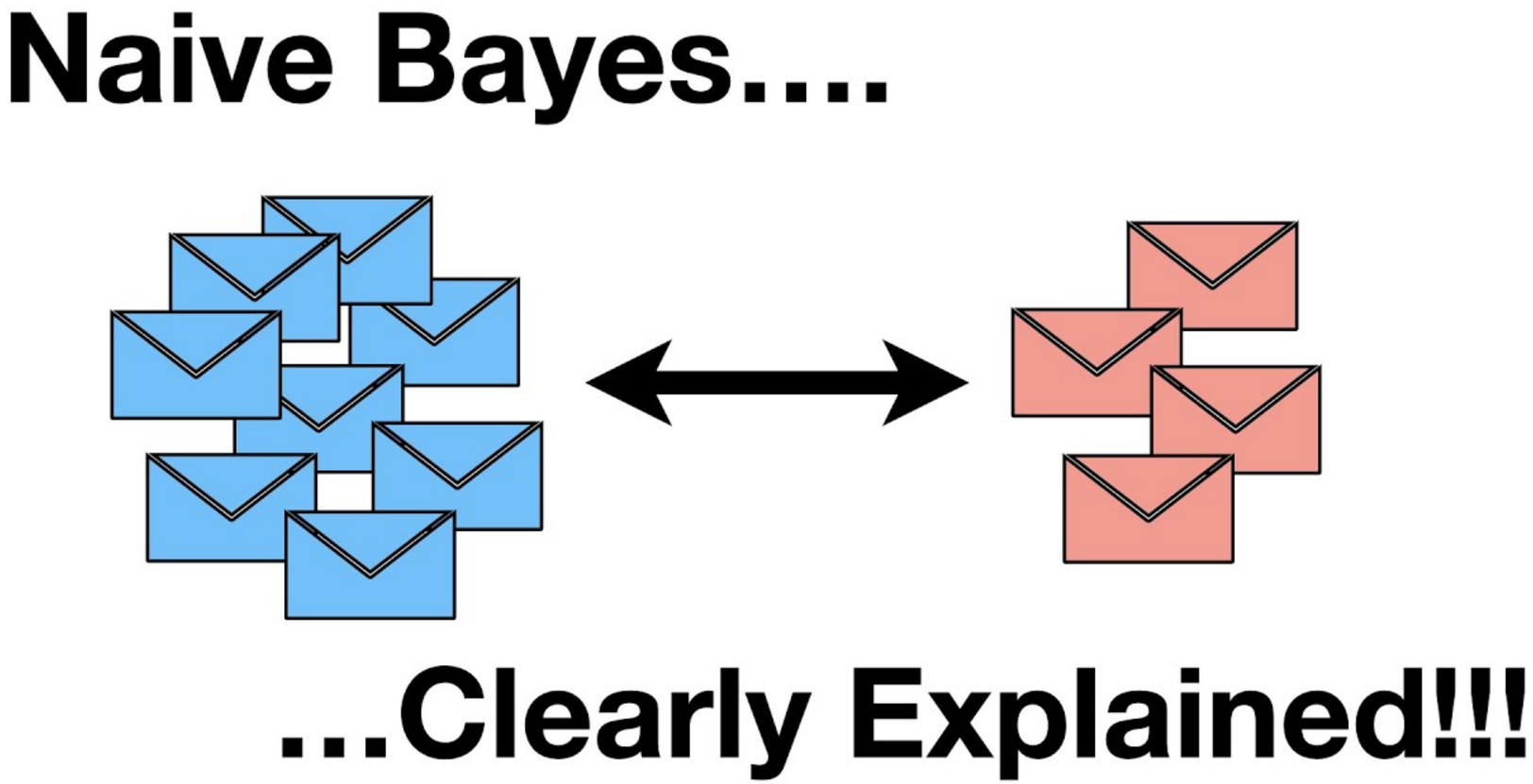
$$P(y = c \mid x) = \frac{P(x \mid y = c)\, P(y = c)}{P(x)} = \frac{\prod_{j=1}^{d} P(x_j \mid y = c) P(y = c)}{P(x)}$$

From Bayes rule

From Naive Bayes assumption

Lastly, we estimated $P(y = c)$ and $P(x_j \mid y = c)$ from our training data.

Naive Bayes classifier for spam detection
Example with worked out numbers



https://www.youtube.com/watch?v=O2L2Uv9pdDA

Activity

- Watch: https://www.youtube.com/watch?v=O2L2Uv9pdDA
  - This takes 15 minutes
- After watching the video, answer the following questions
  1. How are the features defined differently than the binary scores we defined earlier?
  2. How would you evaluate the accuracy of the classifier?
  3. Are there any hyper parameters to tune for this classifier?
  4. Is there any other approach you could imagine using for this classification?

