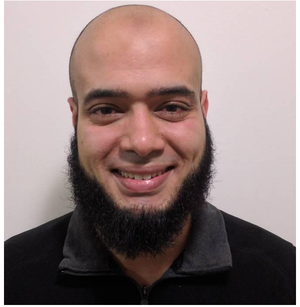
Machine Learning

Probabilistic Modeling

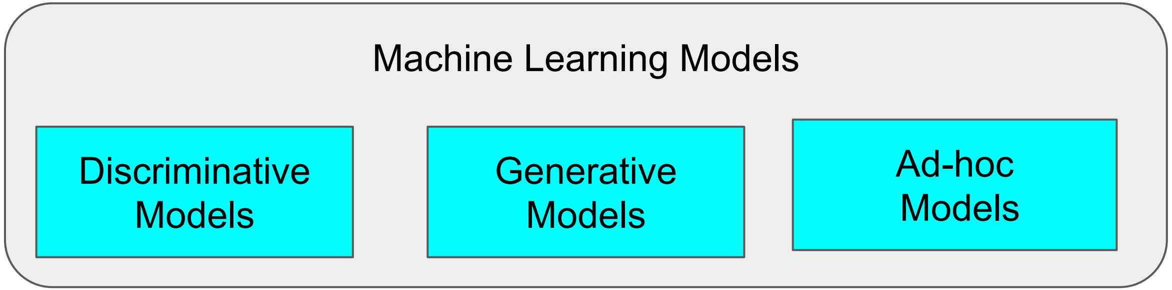
Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

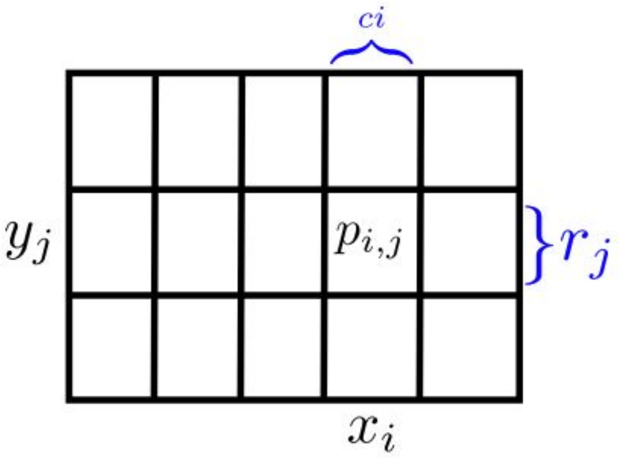


Keep in mind

- This is a **high level** lecture to show you another perspective in ML
  - No intention to dig in any of these details
  - It will be part of your later journey
- Most of ML models are either **Discriminative** (this course) or **Generative** Models (probabilistic nature).
  - A few models are neither (e.g. K-means, learning policy in reinforcement learning and some anomaly detection algorithms, etc)



Recall: Probability



Joint Probability

$$P(X = x_i, Y = y_j) = \frac{n_{i,j}}{N}$$

Marginal Probability

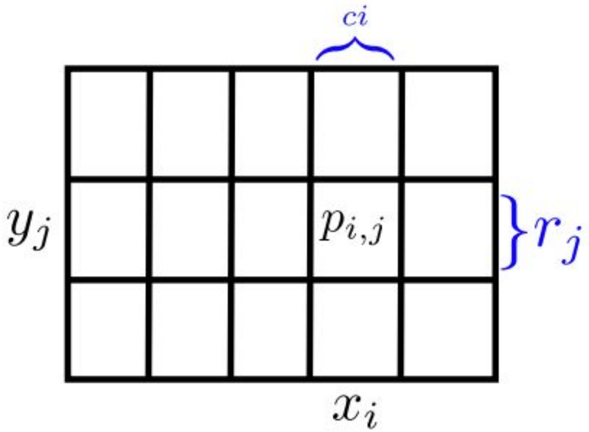
$$P(X = x_i) = \frac{c_i}{N}$$

Conditional Probability

$$P(Y = y_j \mid X = x_i) = \frac{n_{i,j}}{c_i}$$

Src: [Bishop]

Recall: Probability



Marginal Probability

$$P(X = x_i) = \frac{c_i}{N}$$

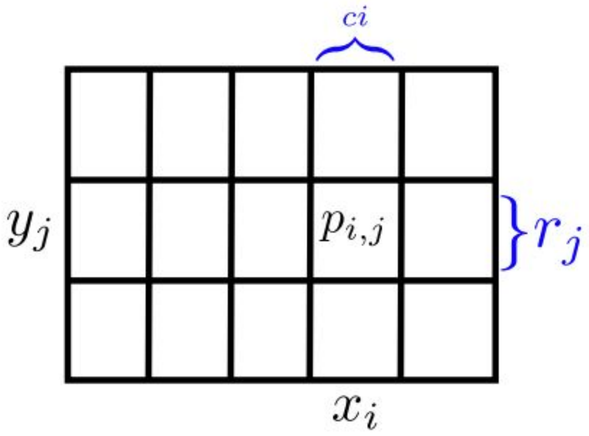
Conditional Probability

$$P(Y = y_j \mid X = x_i) = \frac{n_{i,j}}{c_i}$$

Product Rule

$$P(X = x_i, Y = y_j) = \frac{n_{i,j}}{N} = \frac{n_{i,j}}{c_i} \cdot \frac{c_i}{N}$$
$$= P(Y = y_j \mid X = x_i) P(X = x_i)$$

