Q1: In a Gaussian Naïve Bayes classifier, we typically assume that the standard deviation σ_i of $P(X_i|Y=k)$ is the same for all class values k. Now, suppose we relax this assumption and allow σ_{ik} to vary depending on both the feature index i and the class k. Given that $P(X_i|Y=k)$ follows a Gaussian distribution $N(\mu_{ik}, \sigma_{ik})$, derive the new form of $P(Y|X)$. Does this modified Naïve Bayes classifier still result in a logistic regression-like form for $P(Y|X)$? Justify your answer with a derivation.

Special Notes:

2.1 General Gaussian naive Bayes Classifiers and Logistic Regression [15 points]

Let's make our Gaussian naive Bayes classifiers a little more general by removing the assumption that the standard deviation σ_i of $P(X_i|Y=k)$ does not depend on k. As a result, for each X_i, $P(X_i|Y=k)$ is a Gaussian distribution $N(\mu_{ik}, \sigma_{ik})$, where $i = 1, 2, \ldots, n$ and $k = 0, 1$. Note that now the standard deviation σ_{ik} of $P(X_i|Y=k)$ depends on both the attribute index i and the value k of Y.

Question: is the new form of $P(Y|\mathbf{X})$ implied by this more general Gaussian naive Bayes classifier still the form used by logistic regression? Derive the new form of $P(Y|\mathbf{X})$ to prove your answer.

Solution:

★ **SOLUTION:** **No**, the new $P(Y|\mathbf{X})$ implied by this more general Gaussian naive classifier is no longer the form used by logistic regression. Here is the derivation.

As shown in page 8 of [Mitchell: Naive Bayes and Logistic Regression], we have:

$$\begin{aligned}
P(Y=1|\mathbf{X}) &= \frac{P(Y=1)P(\mathbf{X}|Y=1)}{P(Y=1)P(\mathbf{X}|Y=1)+P(Y=0)P(\mathbf{X}|Y=0)} \\
&= \frac{1}{1+\frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)}} \\
&= \frac{1}{1+\exp(\ln\frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\frac{P(\mathbf{X}|Y=0)}{P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\sum_i \ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)})}
\end{aligned}$$

[4]Note that certain discriminative classifiers are non-probabilistic: they directly estimate a function $f : \mathbf{X} \rightarrow Y$ instead of $P(Y|\mathbf{X})$. We will see such classifiers later this semester, but it is beyond the scope of this question.

Now the standard deviation σ_{ik} of $P(X_i|Y=k)$ depends on both the attribute index i and the value k of Y, so we have:

$$\begin{aligned}
\sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln \frac{\frac{1}{\sqrt{2\pi\sigma_{i0}^2}} \exp(\frac{-(X_i-\mu_{i0})^2}{2\sigma_{i0}^2})}{\frac{1}{\sqrt{2\pi\sigma_{i1}^2}} \exp(\frac{-(X_i-\mu_{i1})^2}{2\sigma_{i1}^2})} \\
&= \sum_i \ln \frac{\sigma_{i1}}{\sigma_{i0}} + \sum_i (\frac{(X_i-\mu_{i1})^2}{2\sigma_{i1}^2} - \frac{(X_i-\mu_{i0})^2}{2\sigma_{i0}^2}) \\
&= \sum_i \ln \frac{\sigma_{i1}}{\sigma_{i0}} + \sum_i \frac{(\sigma_{i0}^2-\sigma_{i1}^2)X_i^2 + 2(\mu_{i0}\sigma_{i1}^2 - \mu_{i1}\sigma_{i0}^2)X_i + \mu_{i1}^2\sigma_{i0}^2 - \mu_{i0}^2\sigma_{i1}^2}{2\sigma_{i0}^2\sigma_{i1}^2} \\
&= \sum_i (\ln \frac{\sigma_{i1}}{\sigma_{i0}} + \frac{\mu_{i1}^2\sigma_{i0}^2 - \mu_{i0}^2\sigma_{i1}^2}{2\sigma_{i0}^2\sigma_{i1}^2}) + \sum_i \frac{(\mu_{i0}\sigma_{i1}^2 - \mu_{i1}\sigma_{i0}^2)}{\sigma_{i0}^2\sigma_{i1}^2} X_i + \sum_i \frac{(\sigma_{i0}^2-\sigma_{i1}^2)}{2\sigma_{i0}^2\sigma_{i1}^2} X_i^2
\end{aligned}$$

As a result, we have $P(Y=1|\mathbf{X})$ as follows:

$$\begin{aligned}
P(Y=1|\mathbf{X}) &= \frac{1}{1+\exp(\ln \frac{1-\pi}{\pi} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})} \\
&= \frac{1}{1+\exp(w_0 + \sum_i w_i X_i + \sum_i v_i X_i^2)}
\end{aligned}$$

where

$$\begin{aligned}
w_0 &= \ln \frac{1-\pi}{\pi} + \sum_i (\ln \frac{\sigma_{i1}}{\sigma_{i0}} + \frac{\mu_{i1}^2\sigma_{i0}^2 - \mu_{i0}^2\sigma_{i1}^2}{2\sigma_{i0}^2\sigma_{i1}^2}) \\
w_i &= \frac{(\mu_{i0}\sigma_{i1}^2 - \mu_{i1}\sigma_{i0}^2)}{\sigma_{i0}^2\sigma_{i1}^2} \\
v_i &= \frac{(\sigma_{i0}^2-\sigma_{i1}^2)}{2\sigma_{i0}^2\sigma_{i1}^2}
\end{aligned}$$

In general we do not have $\sigma_{i1} = \sigma_{i0}$ so the quadratic term $v_i X_i^2$ in $P(Y=1|\mathbf{X})$ will not disappear. Therefore the form of $P(Y=1|\mathbf{X})$ is no longer the form of logistic regression.

Q2: Consider a Gaussian Bayes classifier where Y is a Boolean variable following a Bernoulli distribution with probability $\pi = P(Y=1)$. Instead of assuming independence between features, we now consider two continuous attributes, X_1 and X_2, which follow a bivariate Gaussian distribution $N(\mu_{1k}, \mu_{2k}, \sigma_1, \sigma_2, \rho)$ given $Y = k$. The means μ_{1k} and μ_{2k} depend on Y, but the standard deviations σ_1, σ_2 and correlation ρ remain constant across Y.
Using the given probability density function of the bivariate Gaussian distribution, derive the form of $P(Y|X)$. Does the resulting expression resemble the form used in logistic regression? Justify your answer with mathematical derivation.