- Next class: discuss your answers with your two nearest neighbours
- Pick one person to represent your answers to class

Naive Bayes Classifier
Continuous-valued features

Bayes rule

Continuous-valued features

$$P(y = c \,|\, x) = \frac{f_{x|y=c}(x) P(y = c)}{f_x(x)}$$

(what changed

$$\frac{P(x \mid y=c) P(y=c)}{P(x)}$$

$f_{x|y=c}(x)$ Probability density function: probability distribution for a continuous random variable

Note: $f_{x|y=c}(x)$ is not the probability of random variable taking value x . This probability is generally zero for any $x \in \mathbb{R}^d$, namely, for continuous random variables.

$$P(a \leq X \leq b) = \int_a^b f_{x|y=c}(x)\, dx \,, \quad x \in \mathbb{R}$$

$$P(x \in S) = \int_S f_{x|y=c}(x)\, dx \,, \quad x \in \mathbb{R}^d, \quad S \subseteq \mathbb{R}^d$$

Naive Bayes assumption

Continuous features

$$P(y = c \mid x) = \frac{f_{x|y=c}(x) P(y = c)}{f_x(x)}$$

Naive assumption

$$\downarrow$$

$$= \frac{\prod_{j=1}^{d} f_{x_j|y=c}(x_j)\, P(y=c)}{f_x(x)}$$

Naive Bayes assumption: conditioned on class c, the features in different dimension have independent distribution

$f_{x|y=c}(x)$: conditional density of $x \in \mathbb{R}^d$, given class c.

Gaussian Naive Bayes

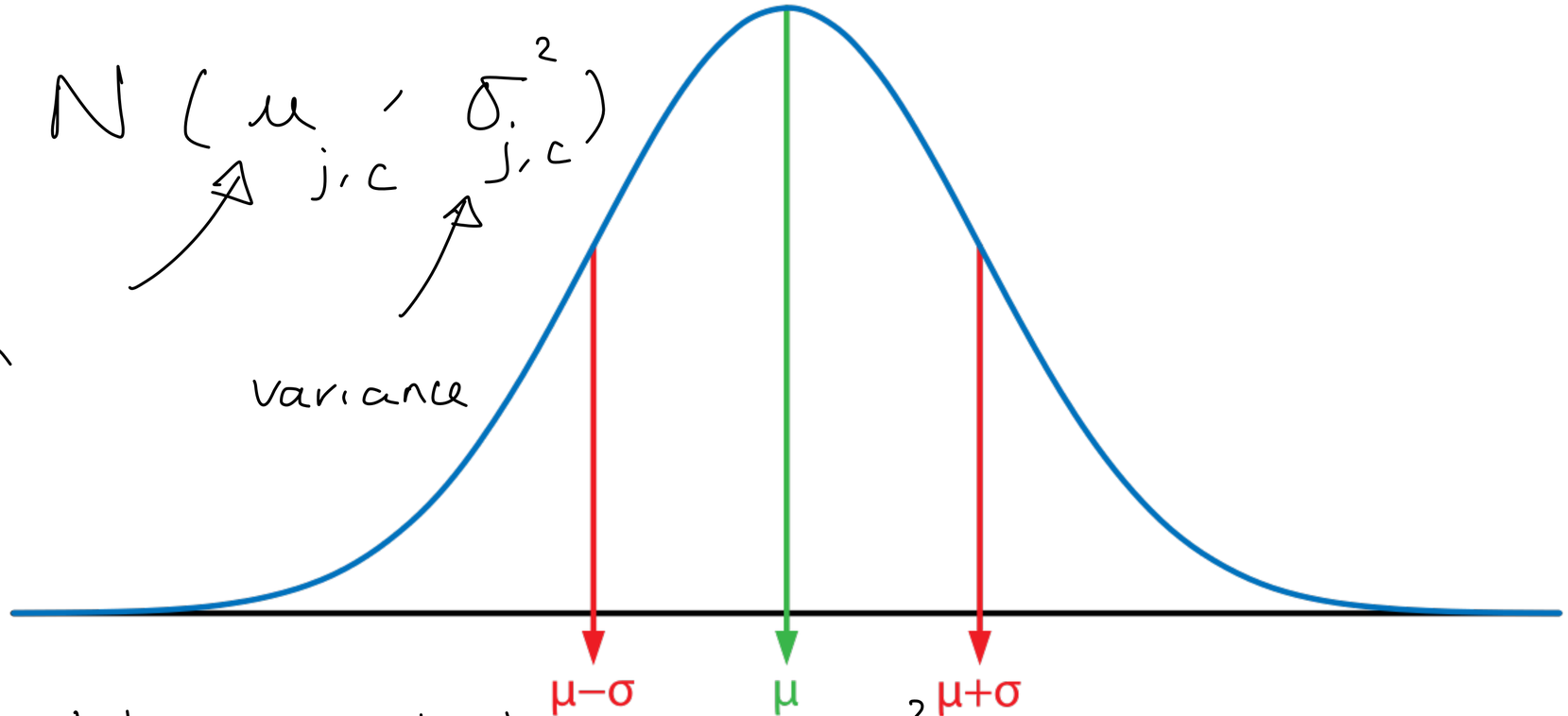
Assumes that the probability density function for each feature follows a gaussian distribution