Recall: Probability



Marginal Probability

$$P(X = x_i) = \frac{c_i}{N}$$

Conditional Probability

$$P(Y = y_j \,|\, X = x_i) = \frac{n_{i,j}}{c_i}$$

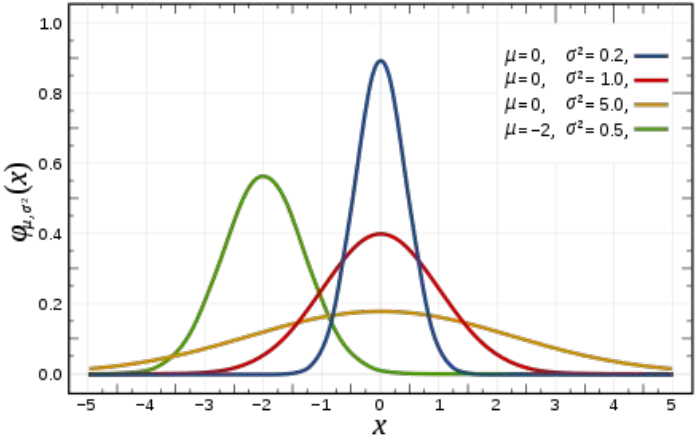
Product Rule

$$P(X = x_i, Y = y_j) = \frac{n_{i,j}}{N} = \frac{n_{i,j}}{c_i} \cdot \frac{c_i}{N}$$
$$= P(Y = y_j \,|\, X = x_i) P(X = x_i)$$

Sum Rule	$p(x) = \sum_y p(x, y)$
Product Rule	$p(x, y) = p(y \,|\, x) p(x)$

Recall: Probability Density Function

- A probability density function of continuous random variable, is a function whose value at any given sample in the sample space can be interpreted as providing a relative **likelihood** that the value of the random variable would be equal to that sample
- Below the Gaussian PDF



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Question!

- Given a Gaussian distribution with mean 1 and sigma 0.5 and a sample x = 0.25. How likely that sample **belongs to (sampled from)** this distribution?

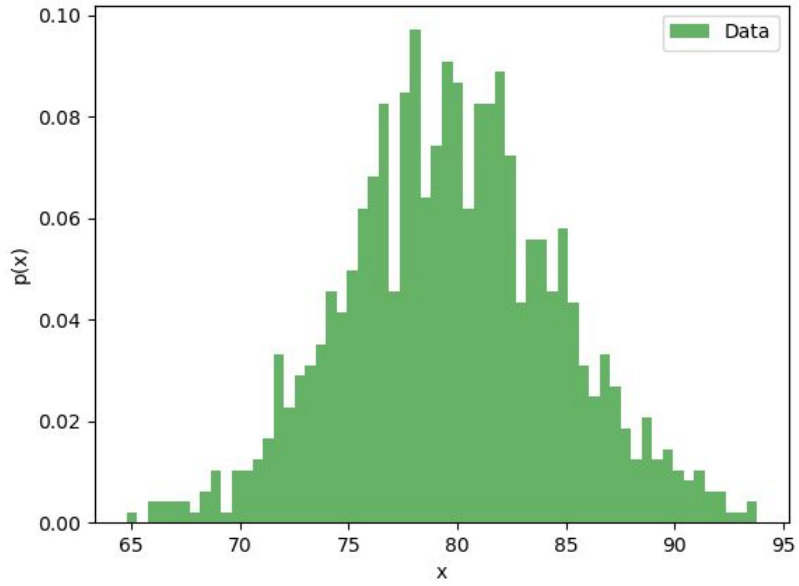
- Simply evaluate the pdf N(x=0.25 | mean=1, std=0.5^2) ≈0.0841
  - This is not a probability, but a density.
  - The higher the density, the more likely it is that x=0.25 was sampled from this Gaussian distribution.

$$f(0.25|1, 0.5^2) = \frac{1}{\sqrt{2\pi(0.5)^2}} \exp\left(-\frac{(0.25-1)^2}{2(0.5)^2}\right)$$

Question!

- Below is a gaussian p(x|θ) of students weights
  - where θ is gaussian mean and sigma
- Given 3 **independent** samples, x1, x2, x3
- How likely all of them cam from this distribution?

- p(x1|θ) * p(x2|θ) * p(x3|θ)



Likelihood

- Assume we have a dataset of N **independent** examples: X1, X2, …., Xn
  - We know the data comes from some distribution, e.g. gaussian
- Given **parameters** (weights) θ of a distribution, what is the probability of observing the dataset?
  - Θ is fixed. Evaluate the data [reasoning context]
  - Intuitively, just multiply p(x) of all the values!

