**1. How do generative and discriminative classifiers differ? Examples? When to prefer generative classifiers?**

* **Difference**: Generative classifiers model the data distribution P(X,Y)P(X, Y)P(X,Y), while discriminative classifiers focus only on the decision boundary P(Y∣X)P(Y|X)P(Y∣X).
* **Examples**: Generative - Naive Bayes, GMM; Discriminative - Logistic Regression, SVM.
* **When to prefer**: Generative classifiers are better when the dataset is small, when you need probabilistic outputs, or when handling missing data.

**2. How is K-Fold Cross Validation used to find the best hyperparameters for a model?**

* The dataset is split into KKK equal parts (folds).
* Train on K−1K-1K−1 folds and validate on the remaining one, repeating this process KKK times.
* Hyperparameters are tuned based on the average validation score across folds, ensuring better generalization.

**3. How to handle class imbalance problems for a binary classifier?**

* **Resampling techniques**: Oversample the minority class (e.g., SMOTE) or undersample the majority class.
* **Adjust class weights**: Use algorithms that allow weighted loss functions to penalize errors in the minority class more.
* **Use specialized models**: Techniques like ensemble methods (e.g., boosting) inherently handle imbalance well.

**4. Explain ROC and AUC. How to compute AUC for a multi-class classifier?**

* **ROC**: A curve showing the tradeoff between the true positive rate (sensitivity) and false positive rate at different thresholds.
* **AUC**: The area under the ROC curve; measures how well the model separates classes (1 is perfect, 0.5 is random).
* **Multi-class AUC**: Compute one-vs-all AUC for each class or average pairwise comparisons between classes.

**5. When to use the Expectation-Maximization algorithm and why?**

* Use it when:
  + You’re working with data that has hidden or latent variables (e.g., GMMs for clustering).
  + Data is incomplete or has missing values.
* **Why**: EM alternates between assigning probabilities to hidden variables (E-step) and updating model parameters (M-step) to maximize the likelihood.

**6. How to choose the optimum number of clusters with K-Means clustering?**

* **Elbow Method**: Plot the WCSS (within-cluster sum of squares) against the number of clusters. The "elbow point" indicates the optimal number.
* **Silhouette Score**: Measures how similar points in a cluster are to points in other clusters (higher is better).
* **Gap Statistic**: Compares clustering performance to random data distribution.

**7. How does bias and variance change with an increasing number of clusters in K-Means?**

* **More clusters**: Reduces bias as the model can better fit the data, but increases variance, leading to overfitting.
* **Fewer clusters**: Increases bias, as it oversimplifies the data, but reduces variance, improving generalization.

**8. What are the advantages of Gaussian Mixture Models (GMMs) over K-Means?**

* GMMs can model **non-spherical clusters**, unlike K-Means, which assumes clusters are spherical.
* GMMs provide **soft clustering**, assigning probabilities to each cluster for a point, rather than a hard assignment.
* They handle **uncertainty** in data better.

**9. When would you choose Hierarchical Clustering over K-Means and vice versa?**

* **Hierarchical**:
  + When you need a hierarchy or dendrogram to understand relationships between clusters.
  + For smaller datasets since it’s computationally expensive for large datasets.
* **K-Means**:
  + When you know the number of clusters and have large datasets.
  + It’s faster and works well with spherical clusters.

**10. What is the ‘Naive’ Bayes assumption?**

* It assumes that all features are **conditionally independent** given the class label.
* This simplifies computation by treating the probability of each feature independently.
* While this assumption is rarely true in real-world data, Naive Bayes often performs surprisingly well due to its simplicity and efficiency.

**11. You come to know that your model is suffering from low bias and high variance. Which algorithm should you use to tackle it? Why?**

* Use **ensemble methods like Random Forest** or **boosting** to reduce variance.
* **Why**: These methods aggregate predictions from multiple models, reducing overfitting. Adding regularization (e.g., L1/L2) can also help.

**12. Should you remove correlated variables before running PCA? If not, how does PCA handle them?**

* No need to remove correlated variables.
* **Why**: PCA identifies correlated features and combines them into principal components, effectively reducing redundancy.

