Lecture 3: Expectation Maximization and Dimensionality Reduction Part 1

Course: 2110573 Pattern Recognition

2026-01-21
version 0.1

1 Introduction: Parametric Density Estimation

In pattern classification, a key goal is to model the probability distribution of our data. A common task is to estimate the class-conditional density, $p(\mathbf{x}|\omega_i)$, for each class ω_i.

One approach is to assume the data follows a specific parametric distribution, such as a Gaussian distribution. The task then becomes finding the parameters θ of that distribution that best fit the data. For a Gaussian, the parameters are the mean and variance, $\theta = \{\mu, \sigma^2\}$.

However, real-world data is often complex and cannot be adequately described by a single Gaussian. For instance, the data might be multi-modal, having multiple peaks in its distribution. This motivates the use of more expressive models, like Gaussian Mixture Models.

2 Gaussian Mixture Models (GMMs)

A Gaussian Mixture Model (GMM) represents a probability distribution as a weighted sum of multiple Gaussian components. The probability density of a data point $\mathbf{x}$ is given by:

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} \phi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{1}$$

where:

- K is the number of Gaussian components in the mixture.
- ϕ_k are the **mixing weights**, which represent the prior probability that a data point was generated by the k-th component. These weights must sum to one: $\sum_{k=1}^{K} \phi_k = 1$.
- $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the k-th Gaussian component with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$.
- $\theta = \{\phi_1, ..., \phi_K, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_K\}$ is the complete set of parameters for the model.

We can think of this process generatively. To create a data point $\mathbf{x}$, we first choose a component k with probability ϕ_k. This chosen component is a **latent variable**—it is a hidden part of the model that we do not observe directly. Once component k is chosen, we then draw the data point $\mathbf{x}$ from its corresponding Gaussian distribution, $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

3 Maximum Likelihood Estimation (MLE) for GMMs

Given a dataset $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, our goal is to find the parameters θ that maximize the likelihood of observing this data. The log-likelihood function is:

$$\ell(\theta) = \log p(X|\theta) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \phi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \tag{2}$$

Maximizing this function directly is difficult because the summation over components k is inside the logarithm. Taking the derivative and setting it to zero does not lead to a closed-form solution.

3.1 A Simpler Case: Known Latent Variables

To build intuition, let's consider a simplified scenario where the latent variables are known. For each data point $\mathbf{x}_n$, we know which component k generated it. Let's define an indicator variable z_{nk} which is 1 if $\mathbf{x}_n$ came from component k, and 0 otherwise.

The **complete-data log-likelihood** is:

$$\begin{aligned}\ell_c(\theta) = \log p(X, Z|\theta) &= \sum_{n=1}^{N} \log p(\mathbf{x}_n, \mathbf{z}_n|\theta) \\ &= \sum_{n=1}^{N} \log \left(\prod_{k=1}^{K} [\phi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{nk}} \right) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left(\log \phi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)\end{aligned}$$

This expression is much easier to optimize because the log now acts directly on the Gaussian and mixing weight terms. We can find the MLE for each parameter by taking derivatives and setting them to zero.

Derivation for Mixing Weights ϕ_k: We must maximize with the constraint that $\sum_k \phi_k = 1$. We use a Lagrange multiplier λ:

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \log \phi_k + \lambda \left(\sum_{k=1}^{K} \phi_k - 1 \right)$$

Taking the derivative with respect to ϕ_j:

$$\frac{\partial \mathcal{L}}{\partial \phi_j} = \sum_{n=1}^{N} \frac{z_{nj}}{\phi_j} + \lambda = 0 \implies \sum_{n=1}^{N} z_{nj} = -\lambda \phi_j$$

Let $N_j = \sum_n z_{nj}$ be the number of points in cluster j. Then $N_j = -\lambda \phi_j$. Summing over all j:

$$\sum_{j=1}^{K} N_j = N = -\lambda \sum_{j=1}^{K} \phi_j = -\lambda \implies \lambda = -N$$