Special Notes:

2.2 Gaussian Bayes Classifiers and Logistic Regression [15 points]

Students in 10-701 are all smart, so clearly we will not be satisfied by only studying a "naive" classifier. In this part, we will turn our attention to a specific class of Gaussian Bayes classifiers (without "naive", yeah!). We consider the following assumptions for our Gaussian Bayes classifiers:

1. Y is a boolean variable following a Bernoulli distribution, with parameter $\pi = P(Y=1)$ and thus $P(Y=0) = 1-\pi$.
2. $\mathbf{X} =< X_1, X_2 >$, i.e., we only consider **two** attributes, where each attribute X_i is a continuous random variable. X_1 and X_2 are **not** conditionally independent given Y. We assume $P(X_1, X_2|Y=k)$ is a **bivariate Gaussian distribution** $N(\mu_{1k}, \mu_{2k}, \sigma_1, \sigma_2, \rho)$, where μ_{1k} and μ_{2k} are means of X_1 and X_2, σ_1 and σ_2 are standard deviations of X_1 and X_2, and ρ is the **correlation** between X_1 and X_2. Note that μ_{1k} and μ_{2k} depend on the value k of Y, but σ_1, σ_2, and ρ do **not** depend on Y. Also recall that the density of a bivariate Gaussian distribution, given $(\mu_{1k}, \mu_{2k}, \sigma_1, \sigma_2, \rho)$, is:

$$P(X_1, X_2|Y=k) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp[-\frac{\sigma_2^2(X_1-\mu_{1k})^2 + \sigma_1^2(X_2-\mu_{2k})^2 - 2\rho\sigma_1\sigma_2(X_1-\mu_{1k})(X_2-\mu_{2k})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}]$$

Solution:

★ **SOLUTION:** Yes, the new $P(Y|\mathbf{X})$ implied by this not-so-naive Gaussian Bayes classifier is still the form used by logistic regression.

$$\begin{aligned}
P(Y=1|\mathbf{X}) &= \frac{P(Y=1)P(\mathbf{X}|Y=1)}{P(Y=1)P(\mathbf{X}|Y=1)+P(Y=0)P(\mathbf{X}|Y=0)} \\
&= \frac{1}{1+\frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)}} \\
&= \frac{1}{1+\exp(\ln\frac{P(Y=0)P(\mathbf{X}|Y=0)}{P(Y=1)P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\frac{P(\mathbf{X}|Y=0)}{P(\mathbf{X}|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)})}
\end{aligned}$$

Note that we must work on the joint distribution $P(X_1,X_2|Y=0)$ and $P(X_1,X_2|Y=1)$ because X_1 and X_2 are no longer conditionally independent given Y. We proceed by focusing on the term $\ln\frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)}$:

$$\begin{aligned}
\ln\frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)} &= \ln\frac{\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}}{\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}}+\ln\exp[(*)] \\
&= \ln\exp[(*)] \\
&= (*)
\end{aligned}$$

where (*) is the following formulation, obtained as the difference between the exponential parts of the bivariate Gaussian densities $P(X_1,X_2|Y=0)$ and $P(X_1,X_2|Y=1)$:

$$\begin{aligned}
(*) &= \frac{\sigma_2^2(X_1-\mu_{11})^2+\sigma_1^2(X_2-\mu_{21})^2-2\rho\sigma_1\sigma_2(X_1-\mu_{11})(X_2-\mu_{21})}{2(1-\rho^2)\sigma_1^2\sigma_2^2} \\
&- \frac{\sigma_2^2(X_1-\mu_{10})^2+\sigma_1^2(X_2-\mu_{20})^2-2\rho\sigma_1\sigma_2(X_1-\mu_{10})(X_2-\mu_{20})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}
\end{aligned}$$

It turns out, by a few steps of derivations, that all quadratic terms X_1^2, X_2^2 and X_1X_2 are canceled in the above (*) and therefore we end up with the following:

$$\begin{aligned}
(*) &= \frac{2\sigma_2^2(\mu_{10}-\mu_{11})+2\rho\sigma_1\sigma_2(\mu_{21}-\mu_{20})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}X_1 \\
&+ \frac{2\sigma_1^2(\mu_{20}-\mu_{21})+2\rho\sigma_1\sigma_2(\mu_{11}-\mu_{10})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}X_2 \\
&+ \frac{\sigma_2^2(\mu_{11}^2-\mu_{10}^2)+\sigma_1^2(\mu_{21}^2-\mu_{20}^2)+2\rho\sigma_1\sigma_2(\mu_{10}\mu_{20}-\mu_{11}\mu_{21})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}
\end{aligned}$$

As a result, we have:

$$\begin{aligned}
P(Y=1|\mathbf{X}) &= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\ln\frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)})} \\
&= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+(*))} \\
&= \frac{1}{1+\exp(w_0+w_1X_1+w_2X_2)}
\end{aligned}$$

As a result, we have:

$$\begin{aligned} P(Y=1|\mathbf{X}) &= \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi}+\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1-\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0+(\mu_0^T-\mu_1^T)\Sigma^{-1}X)} \\ &= \frac{1}{1+\exp(w_0+\mathbf{w}^T\mathbf{X})} \end{aligned}$$

where $w_0=\ln\frac{1-\pi}{\pi}+\frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1-\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0$ is a scalar and $\mathbf{w}=\Sigma^{-1}(\mu_0-\mu_1)$ is an $n\times 1$ parameter vector.
This is still the form of logistic regression (in vector and matrix notation).

Q3: Suppose you are training a generative naive Bayes model using a dataset where the distribution of labels does not accurately represent real-world conditions or the distribution in the test data. Given this scenario, which probability should you use to make the best decision on whether to predict y_0 for a given input X?

Solution:

You're building a **generative naive Bayes model**, which estimates:

$$P(y \mid X) \propto P(X \mid y)P(y)$$

To make predictions, you want to **maximize the posterior** $P(y \mid X)$, which is proportional to the product of:

- The **likelihood:** $P(X \mid y)$
- The **prior:** $P(y)$

Since the label distribution in the training data **does not reflect the real-world/test label distribution**, it's important to use the **true prior** $P(y_0)$ — not the one estimated from the biased training set.

So the best estimate for deciding whether or not to predict y_0 given X is:

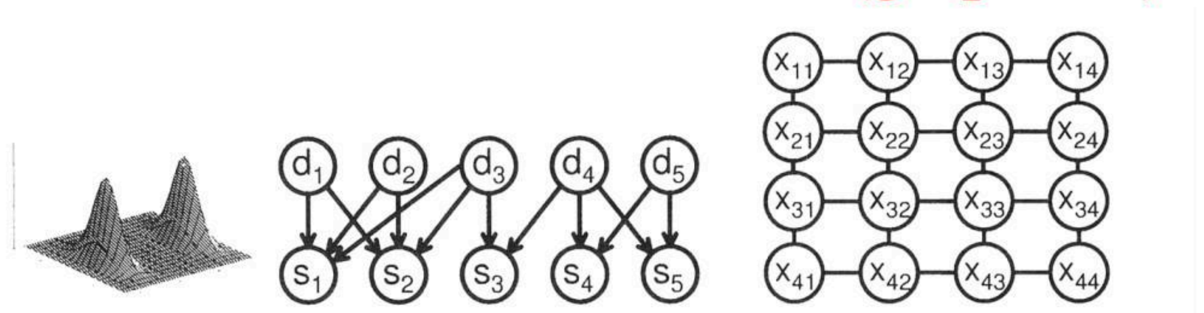
$$P(X \mid y_0)P(y_0)$$

This helps account for the mismatch between the training and test label distributions.

