

Comprehensive Lecture Notes: Naive Bayes and Support Vector Machines

These notes are structured to follow the 40-slide presentation on Naive Bayes (NB) and Support Vector Machines (SVM).

PART 1: NAIVE BAYES CLASSIFIERS

Title Slide

Objective: Introduce Naive Bayes as a fundamental, probabilistic machine learning technique.

What is Naive Bayes?

Key Points:

1. **Definition:** Naive Bayes is a collection of classification algorithms based on Bayes' Theorem.
2. **Family of Algorithms:** It's not one fixed algorithm. The specific type used depends on the data's distribution (e.g., Gaussian for continuous, Multinomial for counts).
3. **Supervised & Generative:** It learns from labeled data (supervised) and models how the data for each class is generated (generative).
4. **Core Principle:** The strong assumption of conditional independence among features.

Bayes' Theorem

Key Points:

1. **Core Concept:** Bayes' Theorem is used to calculate conditional probability (the probability of an event given prior knowledge).
2. **Formula:** $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$
3. **Application:** In classification, we replace A with the **Class (C)** and B with the **Features (X)**. We want to find the probability of a class given the observed features: $P(C|X)$.

Components of the Theorem

Key Points & Variable Descriptions:

1. **Posterior Probability ($P(A|B)$ or $P(C|X)$):** This is the outcome we are trying to predict—the probability that the data instance X belongs to class C .
2. **Likelihood ($P(B|A)$ or $P(X|C)$):** The probability of observing the features X given that the class C is true. This is the hardest part to calculate accurately, and where the "Naive" assumption simplifies things.
3. **Prior Probability ($P(A)$ or $P(C)$):** The inherent, unconditional probability of the class C occurring in the dataset.
4. **Evidence ($P(B)$ or $P(X)$):** The probability of the features X occurring. Since this value is constant across all classes, it is often ignored in classification, and we focus only on maximizing the numerator ($P(X|C) \cdot P(C)$).

Why is it called "Naive"?

Key Points:

1. **The Assumption:** It assumes that the features are conditionally independent of each other, given the class variable.
2. **Example:** If we classify an email as "Spam," the presence of the word "Free" is assumed to be independent of the presence of the word "Money," even though they frequently appear together in spam emails.
3. **Real-World Context:** This assumption is rarely true, yet Naive Bayes often performs surprisingly well, especially in text classification.

The Classifier Equation

Key Points:

1. **Decomposition:** Due to the naive independence assumption, the likelihood $P(X|C)$ can be decomposed into the product of individual feature probabilities: $P(x_1|C) \times P(x_2|C) \times \dots \times P(x_n|C)$.
2. **Maximization:** The final goal is to find the class y that maximizes the product of the Prior Probability and the Likelihood products.
3. **Simplified Formula:** $y = \operatorname{argmax} (P(y) \prod_{i=1}^n P(x_i|y))$

Types of Classifiers

Key Points: Naive Bayes needs to calculate $P(x_i|y)$. The method used depends on the feature type.

1. **Gaussian NB:** For continuous, numerical features.
2. **Multinomial NB:** For discrete features representing counts (e.g., word frequency).
3. **Bernoulli NB:** For binary or boolean features (e.g., presence or absence of a feature).

Gaussian Naive Bayes

Key Points:

1. **Applicability:** Used when features like height, weight, or sensor readings are involved.
2. **Assumption:** These features are assumed to follow a Gaussian (or Normal) distribution.
3. **Calculation:** The model estimates the mean (μ_y) and standard deviation (σ_y) of each feature *for each class*.
4. **PDF:** We then use the Probability Density Function (PDF) of the normal distribution to calculate the likelihood $P(x_i|y)$.

Multinomial Naive Bayes

Key Points:

1. **Applicability:** Best suited for document classification or text data where features are term frequency counts (how many times a word appears).
2. **Mechanism:** It works with count data, estimating the probability of a given word occurring in a document belonging to a specific class.

Bernoulli Naive Bayes

Key Points:

1. **Applicability:** Used when features are binary (0 or 1). In text, this means a word is either present (1) or absent (0) in a document, ignoring the count.
2. **Contrast with Multinomial:** Multinomial cares about *how many times* a feature occurs; Bernoulli only cares about *if* it occurs.