Substituting back, we get $N_j = -(-N)\phi_j$, which gives the MLE:

$$\hat{\phi}_j = \frac{N_j}{N} = \frac{\sum_{n=1}^{N} z_{nj}}{N} \tag{3}$$

Derivation for Means $\boldsymbol{\mu}_k$: We isolate the terms depending on $\boldsymbol{\mu}_j$:

$$\frac{\partial \ell_c}{\partial \boldsymbol{\mu}_j} = \frac{\partial}{\partial \boldsymbol{\mu}_j} \sum_{n=1}^{N} z_{nj} \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = 0$$

Using $\log \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$:

$$\sum_{n=1}^{N} z_{nj} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_j) = 0 \implies \sum_{n=1}^{N} z_{nj} \mathbf{x}_n = \left(\sum_{n=1}^{N} z_{nj} \right) \boldsymbol{\mu}_j$$

This gives the MLE for the mean:

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{n=1}^{N} z_{nj} \mathbf{x}_n}{\sum_{n=1}^{N} z_{nj}} \tag{4}$$

Derivation for Covariances $\boldsymbol{\Sigma}_k$: Similarly, taking the derivative with respect to $\boldsymbol{\Sigma}_j^{-1}$ (and using properties of matrix derivatives) and setting it to zero yields:

$$\hat{\boldsymbol{\Sigma}}_j = \frac{\sum_{n=1}^{N} z_{nj} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j)^T}{\sum_{n=1}^{N} z_{nj}} \tag{5}$$

These results are intuitive: the MLE for each component's parameters are calculated just as they would be for a single Gaussian, but using only the data points assigned to that component.

4 The Expectation-Maximization (EM) Algorithm

In reality, we don't know the latent variables Z. The EM algorithm is an iterative procedure to find MLE solutions in such cases. It works by creating a lower bound on the log-likelihood and iteratively maximizing this bound.

4.1 The Evidence Lower Bound (ELBO)

Let $q(\mathbf{z})$ be an arbitrary probability distribution over the latent variables. We can write the log-likelihood as:

$$\begin{aligned}\ell(\theta) = \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}|\theta) &= \log \sum_{\mathbf{z}} q(\mathbf{z}) \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z})} \\ &\geq \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z})} \quad \text{(Jensen's Inequality)}\end{aligned}$$

This lower bound is often called the **Evidence Lower Bound (ELBO)**:

$$\mathcal{L}(q, \theta) = \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\theta) - \log q(\mathbf{z})] \leq \ell(\theta) \tag{6}$$

4.2 The Two Steps of EM

EM is a two-stage iterative algorithm. Starting with an initial guess for the parameters θ^{old}, we repeat:

1. E-Step (Expectation): We fix the parameters θ^{old} and maximize the ELBO with respect to the distribution $q(\mathbf{z})$. The gap between the log-likelihood and the ELBO is the KL-divergence $KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x}, \theta^{\text{old}}))$. The gap is zero, and the bound is tight, when we set $q(\mathbf{z})$ to be the posterior distribution of the latent variables given the data and current parameters:

$$q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \theta^{\text{old}}) \tag{7}$$

This step involves computing this posterior distribution.

2. M-Step (Maximization): We fix the distribution $q(\mathbf{z})$ found in the E-step and maximize the ELBO with respect to the parameters θ to find θ^{new}:

$$\begin{aligned}\theta^{\text{new}} &= \arg\max_{\theta} \mathcal{L}(q, \theta) \\ &= \arg\max_{\theta} \mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}, \mathbf{z}|\theta)]\end{aligned}$$

This is equivalent to maximizing the **expected complete-data log-likelihood**, where the expectation is taken with respect to the posterior distribution calculated in the E-step.

