Lecture: Advanced Concepts in Pattern Recognition and Anomaly Detection

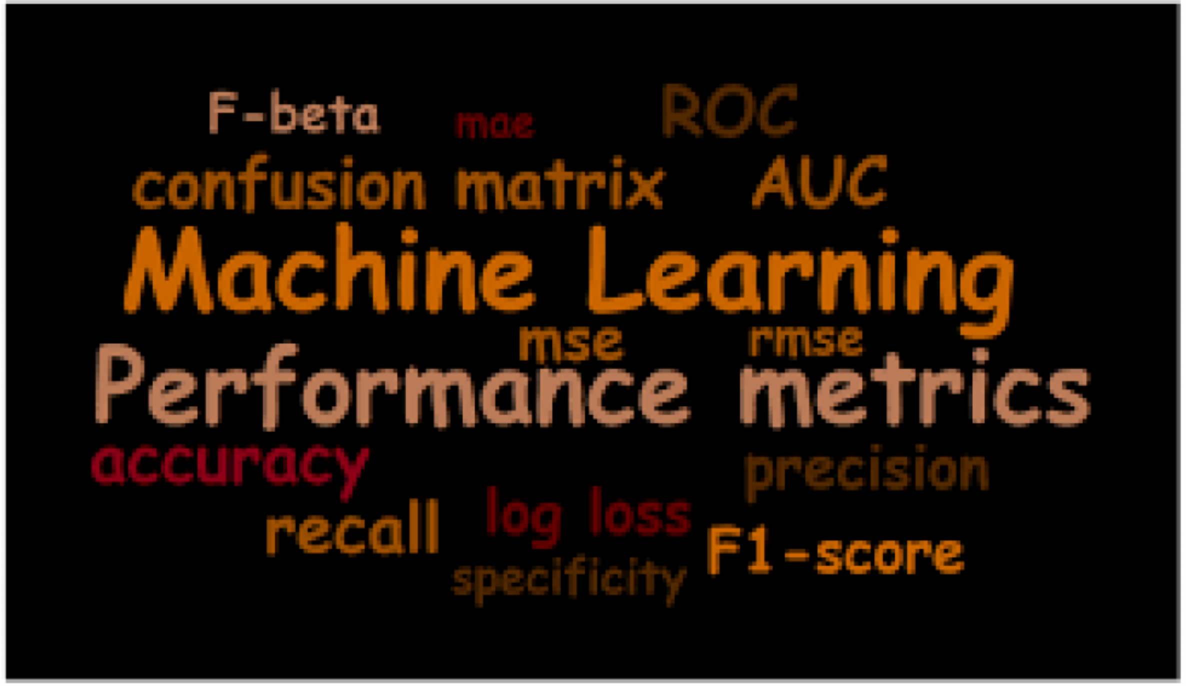
Dr. Amit Ranjan

School of Computer Science

Metric and Hyperparameter Selection

Evaluation Metrics for Different Tasks

- Choosing the right evaluation metric is crucial for assessing model performance.
- Different tasks require different metrics.



Metric and Hyperparameter Selection

Classification Metrics

Example: Email Spam Detection

- **Accuracy**: Measures how many emails were correctly classified as spam or not spam. (Works well when classes are balanced)
- **Precision**: Important when false positives (legitimate emails classified as spam) need to be minimized.
- **Recall**: Important when false negatives (spam emails classified as legitimate) need to be minimized.
- **F1-Score**: Used when there is an imbalance between spam and non-spam emails.
- **ROC-AUC**: Measures how well the model distinguishes between spam and non-spam emails.

Metric and Hyperparameter Selection

Regression Metrics

Example: Predicting House Prices

MAE (Mean Absolute Error): Measures the average absolute difference between actual and predicted house prices.

MSE (Mean Squared Error): Penalizes large errors more than MAE.

RMSE (Root Mean Squared Error): The square root of MSE, useful for understanding errors in original price scale.

R^2 Score: Measures how well the model explains the variance in house prices.

Hyperparameter Selection

- Selecting the right hyperparameters significantly affects model performance.
- Hyperparameters control how models learn.
- Optimizing them improves performance.

Grid Search

- Exhaustive search over a manually defined hyperparameter space.
- Computationally expensive but finds optimal parameters.
- Tries all combinations of given hyperparameters.
- **Example: Tuning an SVM Classifier for Image Recognition**

Hyperparameter Selection

Random Search

- Instead of evaluating all possible values, samples random combinations.
- It is faster than Grid Search, especially when only a subset of hyperparameters significantly affects model performance.
- It works well when the search space is large and only a few parameters are important.
- It allows for exploration across a wider range of values, increasing the chance of finding an optimal configuration.