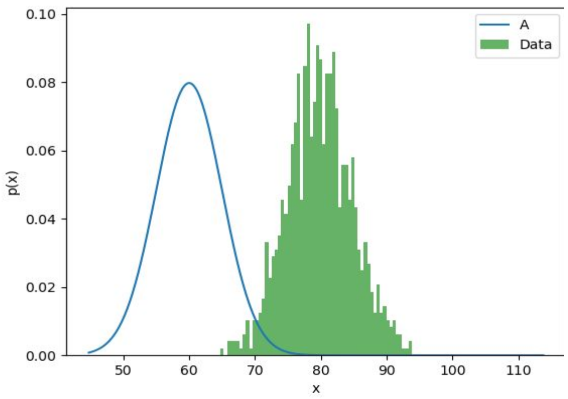
$$\mathcal{L}(\theta|X) = P(X|\theta) = \prod_{i=1}^{n} P(x_i|\theta)$$

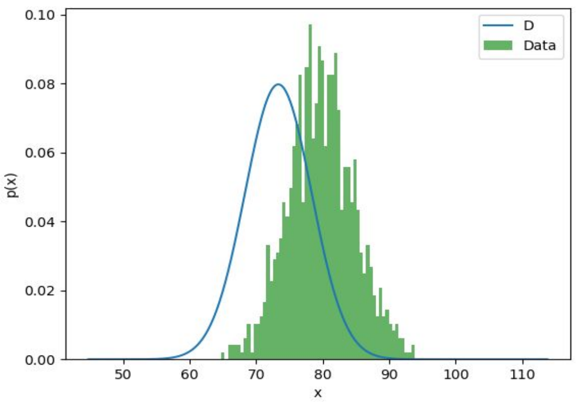
- Another perspective L(θ|Data) - *Optimization context*
  - Given a dataset, how "likely" are different values of θ?
    - Likelihood of the parameters θ, given the observed data
  - Data is fixed. Try different θ to pick the best θ using MLE (Optimization context)
  - *Not a probability function: integrate over different θ != 1*

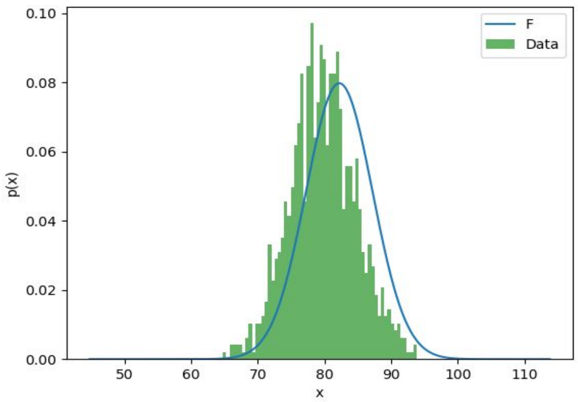
Question!

- We have a dataset of 1000 samples below in green from unknown gaussian
- Which gaussian seems the one that used to sample them? A, D or F

- F. When we multiply all p(x), it will be higher than the others!
    - **We expect most of the data to be at the mean + symmetry**
    - Little problem. Some value might be p(x) = 0! Hence likelihood = 0!



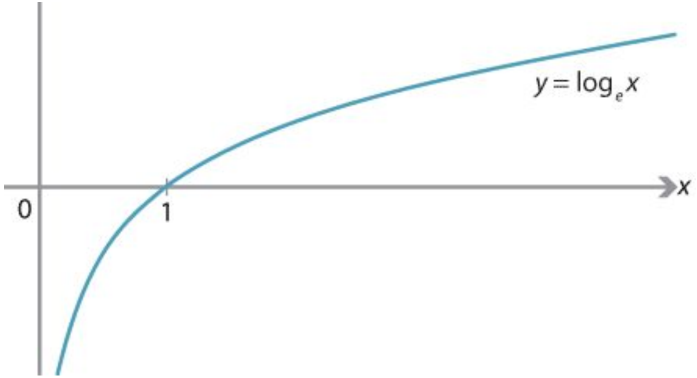




From Likelihood to Log-Likelihood

- Logarithm is monotonic, hence we can replace the multiplication of probabilities with the sum of **logs of probabilities**
  - We don't care about the exact values. Only relative value to **compare**

$$P(X|\theta) = \prod_{i=1}^{n} (p(x_i|\mu, \sigma^2)) \Rightarrow \ell(\mu, \sigma^2) = \sum_{i=1}^{n} \log\left(p(x_i|\mu, \sigma^2)\right)$$



- Recall: log(a*b) = log(a) + log(b)

From Likelihood to Log-Likelihood

- We can use the **log-likelihood score** as an indicator for the likelihood data given a distribution
    - Below X is a an array