5 EM Applied to Gaussian Mixture Models

5.1 E-Step: Calculating Responsibilities

In the E-step, we compute the posterior probability that each data point $\mathbf{x}_n$ belongs to component k, given the current parameters θ^{old}. This posterior is often called the **responsibility** of component k for data point $\mathbf{x}_n$.

$$\begin{aligned}\gamma_{nk} = p(z_{nk} = 1|\mathbf{x}_n, \theta^{\text{old}}) &= \frac{p(z_{nk} = 1, \mathbf{x}_n|\theta^{\text{old}})}{p(\mathbf{x}_n|\theta^{\text{old}})} \\ &= \frac{p(z_{nk} = 1|\theta^{\text{old}}) p(\mathbf{x}_n|z_{nk} = 1, \theta^{\text{old}})}{\sum_{j=1}^{K} p(z_{nj} = 1|\theta^{\text{old}}) p(\mathbf{x}_n|z_{nj} = 1, \theta^{\text{old}})} \\ &= \frac{\phi_k^{\text{old}} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k^{\text{old}}, \boldsymbol{\Sigma}_k^{\text{old}})}{\sum_{j=1}^{K} \phi_j^{\text{old}} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j^{\text{old}}, \boldsymbol{\Sigma}_j^{\text{old}})}\end{aligned}$$

5.2 M-Step: Updating Parameters

In the M-step, we maximize the expected complete-data log-likelihood:

$$Q(\theta, \theta^{\text{old}}) = \mathbb{E}_{Z|X,\theta^{\text{old}}}[\ell_c(\theta)] = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk} \left(\log \phi_k + \log \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right)$$

This has the exact same form as the complete-data log-likelihood with known labels, but the hard indicator assignments z_{nk} are replaced by their "soft" expected values, the responsibilities γ_{nk}. Therefore, the maximization yields the same results, with z_{nk} replaced by γ_{nk}.

Let $N_k = \sum_{n=1}^{N} \gamma_{nk}$ be the effective number of points assigned to cluster k. The update rules are:

$$\phi_k^{\text{new}} = \frac{N_k}{N} = \frac{\sum_{n=1}^{N} \gamma_{nk}}{N} \tag{8}$$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n \tag{9}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \tag{10}$$

5.3 Summary of the EM Algorithm for GMM

1. **Initialize** the parameters $\phi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ for all $k = 1, ..., K$.
2. **Repeat until convergence:**
   (a) **E-Step:** For each data point $\mathbf{x}_n$, compute the responsibilities for each component k:

$$\gamma_{nk} = \frac{\phi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \phi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

   (b) **M-Step:** Update the parameters using the computed responsibilities:

$$N_k = \sum_{n=1}^{N} \gamma_{nk}$$

$$\phi_k^{\text{new}} = \frac{N_k}{N}$$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T$$

The algorithm has converged when the log-likelihood value or the parameter values change by less than a small threshold between iterations.

5.4 Relationship to K-Means Clustering

K-Means can be seen as a special, simplified case of GMM. The main differences are summarized below:

One can derive K-Means from the EM for GMM framework by assuming that all components have the same fixed diagonal covariance matrix, $\boldsymbol{\Sigma}_k = \sigma^2 \mathbf{I}$, and then letting $\sigma^2 \to 0$. In this limit, the soft assignments become hard, and the algorithm becomes equivalent to K-Means.

Feature	K-Means	Gaussian Mixture Model (EM)
Assignments	Hard assignments (each point belongs to one cluster)	Soft (probabilistic) assignments (each point has a probability of belonging to each cluster)
Cluster Shape	Assumes spherical clusters (isotropic covariance)	Can model elliptical clusters (full covariance)
Algorithm	Assign points to closest centroid, Recalculate centroids	E-Step: Compute responsibilities (soft assignments) M-Step: Update parameters $(\phi_k, \mu_k, \Sigma_k)$

Table 1: Comparison of K-Means and GMM

5.5 Model Selection: Choosing the Number of Components K

A crucial question when using GMMs is how to choose the optimal number of components, K. If K is too small, the model will underfit the data. If K is too large, it will overfit.

The log-likelihood will always increase with more components, so it is not a good metric for choosing K. Instead, we can use a criterion that penalizes model complexity. A popular choice is the **Bayesian Information Criterion (BIC)**:

$$BIC = -2\ell(\hat{\theta}) + M\log(N) \tag{11}$$

where:

- $\ell(\hat{\theta})$ is the maximized log-likelihood of the model.
- N is the number of data points.
- M is the number of independent parameters in the model. For a GMM with D-dimensional data and full covariance matrices, $M = (K-1) + K(D) + K(D(D+1)/2)$.