Q4: How do graphical representations of generative models, such as mixture models, directed graphical models, and undirected graphical models (Markov Random Fields), differ in their structure and applications? Provide examples of their use in real-world scenarios.

Solution:

Generative Models (graphical)



d_1 d_2 d_3 d_4 d_5
s_1 s_2 s_3 s_4 s_5
x_{11} x_{12} x_{13} x_{14}
x_{21} x_{22} x_{23} x_{24}
x_{31} x_{32} x_{33} x_{34}
x_{41} x_{42} x_{43} x_{44}

(a) Mixture Model (b) Directed Graphical Model (c) Undirected Graphical Model

Parent node selects between components

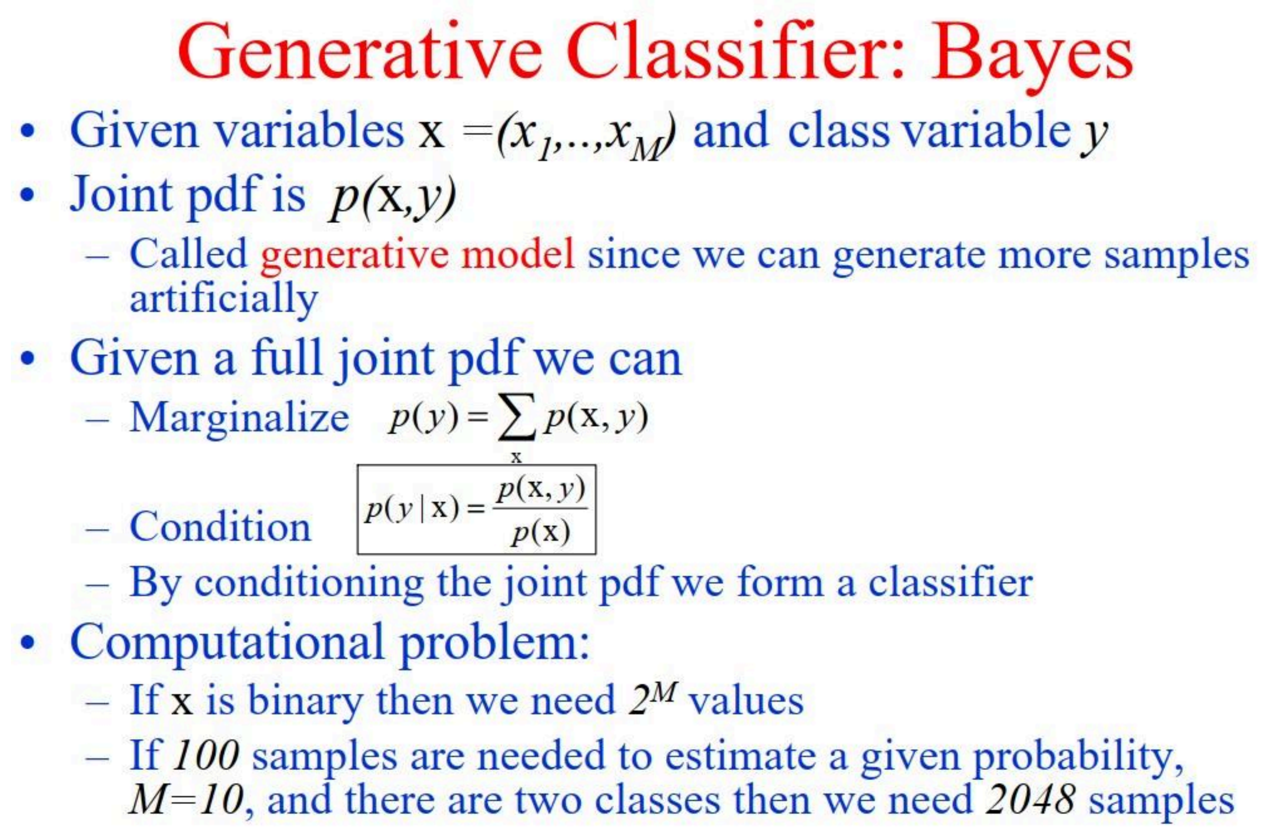
Quick Medical Reference -DT

Diagnosing Diseases from Symptoms

Markov Random Field

Q5: How does a generative classifier based on Bayes' theorem utilize the joint probability distribution, and what computational challenges arise when estimating probabilities for high-dimensional binary variables?

Solution:



Generative Classifier: Bayes

- Given variables $\mathrm{x} = (x_1,..,x_M)$ and class variable y
- Joint pdf is $p(\mathrm{x},y)$
  - Called generative model since we can generate more samples artificially
- Given a full joint pdf we can
  - Marginalize $p(y) = \sum_{\mathrm{x}} p(\mathrm{x}, y)$
  - Condition $\boxed{p(y \mid \mathrm{x}) = \frac{p(\mathrm{x}, y)}{p(\mathrm{x})}}$
  - By conditioning the joint pdf we form a classifier
- Computational problem:
  - If x is binary then we need 2^M values
  - If 100 samples are needed to estimate a given probability, $M=10$, and there are two classes then we need 2048 samples

Q6: Describe the architecture of a Generative Adversarial Network and how the generator and discriminator interact during training.

Solution: A Generative Adversarial Network comprises a generator and a discriminator. The generator produces synthetic data, attempting to mimic real data, while the discriminator evaluates the authenticity of the generated samples. During training, the generator and discriminator engage in a dynamic interplay, each striving to outperform the other. The generator aims to create more realistic data, and the discriminator seeks to improve its ability to differentiate between real and generated samples.