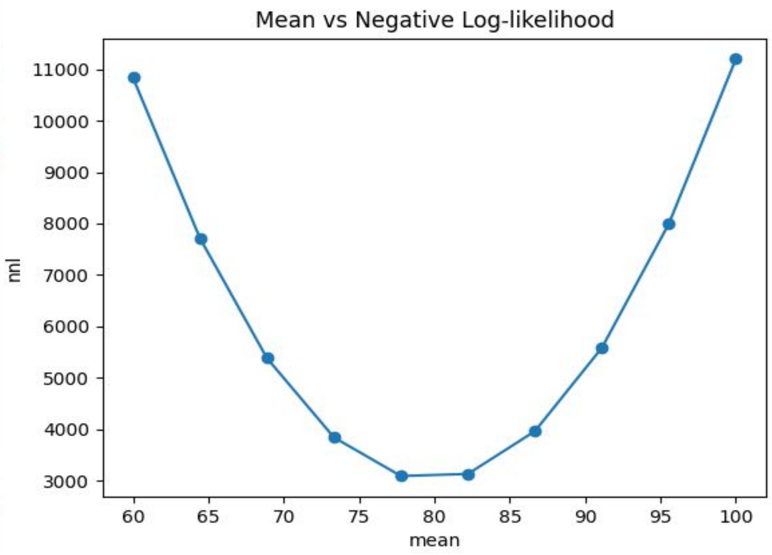
```
def gaussian(x, mu, sigma):
    return (1 / (np.sqrt(2 * np.pi * sigma ** 2))) * \
           np.exp(-0.5 * ((x - mu) / sigma) ** 2)


def likelihood_gaussian(data, mu, sigma):
    return np.prod(gaussian(data, mu, sigma))


def log_likelihood_gaussian(data, mu, sigma):
    return np.sum(np.log(gaussian(data, mu, sigma)))
```

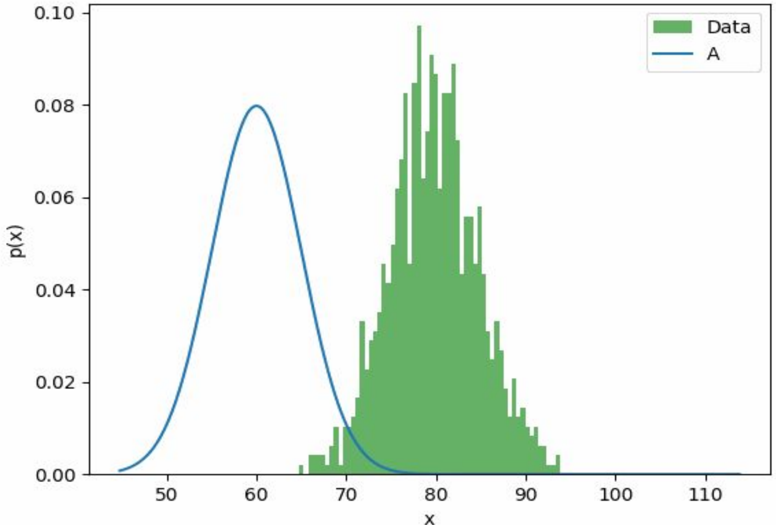
$$\prod_{i=1}^{n} (p(x_i|\mu, \sigma^2))$$

$$\sum_{i=1}^{n} \log (p(x_i|\mu, \sigma^2))$$

Maximum Likelihood Estimation (MLE)

- The best gaussian representing the data is the one with the maximum likelihood. Let's try means in range [60-100] with fixed sigma=5
  - Maximum likelihood is the same indicator as maximum log-likelihood
  - This is also the same as the **minimum negative log-likelihood**, as Max F = - Min F





Maximum Likelihood Estimation (MLE)

- MLE is the *most common* approach to find the optimal parameters (weights) to **fit a distribution** (e.g. a gaussian) to a given data coming from unknown distribution
  - MLE find the parameter **values** that **maximize** the likelihood function
- How to implement
  - First decide the probability function P
    - E.g. bernoulli or gaussian
  - Second, apply F = -log(P)
  - Write down the derivative of F
  - Now, you have a function and its derivative, just apply **gradient descent** to find the minimum
    - The minimum represents the **best parameter estimation**
  - Play in the homework

$$\mathcal{L}(\theta|X) = P(X|\theta) = \prod_{i=1}^{n} P(x_i|\theta)$$

Why prefer log probabilities?!

- Numerical Stability
  - very small numbers ⇒ underflow / very large numbers ⇒ overflow
- Simplification
  - Multiplication to Addition: log(a*b) = log(a) + log(b)
    - Easier to model and compute, e.g. in likelihood
  - Division to Subtraction
- Differentiability: Gradient Descent friendly
- Interpretability
  - Exponential Families: Log probabilities yield simpler forms for these distributions
  - Log-odds e.g. in logistic regression
  - Information Theory: Logarithms are the basis for entropy and information measures

MLE for Univariate Gaussian

- Given a dataset of a **univariate** X coming from a **Gaussian** distribution, the MLE is just the sample mean and variance of X
  - We can compute that analytically [see homework]
  - Maximum likelihood ignores parameter uncertainty (think of a single example)
- Biased sample variance issue
  - The finite sample mean is biased to the sample. But we know **population mean** is different
  - The bigger problem, the dataset variance is based on the biased dataset mean
  - There are approaches to reduce this bias due to finite sample size (e.g. **Bessel's correction**)
    - Divide by n-1 instead of n to convert biased variance to **unbiased** sample variance

Unbiased sample variance

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

MLE variance (biased)

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

Convert MLE to unbiased

$$s^2 = \left(\frac{n}{n-1}\right) \sigma^2$$

Equivalence

- We can prove that **minimizing the cross-entropy** is equivalent to:
  - Minimizing KL divergence [homework]
  - **Maximizing the likelihood** = Maximizing the **log** likelihood
    - Why? the logarithm is monotonic
  - Maximizing the **log** likelihood = Minimizing the **negative log**-likelihood
    - Why? maximizing F = minimizing -F