We fit the GMM for a range of K values and choose the K that minimizes the BIC score. However, just like in the Elbow Method for K-means, if you have better metrics such as downstream performance, use that instead of BIC.

6 EM on a Simple Example

We consider a simple probabilistic model for student grades in a class. There are three possible grades: A, B, and C. The probabilities for each grade depend on a single parameter θ:

- $P(\text{Grade} = A) = 0.5$
- $P(\text{Grade} = B) = 0.5 - \theta$
- $P(\text{Grade} = C) = \theta$

We assume that $0 < \theta < 0.5$. Let N_a, N_b, N_c be the number of students who received grades A, B, and C respectively. The total number of students is $N = N_a + N_b + N_c$.

6.1 Maximum Likelihood Estimation (Complete Data)

First, let's consider the case where we observe the counts for all three grades: N_a, N_b, and N_c. Our goal is to find the maximum likelihood estimate (MLE) for the parameter θ.

The likelihood of observing these counts is given by the multinomial probability distribution:

$$P(N_a, N_b, N_c|\theta) = \frac{N!}{N_a!N_b!N_c!}(0.5)^{N_a}(0.5-\theta)^{N_b}(\theta)^{N_c} \tag{12}$$

To find the MLE, it is easier to work with the log-likelihood function, $\mathcal{L}(\theta) = \log P(N_a, N_b, N_c|\theta)$.

$$\mathcal{L}(\theta) = \log\left(\frac{N!}{N_a!N_b!N_c!}\right) + N_a\log(0.5) + N_b\log(0.5-\theta) + N_c\log(\theta) \tag{13}$$

We maximize this function by taking the derivative with respect to θ and setting it to zero.

$$\frac{d\mathcal{L}}{d\theta} = \frac{-N_b}{0.5-\theta} + \frac{N_c}{\theta} \tag{14}$$

Setting the derivative to zero:

$$\begin{aligned} \frac{N_c}{\theta} &= \frac{N_b}{0.5-\theta} \\ N_c(0.5-\theta) &= N_b\theta \\ 0.5N_c - N_c\theta &= N_b\theta \\ 0.5N_c &= (N_b+N_c)\theta \end{aligned}$$

This gives us the maximum likelihood estimate for θ:

$$\hat{\theta}_{MLE} = \frac{0.5N_c}{N_b+N_c} \tag{15}$$

6.2 Estimation with Incomplete Data using EM

Now, let's consider a more complex scenario where we do not observe all the data. Suppose we only know the number of students with grade C, N_c, and the total number of students, N. The counts N_a and N_b are unknown (latent variables). Our goal is to estimate θ using the Expectation-Maximization (EM) algorithm.

The EM algorithm is an iterative method which alternates between two steps:

1. **E-step (Expectation):** Compute the expected value of the complete-data log-likelihood, with respect to the conditional distribution of the latent variables given the observed data and the current estimate of the parameters.
2. **M-step (Maximization):** Maximize the expected log-likelihood found in the E-step to update the parameter estimates.

Let's denote the parameter estimate at iteration t as $\theta^{(t)}$.

6.2.1 E-step

In the E-step, we need to compute the expectation of the latent variables, N_a and N_b, given the observed data N_c and the current parameter $\theta^{(t)}$. Specifically, we need to find the conditional distribution of the latent variables.