Zero Frequency Problem

Key Points:

1. **The Issue:** If a feature value in the test data was never seen in the training data for a specific class, the likelihood $P(x_i|y)$ becomes 0.
2. **The Disaster:** Since all likelihoods are multiplied together (due to the independence assumption), multiplying by zero causes the entire posterior probability for that class to become 0, regardless of the other feature probabilities. This makes the model unable to classify accurately.

Solution: Laplace Smoothing

Key Points:

1. **Mechanism:** Laplace (or Additive) Smoothing addresses the zero frequency problem by adding a small positive number (α , typically 1) to every count.
2. **Effect:** This ensures that every feature/class combination has a non-zero count, thus avoiding multiplication by zero. It's a method of regularization.

Use Case: Spam Filtering

Key Points:

1. **Process:** This is a classic example using Multinomial NB.
2. **Training Phase:** Calculate the prior probability of an email being Spam, and the likelihood of every word appearing given the email is Spam vs. Ham.
3. **Prediction Phase:** For a new email, multiply the prior by the likelihood product of all words in the new email for both classes (Spam and Ham).
4. **Decision:** The class with the higher score wins.

Advantages

Key Points:

1. **Speed:** NB is extremely fast because it only involves counting frequencies and simple multiplication; no iterative optimization is needed.

2. **Data Efficiency:** It often requires less training data to achieve good results compared to more complex models.
3. **Scalability:** Handles multi-class problems naturally without needing complex "one-vs-rest" strategies.

Disadvantages

Key Points:

1. **Flawed Assumption:** The independence assumption is almost always violated in reality, limiting the model's accuracy on highly correlated data.
2. **Poor Probability Estimator:** While the model often correctly predicts the *class*, the actual probability values it outputs are often exaggerated (overconfident) and should not be trusted as precise probabilities.

Python Code Example

Focus: Demonstrating **GaussianNB** for a continuous dataset (Iris).

Variable/Code Snippet	Description
<code>load_iris()</code>	Function to load a sample dataset with continuous features.
<code>GaussianNB()</code>	Class used for Naive Bayes when features are continuous and assumed normally distributed.
<code>X, y</code>	<code>X</code> is the feature matrix (measurements like sepal length); <code>y</code> is the target vector (species: 0, 1, or 2).
<code>train_test_split()</code>	Utility to divide the dataset into training data (80%) and testing data (20%).
<code>gnb.fit(X_train, y_train)</code>	Training Step. The model calculates the mean and variance for each feature within each class.
<code>gnb.predict(X_test)</code>	Prediction Step. Uses the learned means/variances to calculate the posterior probability for the test data and returns the class with the highest score.
<code>accuracy_score()</code>	Measures the performance of the model by comparing the true labels (<code>y_test</code>) against the predicted labels (<code>y_pred</code>).

PART 2: SUPPORT VECTOR MACHINES

SVM Title

Objective: Introduce Support Vector Machines (SVM) as a highly effective and robust discriminative classifier.

What is SVM?

Key Points:

1. **Discriminative Model:** SVM focuses on finding the boundary that best separates the classes, unlike generative models (like NB) which model the data distribution within each class.
2. **Objective:** To find the optimal **hyperplane** that maximizes the separation distance between the different classes.
3. **Hyperplane:** A boundary used to classify data points. In 2D, it's a line; in 3D, a plane; in higher dimensions, it's a hyperplane.

Geometric Intuition

Key Points:

1. **The Goal:** Not just separating classes, but finding the boundary that is *farthest* from the nearest training data points of any class.
2. **Maximum Margin:** This distance is called the margin, and maximizing it is the core optimization task of SVM. A larger margin leads to better generalization.

Key Terminology

Key Points & Variable Descriptions:

1. **Hyperplane:** The decision boundary itself.
2. **Support Vectors:** The training data points that lie closest to the hyperplane. They are crucial because they directly define the position and orientation of the optimal hyperplane. All other points are irrelevant once the hyperplane is defined.
3. **Margin:** The distance between the hyperplane and the closest support vector from either class.

Maximizing the Margin

Key Points:

1. **Optimization:** SVM is a constrained optimization problem.
2. **Vector w :** The vector perpendicular to the hyperplane. Minimizing the norm ($\|w\|$) is equivalent to maximizing the margin ($2/\|w\|$).

Hard vs Soft Margin

Key Points:

1. **Hard Margin:** Applies to linearly separable data. It demands a perfect separation where all data points must be outside the margin and on the correct side of the hyperplane. It is brittle and sensitive to outliers.
2. **Soft Margin:** Used for real-world, noisy, or non-perfectly separable data. It allows some training points to be misclassified or fall within the margin, prioritizing a wider, more robust margin over absolute perfection.