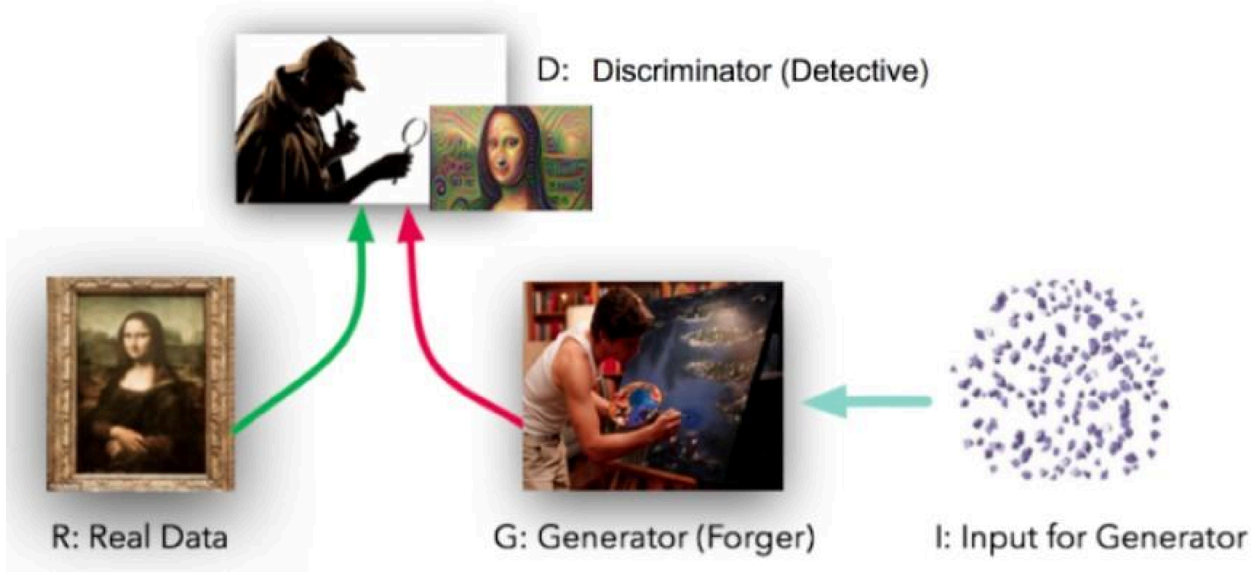


Image Source: https://climate.com/tech-at-climate-corp/gans-disease-identification-model/

Q7: How do conditional generative models differ from unconditional ones? Provide an example scenario where a conditional approach is beneficial.

Solution:

Conditional generative models differ from unconditional ones by considering additional information or conditions during the generation process. In unconditional generative models, such as vanilla GANs or VAEs, the model learns to generate samples solely based on the underlying data distribution. However, in conditional generative models, the generation process is conditioned on additional input variables or labels.

For example, in the context of image generation, an unconditional generative model might learn to generate various types of images without any specific constraints. On the other hand, a conditional generative model could be trained to generate images of specific categories, such as generating images of different breeds of dogs based on input labels specifying the breed.

A scenario where a conditional approach is beneficial is in tasks where precise control over the generated outputs is required or when generating samples belonging to specific categories or conditions. For instance:

- In image-to-image translation tasks, where the goal is to convert images from one domain to another (e.g., converting images from day to night), a conditional approach allows the model to learn the mapping between input and output domains based on paired data.
- In text-to-image synthesis, given a textual description, a conditional generative model can generate corresponding images that match the description, enabling applications like generating images from textual prompts.

Conditional generative models offer greater flexibility and control over the generated outputs by incorporating additional information or conditions, making them well-suited for tasks requiring specific constraints or tailored generation based on input conditions.

Q8: How does overfitting manifest in generative models, and what techniques can be used to prevent it during training?

Solution:

Overfitting in generative models occurs when the model memorizes the training data rather than learning the underlying data distribution, resulting in poor generalization to new, unseen data. Overfitting can manifest in various ways in generative models:

i. Mode Collapse: One common manifestation of overfitting in generative models is mode collapse, where the generator produces a limited variety of samples, failing to capture the full diversity of the data distribution.
ii. Poor Generalization: Generative models might generate samples that closely resemble the training data but lack diversity or fail to capture the nuances present in the true data distribution.
iii. Artifacts or Inconsistencies: Overfitting can lead to the generation of unrealistic or inconsistent samples, such as distorted images, implausible text sequences, or nonsensical outputs.

To prevent overfitting in generative models during training, various techniques can be employed:

i. Regularization: Regularization techniques such as weight decay, dropout, and batch normalization can help prevent overfitting by imposing constraints on the model's parameters or introducing stochasticity during training.
ii. Early Stopping: Monitoring the performance of the generative model on a validation set and stopping training when performance begins to deteriorate can prevent overfitting and ensure that the model generalizes well to unseen data.

iii. Data Augmentation: Increasing the diversity of the training data through techniques like random cropping, rotation, scaling, or adding noise can help prevent overfitting by exposing the model to a wider range of variations in the data distribution.
iv. Adversarial Training: Adversarial training, where the generator is trained to fool a discriminator that is simultaneously trained to distinguish between real and generated samples, can help prevent mode collapse and encourage the generation of diverse and realistic samples.
v. Ensemble Methods: Training multiple generative models with different architectures or initializations and combining their outputs using ensemble methods can help mitigate overfitting by leveraging the diversity of multiple models.
vi. Cross-Validation: Partitioning the dataset into multiple folds and training the model on different subsets while validating on the remaining data can help prevent overfitting by providing more reliable estimates of the model's performance on unseen data.

Q9: What is gradient clipping, and how does it help in stabilizing the training process of generative models?

Solution:

Gradient clipping is a technique used during training to limit the magnitude of gradients, typically applied when the gradients exceed a predefined threshold. It is commonly employed in deep learning models, including generative models like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs).

Gradient clipping helps stabilize the training process of generative models in several ways:

i. Preventing Exploding Gradients: In deep neural networks, particularly in architectures with deep layers, gradients can sometimes explode during training, leading to numerical instability and hindering convergence. Gradient clipping imposes an upper bound on the gradient values, preventing them from growing too large and causing numerical issues.
ii. Mitigating Oscillations: During training, gradients can oscillate widely due to the complex interactions between the generator and discriminator (in GANs) or the encoder and decoder (in VAEs). Gradient clipping helps dampen these oscillations by constraining the magnitude of the gradients, leading to smoother and more stable updates to the model parameters.
iii. Enhancing Convergence: By preventing the gradients from becoming too large or too small, gradient clipping promotes more consistent and predictable updates to the model parameters. This can lead to faster convergence during training, as the model is less likely to encounter extreme gradient values that impede progress.
iv. Improving Robustness: Gradient clipping can help make the training process more robust to variations in hyperparameters, such as learning rates or batch sizes. It provides an

additional safeguard against potential instabilities that may arise due to changes in the training dynamics.

Q10: Discuss strategies for training generative models when the available dataset is limited.