The number of students who did not get grade C is $N - N_c$. These students must have received either grade A or B. The conditional probability of a student getting grade B, given that they did not get grade C, is:

$$P(\text{Grade} = B|\text{Grade} \neq C) = \frac{P(\text{Grade} = B \text{ and Grade} \neq C)}{P(\text{Grade} \neq C)} = \frac{P(B)}{P(A)+P(B)} \tag{16}$$

Using the given probabilities:

$$P(\text{Grade} \neq C) = 0.5 + (0.5-\theta) = 1-\theta \tag{17}$$

So, the conditional probability is:

$$p_B = P(\text{Grade} = B|\text{Grade} \neq C) = \frac{0.5-\theta}{1-\theta} \tag{18}$$

Similarly for grade A:

$$p_A = P(\text{Grade} = A|\text{Grade} \neq C) = \frac{0.5}{1-\theta} \tag{19}$$

Given that there are $N - N_c$ students who are not C's, the number of students with grade B, N_b, follows a binomial distribution:

$$P(N_b|N_c, N, \theta^{(t)}) \sim \text{Binomial}\left(n = N - N_c, p = \frac{0.5-\theta^{(t)}}{1-\theta^{(t)}}\right) \tag{20}$$

The E-step requires us to compute the expected value of the latent variables. The expectation of a binomial distribution is $n \times p$.

$$\mathbb{E}[N_b|N_c, N, \theta^{(t)}] = (N - N_c)\frac{0.5 - \theta^{(t)}}{1 - \theta^{(t)}} \tag{21}$$

Let's call this expected value $\hat{N}_b^{(t+1)}$. The full E-step computes the expectation of the complete-data log-likelihood, but for this problem, we only need the expectation of the sufficient statistics of the latent variables, which are N_b (and N_a).

6.2.2 M-step

In the M-step, we maximize the expected complete-data log-likelihood with respect to θ to get the new estimate $\theta^{(t+1)}$. The complete-data log-likelihood is:

$$\mathcal{L}_c(\theta) = \text{const} + N_b \log(0.5 - \theta) + N_c \log(\theta) \tag{22}$$

We take the expectation of $\mathcal{L}_c(\theta)$ with respect to the conditional distribution of latent variables found in the E-step (using $\theta^{(t)}$). This gives the Q-function:

$$Q(\theta, \theta^{(t)}) = \mathbb{E}_{N_b|N_c,\theta^{(t)}}[\mathcal{L}_c(\theta)] = \text{const} + \mathbb{E}[N_b] \log(0.5 - \theta) + N_c \log(\theta) \tag{23}$$

Here, $\mathbb{E}[N_b]$ is the value $\hat{N}_b^{(t+1)}$ we computed in the E-step.

$$Q(\theta, \theta^{(t)}) = \text{const} + \hat{N}_b^{(t+1)} \log(0.5 - \theta) + N_c \log(\theta) \tag{24}$$

To maximize $Q(\theta, \theta^{(t)})$, we take the derivative with respect to θ and set it to zero.

$$\frac{\partial Q}{\partial \theta} = \frac{-\hat{N}_b^{(t+1)}}{0.5 - \theta} + \frac{N_c}{\theta} = 0 \tag{25}$$

Solving for θ gives us the updated parameter $\theta^{(t+1)}$:

$$\begin{aligned}
\frac{N_c}{\theta} &= \frac{\hat{N}_b^{(t+1)}}{0.5 - \theta} \\
0.5N_c &= (\hat{N}_b^{(t+1)} + N_c)\theta \\
\theta^{(t+1)} &= \frac{0.5N_c}{\hat{N}_b^{(t+1)} + N_c}
\end{aligned}$$

6.2.3 The EM Algorithm Summary

The algorithm proceeds as follows:

1. **Initialize:** Choose an initial value for θ, say $\theta^{(0)}$.
2. **Iterate** for $t = 0, 1, 2, \ldots$ until convergence:
   (a) **E-step:** Compute the expected number of students with grade B:

$$\hat{N}_b^{(t+1)} = \mathbb{E}[N_b|N_c, \theta^{(t)}] = (N - N_c)\frac{0.5 - \theta^{(t)}}{1 - \theta^{(t)}}$$

   (b) **M-step:** Update the estimate for θ:

$$\theta^{(t+1)} = \frac{0.5N_c}{N_c + \hat{N}_b^{(t+1)}}$$

7 Applications of the EM Algorithm

The EM algorithm is a general-purpose method for finding MLE solutions for models with latent variables. Its applications extend far beyond GMMs and include:

- **Computer Vision:** Image segmentation, background modeling.
- **Natural Language Processing:** Probabilistic Latent Semantic Analysis (pLSA) for topic modeling, Hidden Markov Models (HMMs) for part-of-speech tagging.
- **Bioinformatics:** Gene expression analysis, motif finding in DNA sequences.
- **Finance:** Volatility modeling.
- **Missing Data:** EM can be used to estimate missing values in a dataset.