**13. How do you interpret p-value for a model?**

* The p-value tells whether a feature's coefficient significantly contributes to the model.
* A small p-value (e.g., < 0.05) indicates strong evidence to reject the null hypothesis, meaning the feature is significant.

**14. What is the Central Limit Theorem? How does it help in ML?**

* **CLT**: For large sample sizes, the sample mean follows a normal distribution, regardless of the population’s distribution.
* **In ML**: It’s the basis for confidence intervals and hypothesis testing, helping with assumptions about data distributions.

**15. What is the difference between covariance and correlation?**

* **Covariance**: Measures how two variables vary together (scale-dependent).
* **Correlation**: A normalized version of covariance, showing the strength and direction of the relationship (range: -1 to 1).

**16. What do you understand by eigenvectors and eigenvalues?**

* **Eigenvectors**: Directions in which data variance is maximized.
* **Eigenvalues**: Amount of variance captured along an eigenvector.
* **In ML**: Used in PCA to identify key dimensions in data.

**17. How do you know whether a loss function has a global minima or not?**

* If the loss function is **convex** (e.g., Mean Squared Error), it has a single global minimum.
* For non-convex functions, optimization techniques like SGD may find local minima or saddle points.

**18. How does SVM handle non-linear classification problems? Explain the Kernel Trick.**

* SVM maps data to a higher-dimensional space using kernels (e.g., RBF, polynomial) to make it linearly separable.
* **Kernel Trick**: Computes inner products in higher-dimensional space efficiently, avoiding explicit transformations.

**19. What is the role of the misclassification cost parameter CCC in SVM?**

* **C** controls the tradeoff between maximizing the margin and minimizing misclassification errors.
* A **high C** prioritizes correct classification (low bias, high variance).
* A **low C** increases the margin (high bias, low variance).

**20. How do you find efficiently whether two classes are linearly separable?**

* Use a **linear classifier** like Logistic Regression or Linear SVM.
* If they achieve perfect accuracy during training, the classes are likely linearly separable.

**21. When would you prefer LinearSVM over Logistic Regression for a linearly separable binary classification problem?**

* Prefer **LinearSVM** when:
  + You need robust performance with outliers.
  + You prioritize maximizing the margin (better generalization).

**22. For ranking recommendations, which would you choose: LinearSVM or Logistic Regression?**

* **Logistic Regression**: Better for ranking as it outputs probabilities, which can be used to rank items.

**23. Would SVM work well in high dimensions? Why or why not?**

* Yes, SVM works well in high dimensions because:
  + The kernel trick avoids explicit feature transformations.
  + It’s less prone to overfitting due to the margin-based approach.

**24. What is the role of gamma in RBF kernels?**

* **Gamma** controls the influence of a single data point.
* A **small gamma** captures global patterns, while a **large gamma** focuses on local patterns.

**25. Why do we need a large margin classifier in SVM? Why wouldn’t any margin classifier work well?**

* A larger margin improves generalization by reducing overfitting.
* Smaller margins can lead to overfitting, especially with noisy data.

**26. How does a decision tree split a node?**

* It uses metrics like **Gini Index** or **Information Gain** to find the best split that maximizes class separation.

**27. How to handle overfitting with a decision tree classifier?**

* **Techniques**:
  + Limit tree depth.
  + Set minimum samples per leaf node.
  + Use pruning (removing unnecessary branches).

**28. Can decision trees be used for regression problems?**

* Yes, decision trees can predict continuous values by minimizing **variance** within splits instead of classification metrics.

**29. How do Random Forests handle bias-variance tradeoffs?**

* By combining multiple decision trees:
  + Reduces variance (averaging predictions).
  + Maintains low bias (individual trees can capture complexity).

**30. How are Gradient Boosted Trees different from Random Forests?**

* **Random Forest**: Builds trees independently and averages them.
* **Gradient Boosting**: Builds trees sequentially, each focusing on errors of the previous one.

**31. How to handle overfitting in Gradient Boosted Trees?**

* Techniques:
  + Use **early stopping** to halt training when validation error increases.
  + Add **regularization** (e.g., learning rate, max depth, L1/L2 penalties).
  + Reduce the **number of trees** or increase the **minimum samples per leaf**.