Example: Tuning an XGBoost Model for Fraud Detection

Hyperparameter Selection

Bayesian Optimization

- Bayesian Optimization is a probabilistic model-based optimization technique for finding the minimum (or maximum) of an objective function that is expensive to evaluate to find the best hyperparameters efficiently.
- **Efficient**: Works well when objective function evaluations are expensive.
- **Probabilistic**: Provides uncertainty estimates for predictions.
- **Global Optimization**: Can escape local minima due to the exploration-exploitation tradeoff.
- Example libraries: optuna, scikit-optimize.

Curse of Dimensionality

As the number of features increases, data becomes sparse, making models ineffective.

Effects of High Dimensionality

- **Distance Measures Become Less Meaningful**:
    - Euclidean distances become nearly identical for high dimensions.
- **Overfitting**:
    - Models may memorize training data instead of generalizing.
- **Computational Complexity**:
    - More dimensions require more memory and time.

Example: Face Recognition with High-Dimensional Images

- As image resolution increases, each pixel becomes a feature.
- Distance-based algorithms like k-NN perform worse because distances become similar in high dimensions.

Dimensionality Reduction

Principal Component Analysis (PCA)

PCA transforms high-dimensional data into a lower-dimensional space while preserving variance.

- **Mathematical Foundation**:
  - Computes eigenvectors and eigenvalues of the covariance matrix.
  - Projects data onto principal components that explain maximum variance.
- **Formula**:
  - Let X be the data matrix, $C = X^T X$ be the covariance matrix.
  - Eigenvalues λ and eigenvectors v satisfy $Cv = \lambda v$.

Example: Reducing Features in Genomic Data

DNA sequences have thousands of features.

PCA helps reduce dimensions while retaining information.

Dimensionality Reduction

1. **Standardize the Data**:
   - Compute the mean μ and standard deviation σ for each feature.
   - Standardize each feature:
     $X_{standardized} = \frac{X-\mu}{\sigma}$
2. **Compute the Covariance Matrix**:
   - Covariance matrix C is computed as:
     $C = \frac{1}{n}X^T X$
3. **Compute Eigenvalues and Eigenvectors**:
   - Solve for eigenvalues λ and eigenvectors v:
     $Cv = \lambda v$
   - The eigenvectors represent principal components.
4. **Sort and Select Principal Components**:
   - Rank eigenvectors by corresponding eigenvalues.
   - Select top k eigenvectors to form transformation matrix.
5. **Project Data onto New Space**:
   - Transform data using selected principal components:
     $X_{reduced} = X_{standardized}W_k$

Dimensionality Reduction

- **Question:** Given the dataset:

Sample	Feature 1	Feature 2
A	2	3
B	3	4
C	4	5

- **Solution:**

1. Compute the mean of each feature.
2. Center the data by subtracting the mean.
3. Compute the covariance matrix.
4. Compute eigenvalues and eigenvectors.
5. Select principal component(s) and project the data.

Dimensionality Reduction

Solution:

1. **Compute the mean of each feature:**
    - Mean of Feature 1 = (2+3+4)/3 = 3
    - Mean of Feature 2 = (3+4+5)/3 = 4
2. **Center the data by subtracting the mean:**

Sample	Feature 1 (Centered)	Feature 2 (Centered)
A	-1	-1
B	0	0
C	1	1

3. **Compute the covariance matrix:**

   - The covariance between two features is given by:

$$cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

   - Computing covariance for the given data:

$$cov(Feature1, Feature1) = \frac{1}{3-1}((-1 \times -1) + (0 \times 0) + (1 \times 1)) = \frac{1}{2}(1 + 0 + 1) = 1$$

$$cov(Feature1, Feature2) = \frac{1}{3-1}((-1 \times -1) + (0 \times 0) + (1 \times 1)) = \frac{1}{2}(1 + 0 + 1) = 1$$

$$cov(Feature2, Feature2) = \frac{1}{3-1}((-1 \times -1) + (0 \times 0) + (1 \times 1)) = \frac{1}{2}(1 + 0 + 1) = 1$$

   - Therefore, the covariance matrix is:

$$C = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

4. **Compute eigenvalues and eigenvectors:**
   - Solve for λ in $det(C - \lambda I) = 0$:

$$\begin{vmatrix} 1-\lambda & 1 \\ 1 & 1-\lambda \end{vmatrix} = 0$$

   Expanding the determinant:

$$(1-\lambda)(1-\lambda) - 1 = \lambda^2 - 2\lambda = 0$$

   Solving for λ:

$$\lambda_1 = 2, \quad \lambda_2 = 0$$

   - Compute eigenvectors by solving $(C - \lambda I)v = 0$:
     - For $\lambda_1 = 2$:

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

     Choosing $v_1 = 1$, we get $v_2 = 1$, so the eigenvector is:

$$\begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}$$

Dimensionality Reduction

5. **Select principal component(s) and project the data:**

- The principal component is chosen as the eigenvector corresponding to the highest eigenvalue, which is $\lambda_1 = 2$.
- The projection of the centered data onto the principal component is calculated as:

$$X_{new} = X_{centered} \times V$$

Expanding:

$$\begin{bmatrix} -1 & -1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix} = \begin{bmatrix} -1.414 \\ 0 \\ 1.414 \end{bmatrix}$$

- The transformed data in one-dimensional space is:

$$(-1.414, 0, 1.414)$$

Dimensionality Reduction

t-SNE (t-Distributed Stochastic Neighbor Embedding)

- Non-linear technique preserving local similarities.
- Suitable for visualizing complex high-dimensional data.
- **Mathematical Intuition**:
- Converts high-dimensional distances into probabilities.
- Minimizes Kullback-Leibler (KL) divergence between high-dimensional and low-dimensional distributions.

Example: Visualizing High-Dimensional Data in 2D

Used in NLP and image recognition for clustering similar objects.

Dimensionality Reduction

1. **Compute Pairwise Similarities in High-Dimensional Space:**
    - Convert Euclidean distances into probability distributions using a Gaussian distribution.
    - Given two points x_i and x_j, the probability of similarity is:
      $p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma^2)}$
2. **Compute Pairwise Similarities in Low-Dimensional Space:**
    - Uses a Student-t distribution to measure similarities in the reduced space:
      $q_{j|i} = \frac{(1+||y_i - y_j||^2)^{-1}}{\sum_{k \neq i}(1+||y_i - y_k||^2)^{-1}}$
3. **Minimize KL-Divergence:**
    - The cost function optimized in t-SNE is the Kullback-Leibler divergence:
      $KL(P||Q) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$
4. **Gradient Descent Optimization:**
    - Adjusts the low-dimensional points iteratively to minimize KL divergence.

Dimensionality Reduction

- **Question:** Given the dataset:

Sample	Feature 1	Feature 2
A	1.0	2.0
B	1.5	1.8
C	2.0	1.6

- **Solution:**

1. Compute pairwise Euclidean distances.
2. Convert distances into probabilities using Gaussian distribution.
3. Compute pairwise similarities in low-dimensional space.
4. Minimize KL divergence using gradient descent.
5. Obtain final 2D embedding approximating original relationships.

Dimensionality Reduction

1. **Compute Euclidean distances:**
   - Distance(A, B) = $\sqrt{(1.5-1)^2+(1.8-2)^2} = \sqrt{(0.5)^2+(-0.2)^2} = \sqrt{0.25+0.04} = \sqrt{0.29} = 0.54$
   - Distance(A, C) = $\sqrt{(2-1)^2+(1.6-2)^2} = \sqrt{(1)^2+(-0.4)^2} = \sqrt{1+0.16} = \sqrt{1.16} = 1.08$
   - Distance(B, C) = $\sqrt{(2-1.5)^2+(1.6-1.8)^2} = \sqrt{(0.5)^2+(-0.2)^2} = \sqrt{0.25+0.04} = \sqrt{0.29} = 0.54$
2. **Convert distances into probabilities using Gaussian distribution:**
   - Compute similarity scores:

$$p_{j|i} = \frac{\exp(-d_{ij}^2/2\sigma^2)}{\sum_{k\neq i}\exp(-d_{ik}^2/2\sigma^2)}$$

   - Assume $\sigma = 1$, then calculate probabilities.
3. **Compute pairwise similarities in low-dimensional space using Student-t distribution**
   - Use t-distribution with one degree of freedom (heavy-tailed distribution).

$$q_{j|i} = \frac{(1+||y_i-y_j||^2)^{-1}}{\sum_{k\neq i}(1+||y_i-y_k||^2)^{-1}}$$

4. **Minimize KL divergence using gradient descent:**
   - KL divergence loss:

$$KL(P||Q) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

   - Adjust low-dimensional embeddings to minimize this loss.
5. **Obtain final 2D embedding approximating original relationships.**
   - The final embedding is adjusted iteratively until convergence.