8 Introduction to Dimensionality Reduction: The Curse of Dimensionality

In many machine learning applications, we encounter datasets with a very large number of features or dimensions. While having more features might seem to provide more information, it can lead to several problems collectively known as the "curse of dimensionality."

- **Difficulty in Visualization:** It is challenging to visualize and understand the structure of data in high-dimensional spaces. Humans can only perceive up to three dimensions.
- **Increased Sparsity:** As the number of dimensions increases, the data points become increasingly sparse. This sparsity makes it difficult to find meaningful patterns or clusters. For a fixed number of data points, the density of the data decreases exponentially with the number of dimensions.
- **Increased Computational Cost:** Searching for neighbors or performing other computations in high-dimensional spaces is computationally expensive, leading to longer runtimes.
- **Overfitting:** With a high number of features, machine learning models are more likely to overfit the training data, meaning they will perform well on the training data but poorly on unseen data.
- **Degradation of Distance Metrics:** In high dimensions, the concept of distance becomes less meaningful. As shown in the provided slides, for many distributions, the distance between any two points tends to become almost the same, making distance-based algorithms like K-Nearest Neighbors (KNN) less effective.

9 Combating the Curse of Dimensionality

There are two main approaches to combat the curse of dimensionality:

1. **Feature Selection:** We select a subset of the original features and discard the rest. The goal is to keep only the "good" or most informative features.
2. **Feature Transformation (Feature Extraction):** We create a new, smaller set of features by combining the original features. The new features are a transformation of the old features.

9.1 Feature Selection vs. Feature Transformation

- **Feature Selection:**
  - Keeps the original features, which is useful when the interpretability of the model is important (i.e., we want to know which features are most important).
  - It can be hard to automatically select the best subset of features.
- **Feature Transformation:**
  - Creates new features that are combinations of the original features.
  - Often more powerful than feature selection in terms of predictive accuracy.
  - The new features are often harder to interpret, making the model a "black box."

10 Feature Selection Methods

10.1 Simple Methods

- **Drop Missing Features:** If a feature has a large number of missing values, it might be better to drop it.
- **Low Variance Column:** A feature that is constant or has very low variance provides little information and can be removed.
- **Forward or Backward Feature Elimination:** These are greedy algorithms.

- **Forward Selection:** Start with no features and add them one by one, at each step adding the feature that gives the best performance.
- **Backward Elimination:** Start with all features and remove them one by one, at each step removing the feature that results in the smallest decrease in performance.

10.2 Advanced Methods

- **L1 Regularization (Lasso):** Some models, like linear regression with L1 regularization, automatically perform feature selection by shrinking the coefficients of less important features to zero.
- **Tree-based Classifiers:** Models like Random Forest and XGBoost can provide feature importance scores, which can be used to select features.
- **Genetic Algorithms:** A powerful, heuristic-based method inspired by natural selection.

10.3 Genetic Algorithms for Feature Selection

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

The process of natural selection starts with the selection of the fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

The steps in a genetic algorithm for feature selection are:

1. **Initialization:** Create an initial population of N "individuals" (classifiers). Each individual represents a subset of features, often encoded as a binary string (1 if the feature is used, 0 otherwise).
2. **Selection:** Evaluate the fitness of each individual by training a classifier with the corresponding subset of features and calculating a score (e.g., accuracy, F1-score). Rank the individuals based on their fitness and select the top-performing ones to "survive" and "reproduce." The lower-performing half is "killed off."
3. **Crossover:** Create new individuals ("offspring") by combining the "genes" (feature subsets) of the surviving parents. This is done by randomly selecting parts of the feature subsets from the parents.
4. **Mutation:** Introduce random changes (mutations) in the offspring's genes. This helps to maintain diversity in the population and avoid getting stuck in a local optimum. A common mutation rate is 1/k, where k is the number of features, so on average, one feature is flipped per individual.
5. **Repeat:** Repeat the selection, crossover, and mutation steps for a number of generations or until a satisfactory level of performance is reached.