**32. How to handle both numerical and categorical features with tree-based algorithms? What about ordinal features?**

* **Numerical features**: Used as-is since tree algorithms split data on numeric thresholds.
* **Categorical features**: Convert to numerical using techniques like one-hot encoding or target encoding.
* **Ordinal features**: Use their natural ordering directly in the tree.

**33. Why and how does Random Forest prevent overfitting in decision trees?**

* **Why**: Random Forest reduces overfitting by combining predictions from multiple trees.
* **How**:
  + Uses random subsets of data (bagging).
  + Splits nodes based on a random subset of features, reducing correlation between trees.

**34. How does GBDTs decide to split a node? What does it minimize?**

* GBDTs split nodes to minimize the **loss function** (e.g., Mean Squared Error for regression, Log Loss for classification).
* Each tree focuses on reducing the residual errors of the previous tree.

**35. How to find feature importances with GBDTs?**

* Measure the **reduction in the loss function** from splitting on a feature across all trees.
* Features contributing more to splits have higher importance scores.

**36. How can GBDTs be used for feature transformation?**

* Use GBDT predictions (leaf node outputs) as new features for another model.
* Commonly used in ensemble stacking.

**37. What are the regularization factors in GBDTs?**

* **Learning rate**: Reduces the contribution of each tree.
* **Max depth**: Prevents overly complex trees.
* **Min samples per leaf**: Ensures splits have enough data points.
* **L1/L2 regularization**: Penalizes large weights in the model.

**38. How would you compute PCA of a feature matrix XXX?**

* Center the data (subtract the mean of each feature).
* Compute the covariance matrix.
* Find eigenvalues and eigenvectors of the covariance matrix.
* Project data onto the top kkk eigenvectors (principal components).

**39. What is the difference between PCA and SVD?**

* **PCA**: A dimensionality reduction technique based on eigenvectors of the covariance matrix.
* **SVD**: A matrix factorization technique that decomposes any matrix XXX into UΣVTU \Sigma V^TUΣVT. PCA is a special case of SVD applied to centered data.

**40. How would you determine how many principal components to consider in PCA?**

* Look for the **cumulative explained variance** (e.g., select components explaining 95% of the variance).
* Use a scree plot to identify the "elbow point."

**41. Describe a situation where PCA is not a good method for dimensionality reduction.**

* When features are not linearly related (PCA assumes linear relationships).
* When interpretability of features is crucial (PCA transforms features into combinations).

**42. When do we need to standardize the variables before doing PCA?**

* Always standardize when features are on different scales (e.g., age in years and income in dollars).
* Standardizing ensures that PCA doesn’t give undue weight to features with larger variances.

**43. How would you use LSTM for Named Entity Recognition problems?**

* Use an LSTM to capture sequential dependencies in text.
* Add a dense layer for classification at each time step to label entities.
* Optionally, combine with CRFs for better sequence-level predictions.

**44. Which one should you prefer for NER: LSTM only, Linear Chain CRF only, or LSTM + Linear Chain CRF? Why?**

* Prefer **LSTM + CRF**:
  + LSTM captures contextual information.
  + CRF ensures label consistency (e.g., avoids invalid sequences like "I-PER" followed by "B-LOC").

**45. What problem does Bi-LSTM solve compared to only LSTM?**

* Bi-LSTM captures information from both past (left context) and future (right context), improving predictions for tasks like NER or translation.

**46. What is the purpose of the pooling operation in CNNs?**

* Reduces spatial dimensions, retaining important features while making the model more computationally efficient.
* Common types: Max Pooling (retains max value), Average Pooling (computes average).

**47. How would you choose the number of filters and the filter size at each CNN layer?**

* **Number of filters**: Start small (e.g., 32 or 64) and increase for deeper layers.
* **Filter size**: Use smaller filters (e.g., 3x3 or 5x5) to capture finer details without overfitting.

**48. How does Conv1D and MaxPool1D work?**

* **Conv1D**: Applies filters to 1D data (e.g., time series or text).
* **MaxPool1D**: Reduces dimensionality by selecting the maximum value within a sliding window.

**49. What are the advantages of parameter sharing in CNNs?**

* Reduces the number of parameters, making training faster and less prone to overfitting.
* Captures spatial hierarchies efficiently.