Equivalence

- Assume a dataset of N examples over C classes for a multi-classifier
    - The likelihood function of **observing** the given set of **labels** given the **predictions**
        - Labels y are typically one-hot-encoding
    - Apply the -log
        - Switch 2 multiplications into 2 sums
        - Convert internal to yi log yi'
        - Add negative

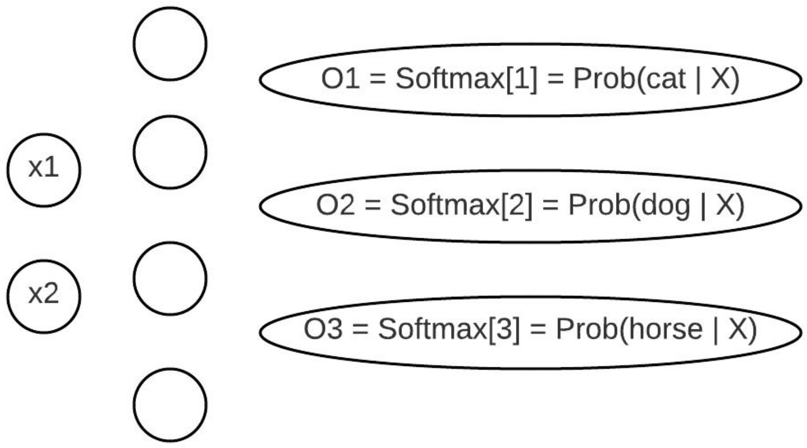
$$\mathcal{L} = \prod_{n=1}^{N} \prod_{i=1}^{C} \hat{y}_{ni}^{y_{ni}}$$

$$-\log(\mathcal{L}) = -\sum_{n=1}^{N} \sum_{i=1}^{C} y_{ni} \log(\hat{y}_{ni})$$

- This is the same as cross-entropy **but** over the whole N examples and move the negative to inside after the first sum!

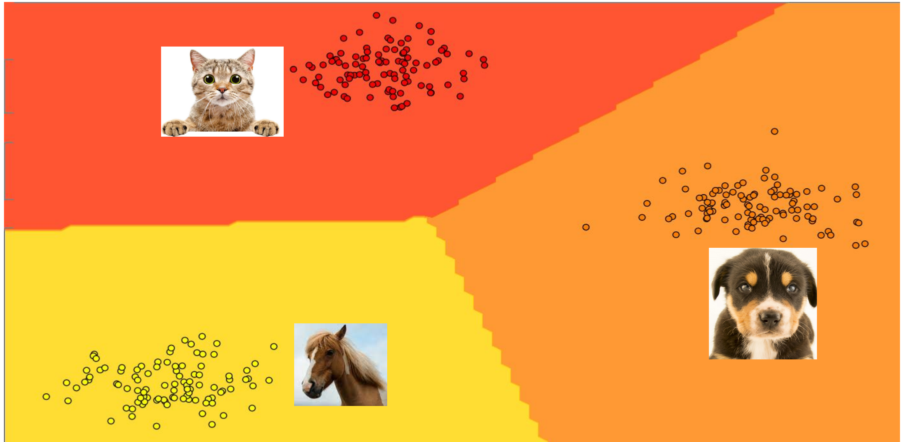
Probabilistic Formulation

- It is common to view the output of **NN for classification** as a conditional probability P(Y | X)
- **Probability (class=cat | img)** can modeled as a **function** (e.g. NN, logistic reg) that transforms **input** X into the output **probability** of being e.g. a cat



Discriminative Model

- A discriminative model discriminate (distinguish) between the values of the outcome
  - In classification (mainly) discriminate classes by finding their decision boundaries
- Examples
  - **Classifiers**: logistic regression, NN, SVM, KNN, etc
  - **Regressors**: Linear regression, NN, SVR, KNN, etc
- Modeled with P(Y|X)
  - X is the input
    - E.g. image, house features, etc
  - Y is a class for classification and *continuous output variable* for regression
  - Learns input vs output relationship

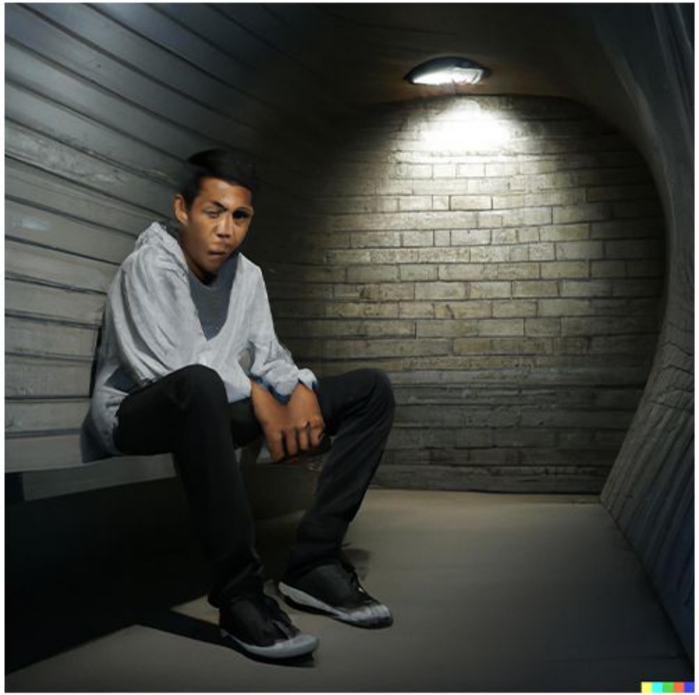


Generation

- What if I want the model to **generate** **a new** cat image not classify it?!
- What if a designer wants to describe a fancy scene and get an image?
- What if I have a text and wants to hear it with Trump voice?
- What if I want to chat with QA service like chatgpt or support services?
- What about code generation?
- This all requires the ability **to generate**!