11 Feature Transformation Methods

11.1 Linear Algebra Review

Before diving into PCA, let's review some key concepts from linear algebra.

- **Matrix as a sequence of column vectors:** A matrix can be viewed as a transformation of space. The columns of the matrix represent where the basis vectors of the original space land after the transformation.
- **Eigendecomposition (ED) and Singular Value Decomposition (SVD):** These are ways of decomposing a matrix into simpler components. They can be viewed as a sequence of rotations and stretches. For a symmetric matrix A, the eigendecomposition is given by:

$$A = Q\Lambda Q^{-1} = Q\Lambda Q^T$$

where Q is a matrix whose columns are the eigenvectors of A, and Λ is a diagonal matrix containing the corresponding eigenvalues.

- **Projection as a change of basis:** Projecting a vector onto a new basis is a way of representing that vector in terms of the new basis vectors. If the basis vectors are orthogonal, the projection is simplified.

11.2 Covariance Matrix

The covariance matrix is a square matrix that describes the variance and covariance between pairs of random variables. For a set of random variables $X_1, X_2, \ldots, X_n$, the covariance matrix Σ is an $n \times n$ matrix where the entry in the (i, j) position is the covariance between X_i and X_j:

$$\Sigma_{ij} = \text{Cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)]$$

where $\mu_i = E[X_i]$ is the mean of X_i.
The covariance matrix is:

- **Symmetric:** $\text{Cov}(X_i, X_j) = \text{Cov}(X_j, X_i)$, so $\Sigma = \Sigma^T$. Because of this, its eigenvalues are real and its eigenvectors are mutually orthogonal.
- **Positive Semi-definite:** For any vector x, $x^T \Sigma x \geq 0$. This implies that all of its eigenvalues are non-negative.

12 Principal Component Analysis (PCA)

PCA is a feature transformation technique that converts a set of correlated features into a set of linearly uncorrelated features called principal components. The goal of PCA is to find a projection (a transformation) that describes the maximum variation in the data. The idea is that the directions of maximum variance are the most informative.

12.1 Mathematical Formulation

We want to find a direction, represented by a unit vector w, such that the variance of the data projected onto this direction is maximized.

The projection of a data point x onto the direction w is given by $w^T x$. The variance of the projected data is:

$$\text{Var}(w^T x) = E[(w^T x - E[w^T x])^2]$$

Since $E[w^T x] = w^T E[x] = w^T \mu_x$, we have:

$$\begin{aligned}
\text{Var}(w^T x) &= E[(w^T x - w^T \mu_x)^2] \\
&= E[(w^T (x - \mu_x))(w^T (x - \mu_x))^T] \\
&= E[w^T (x - \mu_x)(x - \mu_x)^T w] \\
&= w^T E[(x - \mu_x)(x - \mu_x)^T] w \\
&= w^T \Sigma w
\end{aligned}$$

where Σ is the covariance matrix of the data.

So, the problem is to find the vector w that maximizes the variance $w^T \Sigma w$, subject to the constraint that w is a unit vector, i.e., $w^T w = 1$.

12.2 Lagrangian Multipliers

To solve this constrained optimization problem, we can use the method of Lagrange multipliers.

Intuition behind Lagrange Multipliers:

Imagine you are on a hill, and you want to find the highest point on the hill (i.e., maximize your altitude). This is an unconstrained optimization problem, and the solution is the peak of the hill. Now, suppose you are constrained to walk along a specific path on the hill. The highest point on your path is not necessarily the peak of the hill.

Let the function representing the altitude of the hill be $f(x, y)$, and the function representing the path be $g(x, y) = c$. The method of Lagrange multipliers states that at the maximum (or minimum) point of $f(x, y)$ subject to the constraint $g(x, y) = c$, the gradient of f will be parallel to the gradient of g. In other words, their gradient vectors will point in the same (or opposite) direction.