**50. How does CNN help in translation and rotation invariance of images?**

* Translation invariance: Achieved through local receptive fields and pooling layers.
* Rotation invariance: Not inherent but can be achieved through data augmentation or specialized architectures.

**51. How would you choose which layers to freeze and which to retrain in transfer learning?**

* **Freeze early layers**: These layers capture general features like edges and textures, which are useful across tasks.
* **Retrain later layers**: These are task-specific and should be updated to learn features relevant to the new dataset.
* If the new dataset is large, retrain more layers. For small datasets, freeze most layers to avoid overfitting.

**52. What are some advantages of using character embeddings instead of word embeddings?**

* **Handle out-of-vocabulary words**: Character embeddings can represent unseen words by breaking them into subword units.
* **Better for morphologically rich languages**: They capture prefixes, suffixes, and roots.
* **Robust to typos**: Since they work at the character level, small spelling mistakes won’t significantly affect results.

**53. How to train CNN models in parallel? Can LSTM models be trained in parallel? Why or why not?**

* **CNNs**: Can be trained in parallel because convolutions are independent for different filters and images.
* **LSTMs**: Training is sequential because each step depends on the output of the previous step, making parallelization challenging.

**54. Why are large filter sizes in early CNN layers a bad choice? How to choose filter size?**

* **Large filters**: They capture too much detail early, making the model less focused on small, meaningful patterns.
* **Better choice**: Use smaller filters (e.g., 3x3) to capture local features and stack multiple layers to detect complex patterns.

**55. Why are weights initialized with small random numbers in a neural network? What happens when weights are all 0 or constant values?**

* **Small random initialization**: Helps break symmetry so neurons learn diverse features.
* **All zeros**: Leads to identical updates for all neurons in a layer, preventing them from learning independently.
* **Constant values**: Causes a similar problem, as gradients will remain the same for all weights.

**56. Why is the sigmoid activation not good? Why is ReLU or Tanh preferred?**

* **Sigmoid drawbacks**:
  + Outputs saturate at extremes, leading to **vanishing gradients**.
  + Produces outputs between 0 and 1, which are not zero-centered.
* **ReLU advantages**: Simpler computation and avoids saturation for positive inputs.
* **Tanh**: Zero-centered outputs, but still suffers from vanishing gradients for large inputs.

**57. How to handle the dying ReLU problem in activation functions?**

* Dying ReLU occurs when neurons output zero for all inputs, effectively “dead.”
* **Solutions**:
  + Use variants like Leaky ReLU (small positive slope for negative inputs).
  + Apply Batch Normalization to maintain gradient flow.

**58. How does Dropout help in regularization? How is it different from L1 or L2 regularization?**

* **Dropout**: Randomly disables neurons during training, forcing the network to learn more robust features.
* **Difference**:
  + L1/L2 constrain weights to avoid overfitting.
  + Dropout prevents reliance on specific neurons, improving generalization.

**59. When and why use Dropout instead of L1 or L2 regularization?**

* Use Dropout when:
  + The model is very deep and prone to overfitting.
  + You want to improve robustness by introducing noise during training.
* L1/L2 are better for simpler models or when sparsity (L1) is desired.

**60. When and why use Batch Normalization?**

* Use Batch Normalization to:
  + Normalize layer inputs, improving stability and convergence.
  + Reduce the dependence on careful weight initialization.
  + Act as a form of regularization, reducing overfitting.

**61. How to handle vanishing gradient problems in neural networks?**

* **Solutions**:
  + Use activation functions like ReLU that don’t saturate.
  + Apply Batch Normalization to maintain consistent gradients.
  + Use residual connections (ResNets) to allow gradients to flow directly.

**62. Why do we need the bias term in neural networks?**

* The bias term shifts the activation function, enabling the model to fit data that doesn’t pass through the origin.
* Without bias, the network would have limited flexibility in fitting complex patterns.

**63. What are the advantages and disadvantages of SGD over gradient descent?**

* **Advantages**:
  + Faster updates as it uses a single sample at a time.
  + Introduces noise, helping escape local minima.
* **Disadvantages**:
  + Noisy updates can lead to slower convergence.
  + Requires careful tuning of the learning rate.