Solution:

When dealing with limited datasets, training generative models can be challenging due to the potential for overfitting and the difficulty of capturing the full complexity of the underlying data distribution. However, several strategies can be employed to effectively train generative models with limited data:

i. Data Augmentation: Augmenting the existing dataset by applying transformations such as rotation, scaling, cropping, or adding noise can increase the diversity of the training data. This helps prevent overfitting and enables the model to learn more robust representations of the data distribution.
ii. Transfer Learning: Leveraging pre-trained models trained on larger datasets can provide a valuable initialization for the generative model. By fine-tuning the pre-trained model on the limited dataset, the model can adapt its representations to the specific characteristics of the target domain more efficiently.
iii. Semi-supervised Learning: If a small amount of labeled data is available in addition to the limited dataset, semi-supervised learning techniques can be employed. These techniques leverage both labeled and unlabeled data to improve model performance, often by jointly optimizing a supervised and unsupervised loss function.
iv. Regularization: Regularization techniques such as weight decay, dropout, and batch normalization can help prevent overfitting by imposing constraints on the model's parameters or introducing stochasticity during training. Regularization encourages the model to learn more generalizable representations of the data.
v. Generative Adversarial Networks (GANs) with Progressive Growing: Progressive growing GANs (PGGANs) incrementally increase the resolution of generated images during training, starting from low resolution and gradually adding detail. This allows the model to learn more effectively from limited data by focusing on coarse features before refining finer details.
vi. Ensemble Methods: Training multiple generative models with different architectures or initializations and combining their outputs using ensemble methods can help mitigate the limitations of a small dataset. Ensemble methods leverage the diversity of multiple models to improve the overall performance and robustness of the generative model.
vii. Data Synthesis: In cases where the available dataset is extremely limited, data synthesis techniques such as generative adversarial networks (GANs) or variational autoencoders (VAEs) can be used to generate synthetic data samples. These synthetic samples can be

combined with the limited real data to augment the training dataset and improve model performance.

Q11: Describe the concept of learning rate scheduling and its role in optimizing the training process of generative models over time.
Solution:

Learning rate scheduling is a crucial technique in the optimization of neural networks, including generative models, which involves adjusting the learning rate—the step size used to update the model's weights—over the course of training. The learning rate is a critical hyperparameter that determines how much the model adjusts its weights in response to the estimated error each time it is updated. If the learning rate is too high, the model may overshoot the optimal solution; if it's too low, training may proceed very slowly or stall.

In the context of training generative models, such as Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs), learning rate scheduling can significantly impact the model's ability to learn complex data distributions effectively and efficiently.

Role in Optimizing the Training Process:

i. Avoids Overshooting: Early in training, a higher learning rate can help the model quickly converge towards a good solution. However, as training progresses and the model gets closer to the optimal solution, that same high learning rate can cause the model to overshoot the target. Gradually reducing the learning rate helps avoid this problem, allowing the model to fine-tune its parameters more delicately.
ii. Speeds Up Convergence: Initially using a higher learning rate can accelerate the convergence by allowing larger updates to the weights. This is especially useful in the early phases of training when the model is far from the optimal solution.
iii. Improves Model Performance: By carefully adjusting the learning rate over time, the model can escape suboptimal local minima or saddle points more efficiently, potentially leading to better overall performance on the generation task.
iv. Adapts to Training Dynamics: Different phases of training may require different learning rates. For example, in the case of GANs, the balance between the generator and discriminator can vary widely during training. Adaptive learning rate scheduling can help maintain this balance by adjusting the learning rates according to the training dynamics.

Common Scheduling Strategies:

- Step Decay: Reducing the learning rate by a factor every few epochs.
- Exponential Decay: Continuously reducing the learning rate exponentially over time.
- Cosine Annealing: Adjusting the learning rate following a cosine function, leading to periodic adjustments that can help in escaping local minima.

- Warm-up Schedules: Gradually increasing the learning rate from a small to a larger value during the initial phase of training, which can help in stabilizing the training of very deep models.

Q12: Compare and contrast the use of L1 and L2 loss functions in the context of generative models. When might one be preferred over the other?
Solution:

Both loss functions are used to measure the difference between the model's predictions and the actual data, but they do so in distinct ways that affect the model's learning behavior and output characteristics.

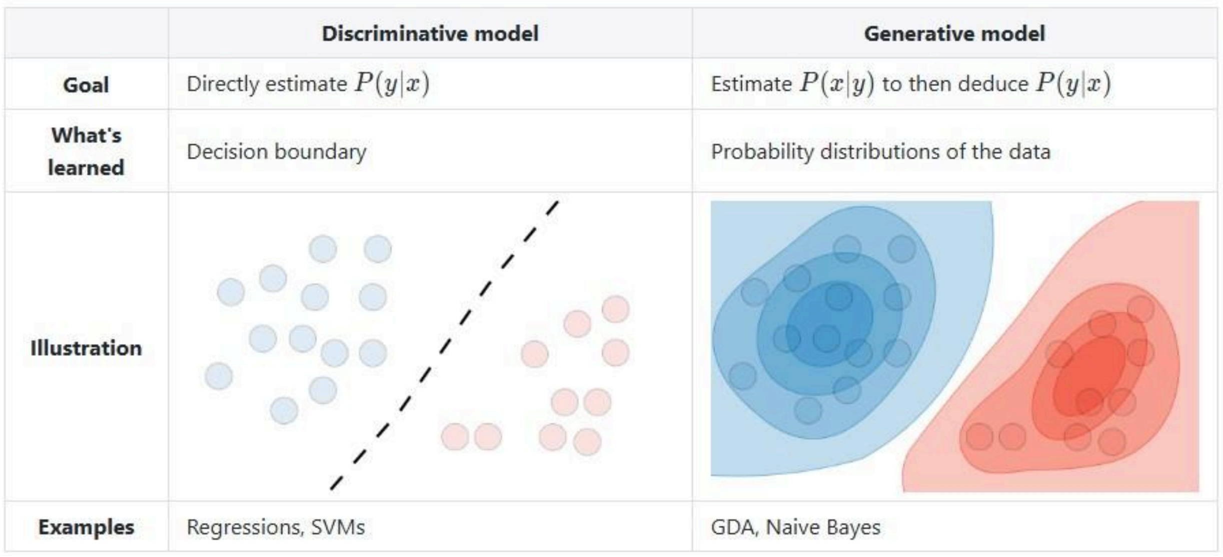
L1 Loss (Absolute Loss): The L1 loss function calculates the absolute differences between the predicted values and the actual values. This approach is less sensitive to outliers because it treats all deviations the same, regardless of their magnitude. In the context of generative models, using L1 loss can lead to sparser gradients, which may result in models that are more robust to noise in the input data. Moreover, L1 loss tends to produce results that are less smooth, which might be preferable when sharp transitions or details are desired in the generated outputs, such as in image super-resolution tasks.

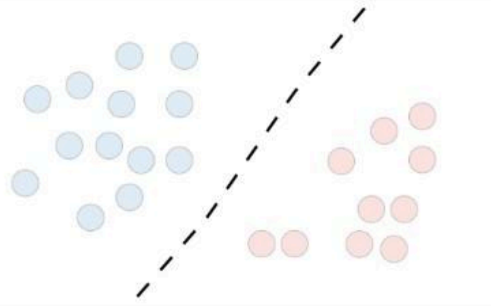
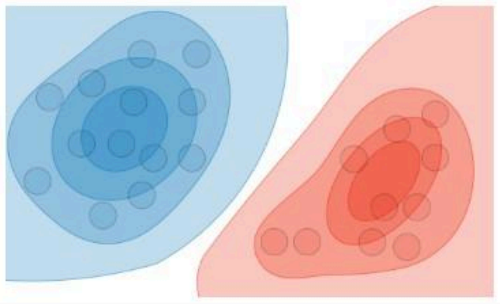
L2 Loss (Squared Loss): On the other hand, the L2 loss function computes the square of the differences between the predicted and actual values. This makes it more sensitive to outliers, as larger deviations are penalized more heavily. The use of L2 loss in generative models often results in smoother outcomes because it encourages smaller and more incremental changes in the model's parameters. This characteristic can be beneficial in tasks where the continuity of the output is critical, like generating realistic textures in images.