Mathematically, this can be expressed as:

$$\nabla f(x, y) = \lambda \nabla g(x, y)$$

where λ is a scalar called the Lagrange multiplier.

To solve this, we define a new function, the Lagrangian, as:

$$L(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - c)$$

and we find the critical points of L by setting its gradient to zero:

$$\nabla L(x, y, \lambda) = 0$$

This gives us a system of equations that we can solve to find the optimal values of x, y, and λ.

Applying Lagrange Multipliers to PCA:

In our case, the function to maximize is $f(w) = w^T \Sigma w$, and the constraint is $g(w) = w^T w = 1$.

The Lagrangian is:

$$L(w, \lambda) = w^T \Sigma w - \lambda(w^T w - 1)$$

To find the maximum, we take the derivative of L with respect to w and set it to zero:

$$\frac{\partial L}{\partial w} = 2\Sigma w - 2\lambda w = 0$$

$$\Sigma w = \lambda w$$

This is the standard eigenvector equation! It tells us that the vector w that maximizes the variance is an eigenvector of the covariance matrix Σ.

12.3 Selecting Eigenvectors

The variance of the data projected onto an eigenvector w is:

$$w^T \Sigma w = w^T(\lambda w) = \lambda w^T w = \lambda$$

Since we want to maximize the variance, we should choose the eigenvector corresponding to the largest eigenvalue. This eigenvector is the first principal component. The second principal component is the eigenvector corresponding to the second-largest eigenvalue, and so on.

The principal components are orthogonal to each other because they are eigenvectors of a symmetric matrix. This means they capture different, uncorrelated aspects of the data.

12.4 Decomposition and Reconstruction

Let X be the original data matrix of size $N \times d$, where N is the number of data points and d is the number of dimensions. Assume X has been centered by subtracting the mean of each feature. Let W be the $d \times k$ matrix whose columns are the first k principal components (eigenvectors).

Decomposition (Projection): To project the original data onto the new k-dimensional subspace, we multiply the data matrix X by the matrix of principal components W:

$$Z = XW$$

The resulting matrix Z is of size $N \times k$ and contains the transformed data, where each column is a new feature (principal component).

Reconstruction: To reconstruct an approximation of the original data from the transformed data, we multiply the transformed data matrix Z by the transpose of the principal components matrix, W^T:

$$\hat{X} = ZW^T$$

The reconstructed data matrix $\hat{X}$ has the same dimensions as the original data matrix X, but it is an approximation because we have discarded some information by reducing the dimensionality. The reconstruction error is the difference between the original data and the reconstructed data, $X - \hat{X}$.

12.5 How many eigenvectors to choose?

- **Explained Variance:** The total variance in the data is the sum of the eigenvalues. We can choose enough principal components to explain a certain percentage of the total variance (e.g., 95%).
- **Reconstruction Error:** We can reconstruct the original data from the projected data. We can choose enough principal components to keep the reconstruction error below a certain threshold.
- **Scree Plot:** A plot of the eigenvalues in descending order. We can look for an "elbow" in the plot, which indicates a point of diminishing returns.

12.6 Practical Issues

- **Normalization:** If the features have different scales, it is important to normalize them (e.g., to have zero mean and unit variance) before applying PCA. Otherwise, features with larger scales will dominate the principal components.
- **The Gram Matrix Trick:** For high-dimensional data (e.g., images), the covariance matrix Σ can be very large (d x d, where d is the number of dimensions). If the number of data points N is much smaller than d, we can use a trick involving the Gram matrix. Instead of solving the eigenvalue problem for the $d \times d$ matrix XX^T, we can solve it for the $N \times N$ matrix X^TX. The eigenvalues will be the same, and the eigenvectors can be recovered.

13 Summary

- Feature selection and dimensionality reduction techniques are essential for dealing with high-dimensional data and the "curse of dimensionality."
- PCA is a powerful feature transformation technique that finds a new set of uncorrelated features (principal components) that capture the maximum variance in the data.
- The principal components are the eigenvectors of the covariance matrix of the data.