Generative Examples

DALLE 2



Midjourney

concerned teen with depression sitting on a bench inside a tunnel 8k ultra high quality, realistic, detailed, clear lines --ar 4:5 --v 5 --q 2 --s 750

Generative Models

- Approaches that explicitly or implicitly model the distribution of (input, output)
    - Learn the **joint** probability distribution **P(X,Y)** or the data distribution **P(X)** itself
- Then, we can use it to generate new **synthetic** samples
    - In chat, like Chatgpt, we can generate (predict) the next word in a sequence
        - LLM: Large Language models
            - LaMDA: Language Model for **Dialogue** Applications
        - Generative AI (GenAI): learns to generate a content (e.g. image, audio, language).
            - Pretrained (Foundation) models are fine-tuned for more apps
            - You may give a prompt / **hallucination** is a big challenge
- Compared to discriminative models: Requires more data, more computationally expensive but can be used for more than discrimination
    - many old models can be very hard to compute or get a weak model due to strong assumptions
    - Outliers negatively affects the resulting model

Generative Models

- Assume we have input example X of n features
- And we have C labels
- We want to model P(C, X), which can be expressed with the chain rule as
  - Chain rule: P(A, B) = P(B|A) P(A) = P(A|B) P(B)

$$
\begin{aligned}
p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2, \dots, x_n, C_k) \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2 \mid x_3, \dots, x_n, C_k)\, p(x_3, \dots, x_n, C_k) \\
&= \cdots \\
&= p(x_1 \mid x_2, \dots, x_n, C_k)\, p(x_2 \mid x_3, \dots, x_n, C_k) \cdots p(x_{n-1} \mid x_n, C_k)\, p(x_n \mid C_k)\, p(C_k)
\end{aligned}
$$

Generative Models Approaches

- Gaussian Mixture Models
- Naive Bayes
- Hidden Markov Models
- Deep Learning
    - Generative Adversarial Networks (GANs)
    - Variational Autoencoders (VAEs)
    - Diffusion models (e.g. Stable Diffusion)
    - Autoregressive models (e.g. GPT: Generative Pre-trained Transformers)

Question!

- You just enrolled in the CS department at Cairo University
- What might be the percentages of girls?

- In the morning in the first class, there were 39 man and 1 girl!
- One of the class students is coming from 200 meters far toward the class
- What is the probability of being a girl based on the data you have now?

- At 100 meters, seems the person is dressing a **skirt** …maybe!
- Is the the probability of being a girl the same? Increasing? decreasing?

- At 50 meters, seems the person has a **very long hair**
- Is the the probability of being a girl the same? Increasing? decreasing?

Prior, Likelihood and Posterior

- *Tip: ignore below beta calculations :)*
- Prior: Your **belief** before data.
  - So we assumed girls are very close to 50% as we see in many classes
  - It is our **prior** belief about the data without seeing the data e.g. Beta(50, 50)
- Likelihood: How the available **data** changes your **belief**.
  - Now, we realized current data is 1/40 = 0.025 of being a girl
  - But we feel skeptical being that far from what we think about the percentages
- Posterior: Your updated belief after considering the data.
  - To avoid data **extreme** conclusions, use your prior to update your belief
    - E.g. Beta(50+1,50+39) ⇒ E[Beta()] = (50 + 1) / (50 + 1 + 50 + 39) = 0.364
  - Update the belief when you saw a skirt ⇒ **0.70**
  - Update the belief when you saw a long hair ⇒ **0.85**

Bayes' Theorem

- **Bayes' Theorem**: Update the probability estimate with more data (evidence)
- If X is the input, Y is the output, K is one of the classes

$$\text{Posterior Probability} = \frac{\text{Likelihood} \times \text{Prior Probability}}{\text{Evidence}}$$

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

$$P(\mathbf{x}) = \sum_k P(\mathbf{x}|y = k)P(y = k)$$

Bayes' Theorem: Simple example

- We have an email that has the word "X = Free". We want to know to what degree the email can be a "Y= Spam" because of this word
  - **Prior Probability P(Y)**: Our **initial belief** that 20% of emails are spam
  - **Likelihood P(X|Y)**: We got new information: email's text with **word free** ⇒ P(free | spam).
    - How probable the **observed data** is, **given** a set of **parameters** in the model: **P(X|θ)**
      - If the email is really spam, what is the probability to see word free?
      - Observe It is P(Input | Output) - **flipped**
  - **Evidence P(X)**: The **overall probability** of observing word "free", regardless spam or not
    - We loop on all possible classes to **marginalize**
  - **Posterior Probability P(Y|X)**: The **updated** probability of being spam with word "free"
    - P(Output | Input): is what we want for the conditional distribution
- Prior Probability (Spam) = **0.2** / Prior Probability (Not Spam) = 0.8
  - Sums to 1. For example our dataset has 200 spam emails and 800 non-spam emails
- Likelihood ("free" | Spam) = **0.7** / Likelihood ("free" | Not Spam) = 0.1