**64. How does momentum help in SGD?**

* Momentum smoothens updates by adding a fraction of the previous update to the current one.
* This accelerates convergence, especially in directions with consistent gradients, and helps overcome small local minima.

**65. Would you use squared error loss or binary cross-entropy for binary classification? Why?**

* **Binary cross-entropy** is preferred because it:
  + Better handles probabilities and log-likelihood.
  + Avoids issues like vanishing gradients seen with squared error in this setting.

**66. For online learning, which would you prefer: SGD or Adagrad? Why?**

* Prefer **Adagrad** for online learning because:
  + It adapts the learning rate for each parameter, reducing manual tuning.
  + Effective for sparse data as it adjusts learning rates dynamically.

**67. How to train deep neural networks in a distributed manner? What are the advantages and disadvantages?**

* **Methods**:
  + **Data parallelism**: Split data across machines, each training a copy of the model.
  + **Model parallelism**: Split the model itself across machines.
* **Advantages**:
  + Faster training by utilizing multiple machines.
* **Disadvantages**:
  + Communication overhead between machines.
  + Complexity in synchronization.

**68. How to handle the exploding gradient problem?**

* Use **gradient clipping** to cap gradients at a threshold.
* Apply proper initialization techniques (e.g., Xavier or He initialization).
* Use normalized architectures like Batch Normalization.

**69. How does Batch Normalization differ during training and inference?**

* **Training**: Uses mini-batch statistics (mean, variance) for normalization.
* **Inference**: Uses the moving average of batch statistics collected during training.

**70. Why don’t we use Dropout during inference?**

* Dropout introduces randomness by disabling neurons, which is only useful during training.
* During inference, all neurons are used, and Dropout scaling ensures consistent outputs.

**71. Why do we need to shuffle data during training?**

* **Shuffling ensures that the model doesn’t learn patterns based on the order of data (e.g., all "class A" samples appearing first followed by "class B").**
* **Example: If training data for a stock price model is time-ordered, the model might mistakenly infer trends from order rather than data.**

**72. How can we alter the learning rate depending on the training loss? Is it okay to have a constant learning rate?**

* **Adaptive learning rates reduce overfitting or overshooting as the model converges. Common methods include:**
  + **Reduce learning rate by a factor when validation loss plateaus.**
* **Example: Use a scheduler in PyTorch or TensorFlow that reduces the learning rate by half after 3 epochs of no improvement in validation loss.**

**73. For distributed training with KKK machines/cores, should we use a higher or lower learning rate? Why?**

* **Use a higher learning rate because the gradient updates from KKK machines are averaged, effectively scaling down the step size.**
* **Example: When training ResNet on 8 GPUs, start with a learning rate of 0.08 instead of 0.01 (single GPU default).**

**74. How does batch size affect training of neural networks? What if we choose a batch size of 1?**

* **Small batch sizes (e.g., 16):**
  + **Introduce noise in gradient updates, which can help avoid local minima but may slow down training.**
* **Large batch sizes (e.g., 512):**
  + **More stable gradients but may lead to overfitting.**
* **Batch size of 1: Leads to noisy updates and inefficient hardware usage.**
* **Example: For image classification, start with 64 or 128. Using 1 would be inefficient unless processing very large images (e.g., medical imaging).**

**75. How is Word2Vec different from GloVe?**

* **Word2Vec:**
  + **Learns embeddings by predicting context words (Skip-Gram) or target words (CBOW).**
* **GloVe:**
  + **Learns embeddings from a co-occurrence matrix.**
* **Example:**
  + **In Word2Vec: "king - man + woman = queen" because embeddings encode semantic relationships.**
  + **GloVe uses global co-occurrence, capturing broader contextual relationships.**

**76. For rare words, which should be used for Word2Vec training: CBOW or Skip-Gram?**

* **Use Skip-Gram: It predicts context for each word, making it better for infrequent terms.**
* **Example: Words like "epitome" or "taciturn" (rare) are better learned via Skip-Gram as CBOW averages the context, which may dilute rare-word meaning.**