Preference and Application:

- Preference for L1 Loss: You might prefer L1 loss when the goal is to encourage more robustness to outliers in the dataset or when generating outputs where precise edges and details are important. Its tendency to produce sparser solutions can be particularly useful in applications requiring high detail fidelity, such as in certain types of image processing where sharpness is key.
- Preference for L2 Loss: L2 loss could be the preferred choice when aiming for smoother outputs and when dealing with problems where the Gaussian noise assumption is reasonable. Its sensitivity to outliers makes it suitable for tasks where the emphasis is on minimizing large errors, contributing to smoother and more continuous generative outputs.

Q13: How do discriminative and generative models differ in their learning approach, what are their advantages, and in what scenarios would each be preferred in deep learning applications?
Solution:



	Discriminative model	**Generative model**
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

Q14: How does Bayes's theorem ($P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$) establish the relationship between generative and discriminative models?

Solution:

The Approach of Generative Models

In the case of generative models, to find the conditional probability **P(Y|X)**, they estimate the priorprobability **P(Y)** and likelihood probability **P(X|Y)** with the help of the training data and use the Bayes Theorem to calculate the posterior probability **P(Y |X):**

$$posterior = \frac{prior \times likelihood}{evidence} \Rightarrow P(Y|X) = \frac{P(Y) \cdot P(X|Y)}{P(X)}$$

The Approach of Discriminative Models

In the case of discriminative models, to find the probability, they directly assume some functional form for **P(Y|X)** andthen estimate the parameters of **P(Y|X)** with the help of the training data.

The Mathematics of Discriminative Models

Training discriminative classifiers or discriminant analysis involves estimating a function **f: X -> Y**, or probability **P(Y|X)**

- Assume some functional form for the probability, such as **P(Y|X)**
- With the help of training data, we estimate the parameters of **P(Y|X)**

The Mathematics of Generative Models

Training generative classifiers involve estimating a function **f: X -> Y**, or probability **P(Y|X):**

- Assume some functional form for the probabilities such as **P(Y), P(X|Y)**
- With the help of training data, we estimate the parameters of **P(X|Y), P(Y)**
- Use the Bayes theorem to calculate the posterior probability **P(Y |X)**

Q15: What are the key probabilistic differences between generative models $(P(X, Y) or P(X))$ and discriminative models $(P(Y|X))$?

Solution:

Informally:

- **Generative** models can generate new data instances.
- **Discriminative** models discriminate between different kinds of data instances.

A generative model could generate new photos of animals that look like real animals, while a discriminative model could tell a dog from a cat. GANs are just one kind of generative model.

More formally, given a set of data instances X and a set of labels Y:

- **Generative** models capture the joint probability p(X, Y), or just p(X) if there are no labels.
- **Discriminative** models capture the conditional probability p(Y | X).

A generative model for images might capture correlations like "things that look like boats are probably going to appear near things that look like water" and "eyes are unlikely to appear on foreheads." These are very complicated distributions.

In contrast, a discriminative model might learn the difference between "sailboat" or "not sailboat" by just looking for a few tell-tale patterns. It could ignore many of the correlations that the generative model must get right.

Discriminative models try to draw boundaries in the data space, while generative models try to model how data is placed throughout the space. For example, the following diagram shows discriminative and generative models of handwritten digits:

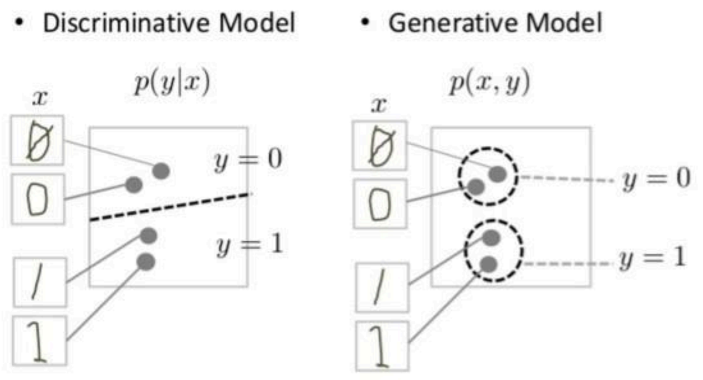
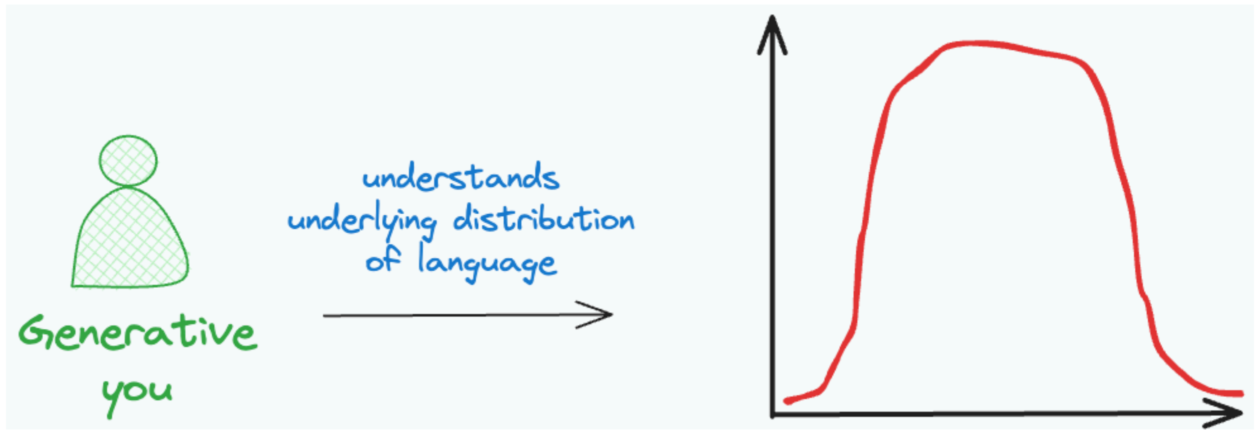


Figure 1: Discriminative and generative models of handwritten digits.

The discriminative model tries to tell the difference between handwritten 0's and 1's by drawing a line in the data space. If it gets the line right, it can distinguish 0's from 1's without ever having to model exactly where the instances are placed in the data space on either side of the line.

In contrast, the generative model tries to produce convincing 1's and 0's by generating digits that fall close to their real counterparts in the data space. It has to model the distribution throughout the data space.