$f_{x_j|c}(x_j)$: conditional density of feature j given class C has a Gaussian distribution

$$f_{x_j|c}(x_j) = N(\mu_{j,c}, \sigma^2_{j,c})$$

mean

variance



use training data to estimate $\mu_{j,c}$, $\sigma^2_{j,c}$

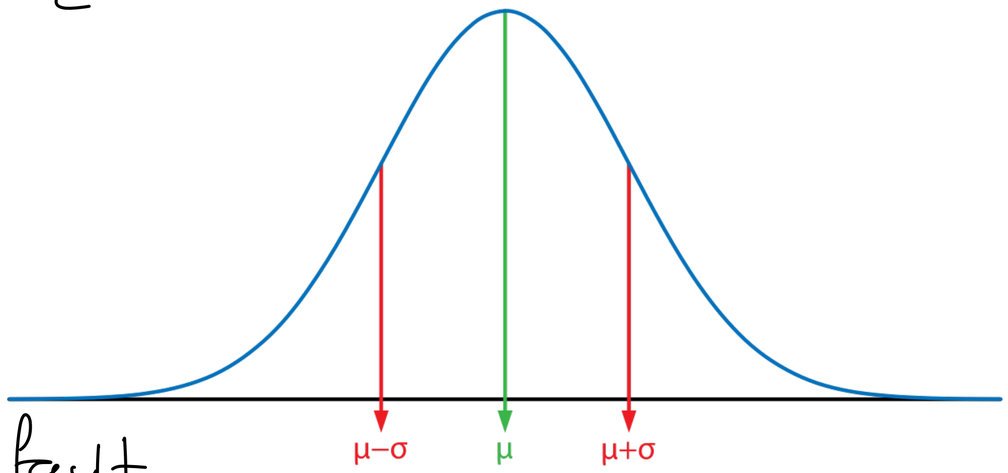
Gaussian Naive Bayes

Dataset $\{x^i, y^i\}_{i=1}^N$, $x^i \in \mathbb{R}^d$

Estimating the conditional Gaussian distribution for each feature using sample data

Recall : $\mu_{j,c} \approx \frac{1}{|I_c|} \sum_{i \in I_c} x_j^i$,

I_c : indices of data that have label class c.



example x_1 weight , class :
x_2 height fault, no fault
x_3 width 1 , 2

$\mu_{1,1}$ is average weight of faulty objects

$\mu_{2,1}$ is average height of faulty object.

Gaussian Naive Bayes

Classifier prediction

these are now Gaussian with mean & variance estimated from data

$$P(y = c \mid x) = \frac{f_{x|y=c}(x)P(y = c)}{f_x(x)} = \frac{\prod_{j=1}^{d} f_{x_j|y=c}(x_j)\, P(y=c)}{f_x(x)}$$

Note: $f_{x|y=c}(x)$ in statistics is also referred to as likelihood function, when considered as a function of class c, given a fixed x.

Suppose we get new x^{test}, you want to predict $y^{test} \in \{1, 2, \ldots, K\}$.

you compare the numerators above for each class c: $\prod_{j=1}^{d} f_{x_j|y=c}(x_j^{test})\, P(y=c)$,

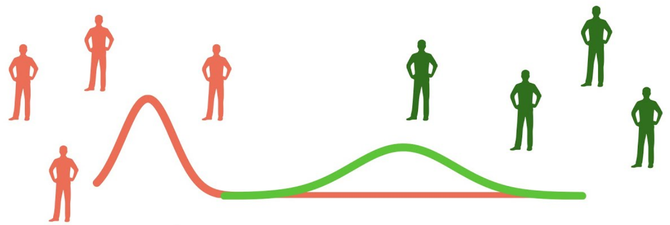
$$f_{x_j|y=c}(x_j^{test}) = \frac{1}{\sqrt{2\pi}\,\sigma_{j,c}} e^{-\frac{(x^{test} - \mu_{j,c})^2}{2(\sigma_{j,c})^2}}$$

Gaussian Naive Bayes

Naive Bayes assumption and final classifier

$$P(y = c \mid x) = \frac{f_{x_1|y=c}(x_1) f_{x_2|y=c}(x_2) \ldots f_{x_d|y=c}(x_d) P(y = c)}{f_x(x)}$$

Guassian Naive Bayes….



…Clearly Explained!!!

Naive Bayes
Summary

- Probabilistic classifier
  - Features can be finite- or continuous-valued
- Uses Bayes rule and empirical probabilities computed based on data
- Assumes different features are independent given the class label
  - In practice hard to verify the assumption but works well on certain problems

Summary

- Recurrent neural network for time-series prediction
  - Back propagation for computing gradients
  - Gradient vanishing/exploding -> motivates other approaches (LSTM, transformers)

- Back to classification
  - Naive Bayes probabilistic classifier
  - Conditional probability distribution and Naive Bases assumption

- Your tasks this week
  - Problem set 5
  - Work on python homework and exercises