**77. Is it possible for both validation loss and validation accuracy to increase?**

* **Yes, if the model becomes overconfident, increasing certainty on correct predictions (accuracy) while also making larger errors on wrong predictions (higher loss).**
* **Example: In image classification, the model misclassifies a "cat" as a "dog" with 95% confidence, increasing loss even if accuracy improves overall.**

**78. What can go wrong if we use a linear activation instead of ReLU?**

* **Linear activations limit the network to simple linear relationships.**
* **Example: Predicting housing prices with features like location, size, and amenities would fail to capture complex non-linear dependencies if only linear activations are used.**

**79. How does Item-Based Collaborative Filtering (CF) work in recommendations? What if the number of items is in billions?**

* **Finds similar items based on user interactions and recommends them.**
* **Example: On Amazon, if you bought a "smartphone," CF might suggest screen protectors and cases bought by others who purchased the same phone.**
* **For billions of items, use approximate nearest neighbor techniques like Locality Sensitive Hashing (LSH) to speed up similarity searches.**

**80. When would you choose Item-Based CF over User-Based CF?**

* **Use Item-Based CF when:**
  + **Users are transient (e.g., anonymous visitors on e-commerce sites).**
  + **Items have fewer changes compared to users.**
* **Example: In a bookstore, comparing books is easier because the catalog changes slowly, while user preferences can vary greatly.**

**81. How is matrix factorization useful in recommendation systems?**

* **Decomposes user-item interactions into latent features representing user preferences and item properties.**
* **Example: In Netflix, matrix factorization might reveal that a user who likes "action" movies will also like a new "superhero" movie, even without direct ratings.**

**82. What are the advantages and disadvantages of SGD over ALS in matrix factorization?**

* **SGD:**
  + **Advantage: Can handle large, sparse data incrementally.**
  + **Disadvantage: Requires tuning the learning rate.**
* **ALS:**
  + **Advantage: Faster for batch processing.**
  + **Disadvantage: Computationally expensive for dynamic or very sparse data.**
* **Example: Use SGD for real-time updates in a news recommendation system.**

**83. How would you find KKK nearest neighbors efficiently with billions of data?**

* **Use approximate methods like LSH or KD-Trees to reduce search complexity.**
* **Example: Finding similar users for recommendations on Spotify with billions of tracks and users requires hashing-based approximations.**

**84. How does Locality Sensitive Hashing (LSH) work for finding nearest neighbors? What hash function is used?**

* **How it works: Maps similar points into the same hash bucket using hash functions sensitive to proximity.**
* **Example: For cosine similarity, use random projection hash functions to cluster text documents or user preferences.**

**85. Why is the choice of activation function important in neural networks?**

* **Activation functions enable the model to capture non-linear patterns.**
* **Example: Using ReLU in image classification ensures the network captures complex features like edges, textures, and patterns beyond linear relationships.**

**86. Why does overfitting occur in neural networks, and how can it be prevented?**

* **Why: The model memorizes training data, including noise, instead of learning generalizable patterns.**
* **Prevention:**
  + **Regularization (L1/L2 or Dropout).**
  + **Data augmentation to increase training diversity.**
* **Example: In training a model on cat images, flipping and rotating images during augmentation can prevent overfitting.**

**87. How do Autoencoders differ from PCA for dimensionality reduction?**

* **Autoencoders: Non-linear, learn complex relationships through a neural network.**
* **PCA: Linear, identifies orthogonal directions of maximum variance.**
* **Example: Autoencoders are better for reducing dimensionality in high-resolution image datasets with non-linear patterns.**

**88. What is the main goal of using Attention Mechanisms in NLP?**

* **Attention mechanisms focus on important parts of the input while generating output.**
* **Example: In translation, attention ensures the model emphasizes words like "gato" (cat) in Spanish when generating "cat" in English.**

**89. How do Transformers improve upon RNNs for sequence tasks?**

* **Advantages:**
  + **Parallel processing through self-attention.**
  + **Better handling of long-range dependencies.**
* **Example: In sentiment analysis, Transformers handle long reviews more effectively than RNNs, which struggle with long dependencies.**

**90. Why are pre-trained models like BERT or GPT important?**

* **They provide general language understanding that can be fine-tuned for specific tasks.**
* **Example: Fine-tune BERT for sentiment analysis to classify tweets as positive or negative.**