Bayes’ Theorem: Simple example

- Evidence (Marginal distribution / denominator)
  - Iterate on possible classes: spam / !spam
  - P(X) = (0.2×0.7) + (0.8×0.1) = 0.14 + 0.08 = 0.22

$$P(\mathbf{x}) = \sum_k P(\mathbf{x}|y=k)P(y=k)$$

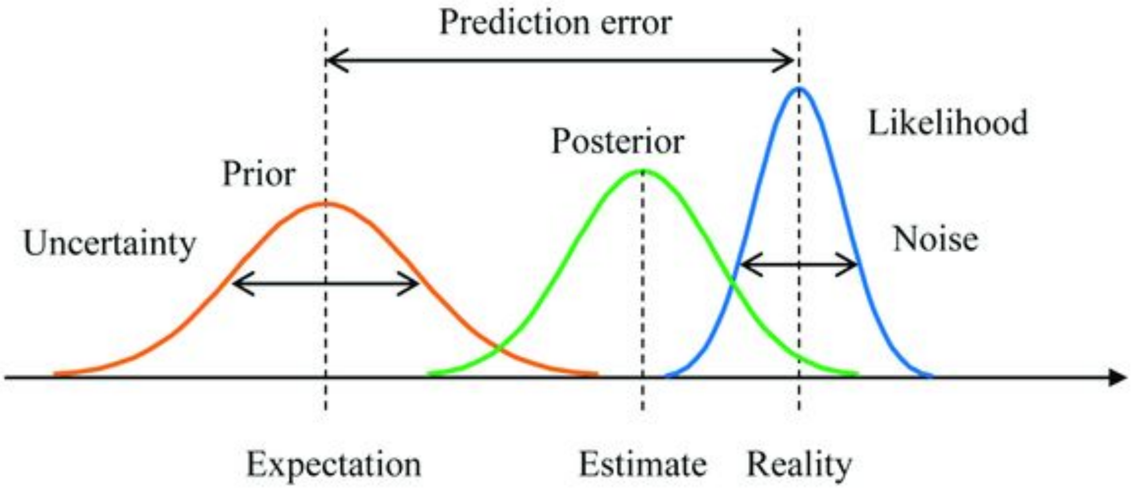
- Posterior Probability (Spam | "free")
  - 0.7 x 0.2 / 0.22 = 0.64

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

- Overall
  - Before we see a new email: priority probability of spam is 0.2
  - When we saw a an email with word “free”, the updated probability jumped to 0.64
- Generative Models
  - P(X, Y) = P(X | Y) P(Y) ⇒ Likelihood x Prior ⇒ computed in Bayes’ theorem from data
  - **Parameter Estimation** of the distribution may be done with Maximum Likelihood Estimation
  - **Bayesian Inference**: parameters and predictions are treated as probability distributions

From Discriminative to Generative

- Generative models model P(X,Y) or P(X|Y) + P(Y).
  - In the Bayesian framework, these prior probabilities play a crucial role
- Discriminative models focus primarily on the likelihood, in the form P(Y|X), while bayes rule enhance the likelihood with the prior



Img src

Naive Bayes Classifier: A Generative Model

- Naive Bayes classifier is a **classification** algorithm: it has a **naive** assumption that all features are **mutually independent**, **conditional** on the kth **category**

$$p(x_i \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k)\,.$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k \mid x_1, \ldots, x_n) &= \propto p(C_k, x_1, \ldots, x_n) \\ &\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k)\, \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,, \end{aligned}$$

$$p(C_k \mid x_1, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k) \qquad Z = p(\mathbf{x}) = \sum_k p(C_k)\, p(\mathbf{x} \mid C_k)$$

Naive Bayes Classifier: Implementation

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

- Assume we have data
  - X: (300, 5): 300 examples, each of 5 features
  - Y: 3 classes 0, 1, 2
- **Prior Probability** P(Y): Let’s use the frequency of each class
- Assume the underlying probability is gaussian
  - For each feature, compute its mean and std. This is its pdf (using MLE)
- **Likelihood** P(X[i]|θ): Given a sample x, we can use gaussian pdf to evaluate
  - How probable this X[i] in this gaussian distribution
- **P(X)**: it is just a constant for a classifier
  - So, if we ignored, we have unscaled probabilities. We just needs the maximum
- Posterior Probability P(Y|X): likelihood x prior

```python
# Gaussian Probability Density Function
def gaussian_pdf(x, mean, std):
    return (1 / (np.sqrt(2 * np.pi * std ** 2))) * \
            np.exp(-((x - mean) ** 2) / (2 * std ** 2))
```