<u>Q16: Solution:</u>
The first approach is generative. This is because you learned the underlying distribution of each language.
In other words, you learned the joint distribution P(Words, Language).

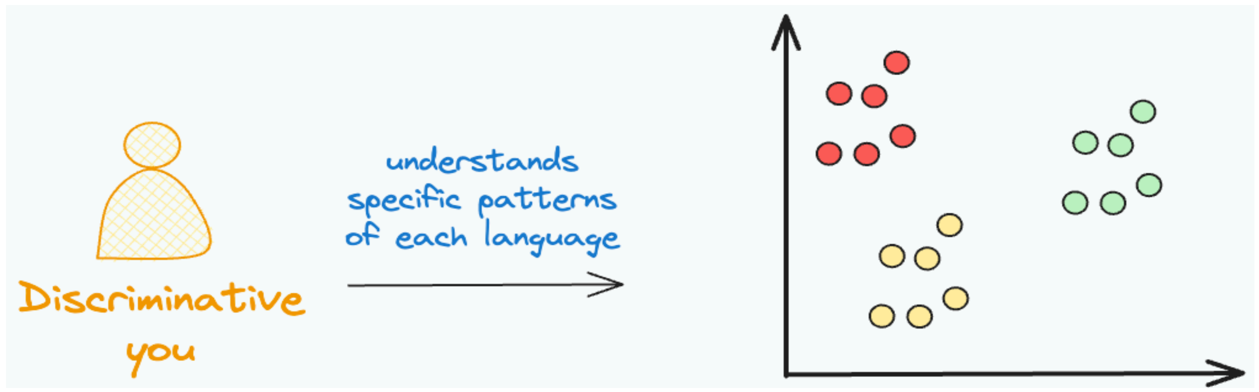


Moreover, as you understand the underlying distribution, now you can generate new sentences, can't you?

The second approach is discriminative. This is because you only learned specific distinctive patterns of each language.

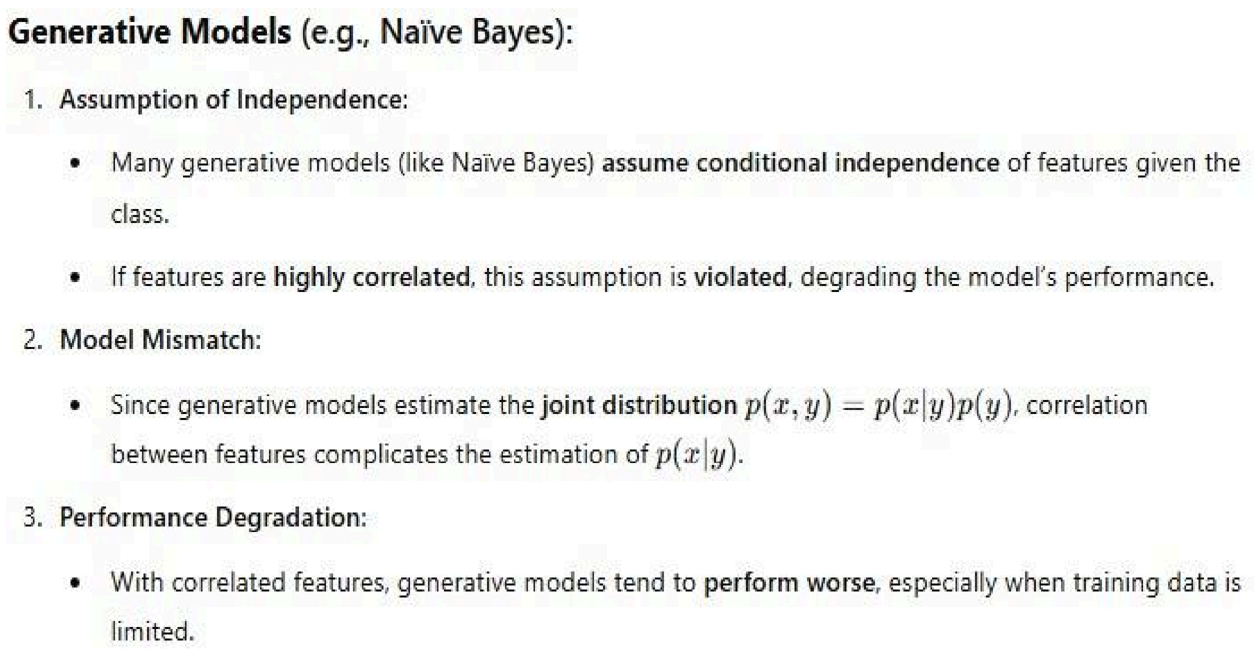
It is like:

- If so and so words appear, it is likely “Language A”.
- If this specific set of words appears, it is likely “Language B”.
- and so on.



In other words, you learned the conditional distribution P(Language|Words).

Q17: What happens when we give correlated features in generative and discriminative models?
Solution:
Impact of correlated features:



Generative Models (e.g., Naïve Bayes):

1. **Assumption of Independence:**
   - Many generative models (like Naïve Bayes) **assume conditional independence** of features given the class.
   - If features are **highly correlated**, this assumption is **violated**, degrading the model's performance.
2. **Model Mismatch:**
   - Since generative models estimate the **joint distribution** $p(x, y) = p(x|y)p(y)$, correlation between features complicates the estimation of $p(x|y)$.
3. **Performance Degradation:**
   - With correlated features, generative models tend to **perform worse**, especially when training data is limited.

Discriminative Models (e.g., Logistic Regression):

1. **Direct Conditional Modeling:**
   - These models learn $p(y|x)$ **directly**, without needing assumptions on the feature distribution.
2. **Robust to Correlations:**
   - Correlated features **do not harm** discriminative models as much, though they may lead to **redundancy** or **multicollinearity**, which can affect **model interpretability** or lead to **overfitting** if not regularized.
3. **Better Generalization:**
   - Typically outperform generative models on classification tasks with **correlated features**, especially with large datasets.