Cost Function (Hinge Loss)

Key Points:

1. **Hinge Loss:** The cost function used in SVM optimization.
2. **Loss Calculation:** Loss is zero if the predicted class ($f(x)$) and true class (y) agree with sufficient confidence (i.e., the point is correctly classified and outside the margin). If the point is inside the margin or misclassified, the loss increases linearly.

Regularization Parameter C

Key Points:

1. **Purpose:** The hyperparameter C controls the trade-off between maximizing the margin size and minimizing misclassification error (slack variables).
2. **Large C :** High penalty for misclassification. Leads to a smaller, stricter margin (risk of overfitting).
3. **Small C :** Low penalty for misclassification. Leads to a larger, softer margin (risk of underfitting).

The Non-Linear Problem

Key Points:

1. **The Challenge:** Data often cannot be separated by a straight line or flat plane (non-linear data). Example: concentric circles.

The Kernel Trick

Key Points:

1. **Solution:** To handle non-linear data, SVM maps the data from the current dimension to a **higher-dimensional feature space** where separation becomes linear.
2. **The Trick:** Instead of calculating the computationally expensive transformation and projection of every data point, the **kernel function** calculates the dot product between the points in the *higher dimension* while operating only in the original dimension. This is the "trick."

Types of Kernels

Key Points:

1. **Linear Kernel:** Simple dot product; useful for highly dimensional, sparse data (like text).
2. **Polynomial Kernel:** Allows for curved decision boundaries.
3. **RBF (Radial Basis Function) Kernel:** Most common and powerful. Maps the data into potentially infinite dimensions, allowing for complex, non-linear decision boundaries.

RBF Kernel and Gamma (γ)

Key Points:

1. **RBF Function:** $K(x, x') = \exp(-\gamma\|x - x'\|^2)$. It measures similarity based on proximity.
2. **Hyperparameter γ :** Controls the influence of a single training example (support vector).
 - **High γ (Small Radius):** Only points extremely close to the support vector influence the decision boundary. Boundary becomes very complex and jagged (overfitting).
 - **Low γ (Large Radius):** Points further away still influence the boundary. Boundary becomes smoother (underfitting).

Support Vector Regression (SVR)

Key Points:

1. **Extension to Regression:** SVM principles can be adapted for regression tasks.
2. **Mechanism:** Instead of finding a hyperplane to separate classes, SVR finds a band or "tube" (defined by a threshold ϵ) around the predicted value.
3. **Tolerance ϵ :** Data points within this ϵ -tube are considered correctly predicted and incur no penalty. Only points outside the tube contribute to the loss function.

Advantages of SVM

Key Points:

1. **High Dimensionality:** Highly effective in spaces with more features than samples.
2. **Memory Efficient:** Only the small set of Support Vectors (the critical points) are stored in memory after training.
3. **Robust:** Works well on structured, well-defined datasets.

Disadvantages of SVM

Key Points:

1. **Scalability:** Training time can be very long for large datasets (millions of records) due to the computational complexity of the optimization problem.
2. **Parameter Tuning:** The performance is highly dependent on selecting the correct C and γ values (requires cross-validation).
3. **Interpretability:** Since it involves complex geometry and high-dimensional mapping, it is difficult to interpret the model's decision process.

Python Code Example

Focus: Demonstrating **SVC** (Support Vector Classification) with the RBF kernel.

Variable/Code Snippet	Description
<code>SVC</code>	The core class for Support Vector Classification in scikit-learn.
<code>datasets.load_iris()</code>	Loads the Iris dataset.
<code>SVC(kernel='rbf', C=1.0, gamma='scale')</code>	Initialization. Sets up the model to use the RBF kernel (non-linear) with standard regularization $C = 1.0$.
<code>model.fit(X_train, y_train)</code>	Training Step. The model identifies the optimal hyperplane and determines which points are the Support Vectors based on the RBF kernel and C value.
<code>model.predict(X_test)</code>	Prediction Step. Uses the learned hyperplane to classify the test samples.
<code>accuracy_score()</code>	Utility to calculate how many of the test predictions matched the true labels.
<code>iris.target_names</code>	An array used to map the numerical predictions (0, 1, 2) back to human-readable species names ('setosa', 'versicolor', 'virginica').