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

```python
def get_data():
    # Generate data that represents 3 classes
    # for each class, generate 100 examples each of 5 features
    # all data follows gaussian (mean, sgima) are provided
    X0 = np.random.normal(2, 1, (100, 5))
    X1 = np.random.normal(4, 1, (100, 5))
    X2 = np.random.normal(6, 1, (100, 5))

    # Combine into one dataset
    X = np.vstack([X0, X1, X2])     # 300 x 5
    y = np.array([0] * 100 + [1] * 100 + [2] * 100) # classes
    # y: 100 0, then 100 1, then 100 2 for ground truth

    # Create some test data
    X_test = np.array([[1.5, 2, 2.2, 1.9, 2.1],
                       [4.2, 3.8, 4.5, 3.9, 4.0],
                       [6.1, 5.9, 6.0, 6.2, 6.1]])

    return X, y, X_test
```

```python
X, y, X_test = get_data()
gnb = GNB()
gnb.train(X, y)
print( gnb.predict(X_test) )
```

```python
def train(self, X, y):
    self.classes = np.unique(y)  # [0, 1, 2]
    self.means, self.stds, self.priors = {}, {}, {}

    for c in self.classes:
        X_c = X[y == c]
        self.means[c] = np.mean(X_c, axis=0)
        self.stds[c] = np.std(X_c, axis=0)
        self.priors[c] = len(X_c) / len(X) # frequency
```

- Notice, we collect all the data of cth class
- This represents the conditional dependency on the class P(x | c)
    - Recall: conditional implies cut only those examples

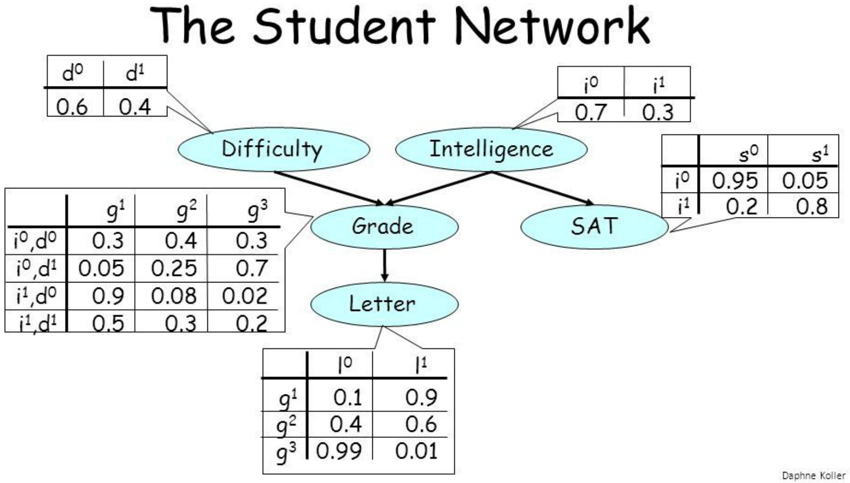
```python
    def predict(self, X):
        num_samples = X.shape[0]
        preds = np.zeros(num_samples)

        for i in range(num_samples):
            posteriors = {}

            for c in self.classes:
                # compute probability of each independent feature
                props = gaussian_pdf(X[i, :], self.means[c], self.stds[c])
                # multiply 5 props to get likelihood
                likelihood = np.prod(props)
                # compute posteriors, without constant Z
                posteriors[c] = likelihood * self.priors[c]

            # return the index of the max class value
            preds[i] = max(posteriors, key=lambda k: posteriors[k])

        return preds
```

Frequentist vs. Bayesian inference

- Frequentist and Bayesian are two different approaches to **statistical inference**, and both have their merits and disadvantages.
  - used for making predictions, estimating parameters, and testing hypotheses
- Frequentist Inference
  - Views probability as the long-term frequency of events
  - Considers parameters to be fixed but unknown
    - MLE is common for estimating the parameters from the sampled data
    - MLE provides a point estimate without directly giving a measure of uncertainty
- Bayesian Inference
  - Views probability as a measure of belief.
  - Treats data as fixed once observed. Considers parameters to be random variables.
  - Allows for the inclusion of **prior** information.
- Good article

Graphical Models

- Graphical models are a visual framework to represent complex systems and their probabilistic relationships using graphs
  - nodes represent random variables
  - edges represent probabilistic relationships between them
- Key professor Daphne Koller - course



Probabilistic Modeling

- Why?
  - **Uncertainty** Quantification: Data (input/label), parameters P(θ|D), Model!
  - Flexibility ([un]supervised), Interpretability, Bayesian Updating
- Key Components: Random Variables, Parameters, Likelihood, Prior, Posterior Distribution
- Models
  - **Bayesian** Models: typically generative models, but can be discriminative $P(\theta|D) = \frac{P(D|\theta) \times P(\theta)}{P(D)}$
  - **Graphical** Models: both generative and discriminative
    - Bayesian Networks, Markov Random Fields (MRFs), Factor Graphs, DBNs, Conditional Random Fields (CRFs)
  - **Time Series** Models: typically discriminative models (forecasting), but can be generative
    - Statistical (ARIMA, STL, ETS), Bayesian (BSTS, Gaussian Processes), Hidden Markov Models, State Space Models, Kalman Filters, Random Forest, Gradient Boosting, LSTM, Transformers (not all are probabilistic)

Relevant Materials

- Likelihood
  - Link - link - link - link - link - video
- Probabilistic modeling
  - Link
- Bayesian or Frequentist
  - Bayesian or Frequentist, Which Are You? By Michael I. Jordan (Part 1 of 2)
  - Frequentism and Bayesianism: A Practical Introduction
  - Understanding frequentist vs. Bayesian inference

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”