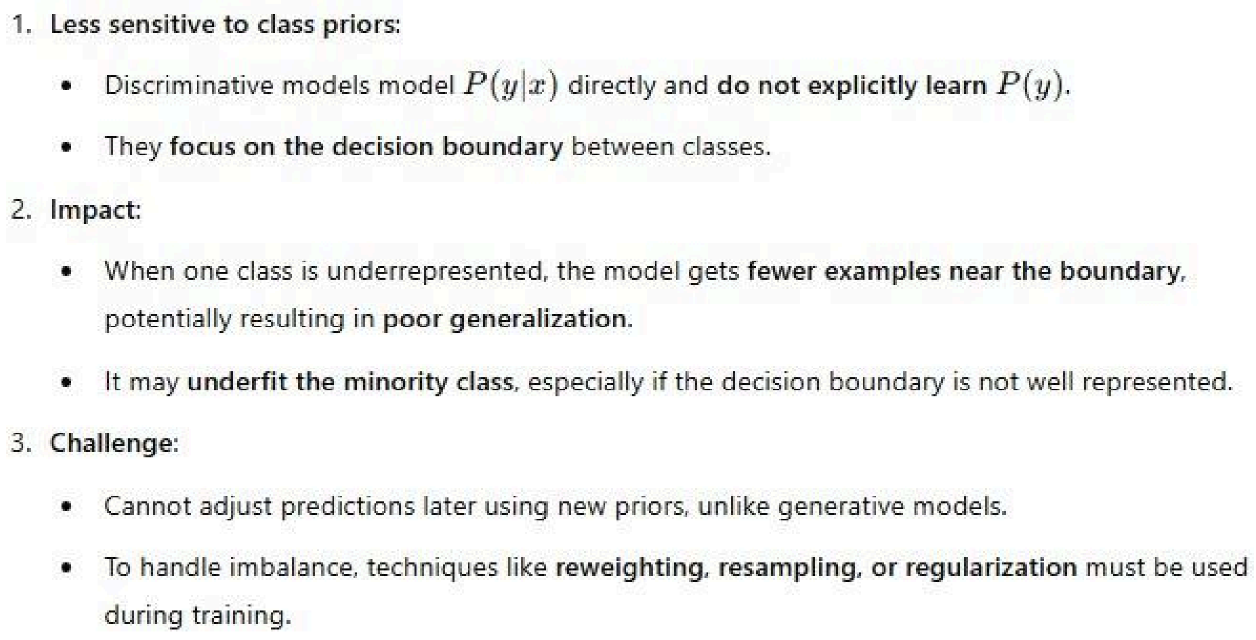
Q18: What happens when training data is biased over one class in generative and discriminative models?

Solution:

In generative models:

1. **Sensitive to class imbalance:**
   - Generative models estimate both $P(y)$ and $P(x|y)$.
   - If the training data is **biased toward one class**, the model **overestimates the prior** $P(y)$ for that class.
2. **Impact:**
   - This leads to **skewed predictions** because Bayes' rule combines the prior $P(y)$ and likelihood $P(x|y)$.
   - If class priors are not corrected, the classifier may **strongly favor the overrepresented class.**
3. **Advantage:**
   - If the **true prior** $P(y)$ is known or can be estimated separately, it can be **substituted**, mitigating bias.
   - Thus, **generative models can correct for training bias** by adjusting the priors externally.

In discriminative models:

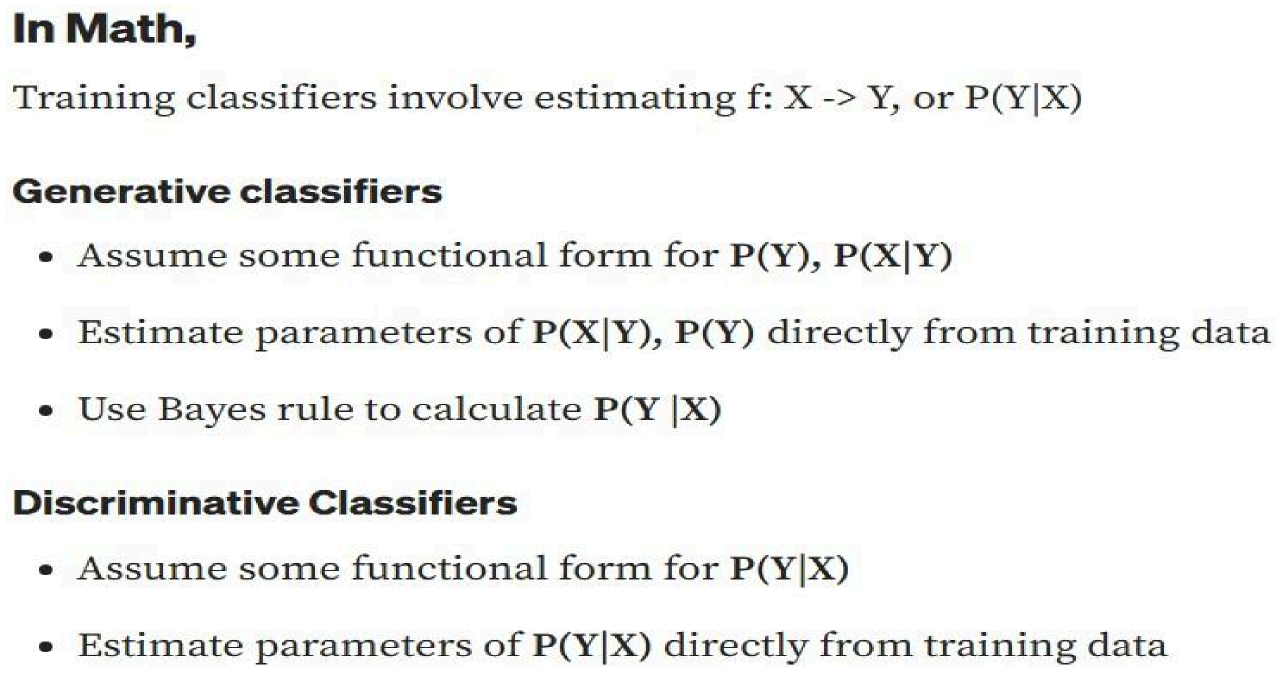


1. **Less sensitive to class priors:**
   - Discriminative models model $P(y|x)$ directly and **do not explicitly learn** $P(y)$.
   - They **focus on the decision boundary** between classes.
2. **Impact:**
   - When one class is underrepresented, the model gets **fewer examples near the boundary,** potentially resulting in **poor generalization.**
   - It may **underfit the minority class**, especially if the decision boundary is not well represented.
3. **Challenge:**
   - Cannot adjust predictions later using new priors, unlike generative models.
   - To handle imbalance, techniques like **reweighting, resampling, or regularization** must be used during training.

Q19: How do generative and discriminative classifiers differ in terms of the probability distributions they model and the assumptions they make during training?

Solution:

A Generative Model learns the joint probability distribution p(x,y). It predicts the conditional probability with the help of Bayes Theorem. A Discriminative model learns the conditional probability distribution p(y|x). Both of these models were generally used in supervised learning problems.



In Math,

Training classifiers involve estimating f: X -> Y, or P(Y|X)

Generative classifiers

- Assume some functional form for **P(Y), P(X|Y)**
- Estimate parameters of **P(X|Y), P(Y)** directly from training data
- Use Bayes rule to calculate **P(Y |X)**

Discriminative Classifiers

- Assume some functional form for **P(Y|X)**
- Estimate parameters of **P(Y|X)** directly from training data

Examples:
Generative classifiers

- Naïve Bayes
- Bayesian networks
- Markov random fields
- Hidden Markov Models (HMM)

Discriminative Classifiers

- Logistic regression
- Scalar Vector Machine
- Traditional neural networks
- Nearest neighbour
- Conditional Random Fields (CRF)s