1 Definitions

1.1 Gaussian Mixture Models (GMM)

1. **GMM**: Probabilistic models used for clustering data. The algorithm assumes that the data is generated by a mixture of K Gaussian distributions, where K is a hyperparameter. GMM works by iteratively estimating the parameters of the Gaussian distributions and the weights of the mixture components using the Expectation-Maximization (EM) algorithm.

2. **EM**: An iterative method used for estimating the parameters of statistical models.

   Let Z be a categorical random variable with components $z_1, z_2, \ldots, z_k$, where each component is 0 or 1 i.e. $P(z_j = 1)$ is the probability that a point comes from the Gaussian distribution j.

   Let $\boldsymbol{\theta} = \mu_1, \mu_2, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k, \pi_1, \ldots, \pi_k$, where $\pi_j = P(z_j = 1)$.

   The likelihood is $\prod_{i=1}^{N} P(x_i \mid \boldsymbol{\theta})$.
   Hence, the log-likelihood is $\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log P(x_i \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)$.

   E-Step: Calculate $P(z_j = 1 \mid x_i, \boldsymbol{\theta})\ \forall i, j$.

$$
\begin{aligned}
&P(z_j = 1 \mid x_i, \boldsymbol{\theta}) \\
&= \frac{p(x_i \mid z_j = 1, \mu_j, \Sigma_j) p(z_j = 1 \mid \pi_j)}{p(x_i \mid \boldsymbol{\theta})} \\
&= \frac{\mathcal{N}(x_i \mid \mu_j, \Sigma_j)\pi_j}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_l, \Sigma_k)}
\end{aligned}
$$

   M-Step: Apply MLE and update the parameters $\pi_j, \mu_j, \Sigma_j\ \forall j$.
   Let's find the MLE for μ_j.

$$
\begin{aligned}
&\frac{\partial l}{\partial \mu_j} \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \\
&= \sum_{i=1}^{N} \frac{1}{\sum_{l=1}^{k} \pi_l \mathcal{N}(x_i \mid \mu_l, \Sigma_l)} \frac{\partial l}{\partial \mu_j} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \\
&= \sum_{i=1}^{N} \frac{1}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)} \frac{\partial l}{\partial \mu_j} \pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j) \\
&= \sum_{i=1}^{N} \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_l \mathcal{N}(x_i \mid \mu_k, \Sigma_k)} \frac{\partial l}{\partial \mu_j} \frac{(x_i - \mu_j)^2}{2\Sigma_j} \\
&= \sum_{i=1}^{N} P(z_j = 1 \mid x_i, \theta) \Sigma_j^{-1} (x_i - \mu_j)
\end{aligned}
$$

   We can set this to 0, and solve for μ_j to get $\mu_j = \frac{\sum_{i=1}^{N} P(z_j=1|x_i,\theta) x_i}{\sum_{i=1}^{N} P(z_j=1|x_i,\theta)}$.

   We can do similar calculations for the other two parameters π_j and Σ_j.
   $\pi_j = \frac{\sum_{i=1}^{N} P(z_j=1|x_i,\theta)}{N}$
   $\Sigma_j = \frac{\sum_{i=1}^{N} P(z_j=1|x_i,\theta)(x_i-\mu_j)(x_i-\mu_j)^\top}{\sum_{i=1}^{N} P(z_j=1|x_i,\theta)}$

1.2 Kernel Regression

1. Kernel regression is a non-parametric technique (we don't make any assumptions about the data) to estimate the conditional expectation of a random variable, which helps us find a non-linear relationship between a pair of random variables X and Y. In other words, it helps us to formulate a prediction function $\hat{y} = h(\mathbf{x})$ that works with data that is non-linear in $\mathbf{x}$ and y.

2. The process for kernel regression is as follows:

   (a) **Step 1:** Compute $\boldsymbol{\alpha} = (K+\lambda I)^{-1}\mathbf{y}$ where $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ and $k(\mathbf{x}, \mathbf{z})$ is your kernel function.

   (b) **Step 2:** Given a new point $\mathbf{x}$, predict $\hat{y} = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}^{(i)})$

1.3 Kernels

1. Kernel function

   For a given feature transform function $\phi(\mathbf{x})$, a kernel function is a function that takes in two points, $\mathbf{x}$ and $\mathbf{z}$, and returns the value $\phi(\mathbf{x})^\top \phi(\mathbf{z})$.

   The purpose of a kernel function is to compute $\phi(\mathbf{x})^\top \phi(\mathbf{z})$ without having to explicitly compute the feature transforms $\phi(\mathbf{x})$ and $\phi(\mathbf{z})$ that can be prohibitively expensive in both memory and computation time.

2. Common kernels functions

Name	**Function**	**Feature space description**
Linear	$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$	Same as original input space
Polynomial (v1)	$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^d$	All polynomials of degree d
Polynomial (v2)	$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + 1)^d$	All polynomials up to and including degree d
RBF	$k(\mathbf{x}, \mathbf{z}) = e^{-\gamma \|\mathbf{x}-\mathbf{z}\|_2^2}$	Equivalent to polynomial (v2) with infinite d
Boxcar	$k(\mathbf{x}, \mathbf{z}) = \begin{cases} 1 & \text{if } \|\mathbf{x}-\mathbf{z}\|_2 \leq \frac{width}{2} \\ 0 & \text{otherwise} \end{cases}$	Simple toy kernel

1.4 The Kernel Trick

The so-called kernel trick is to reformulate an optimization problem involving feature-transformed input points $\phi(\mathbf{x})$, such that $\phi(\mathbf{x})$ *never* appears alone in the optimization and *only* appears in a dot product with another feature-transformed point, $\phi(\mathbf{x})^\top \phi(\mathbf{z})$. This dot product can then be replaced by the kernel function associated with ϕ,

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$$

The kernel trick can be used to efficiently complex feature transforms to a wide variety of machine learning techniques, including linear regression, logistic regression, PCA, and SVMs.

1.5 Kernel Matrix

The kernel trick often includes applying the kernel to all pairs of N training points $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$. These kernel values can be represented by the symmetric kernel matrix K where $K_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$.

2 Gaussian Mixture Models

2.1 GMM vs K-means

What is the key difference between mixture modeling and K-means?

K-means is hard assignment (each point belongs to only one cluster) while mixture modeling is soft assignment (calculates probability that a point belongs to a cluster).

In GMM, we aim to maximize our likelihood. That is, $\text{argmax}_\theta \prod_{i=1}^{N} P(x_i \mid \theta)$.

$$\begin{aligned}\underset{\theta}{\text{argmax}} \prod_{i=1}^{N} P(x_i \mid \theta) &= \underset{\theta}{\text{argmax}} \prod_{i=1}^{N} \sum_{k=1}^{K} P(x_i, z_i = k \mid \theta) \\ &= \underset{\theta}{\text{argmax}} \prod_{i=1}^{N} \sum_{k=1}^{K} P(z_i = k) P(x_i \mid z_i = k, \theta)\end{aligned}$$

What happens to this expression if we assume a hard-assignment? Simplify using the assumption.

Hard assignment means that $P(z_i = k) = 1$ if the point belongs to the jth cluster.

Thus we have:
$\text{argmax}_\theta \prod_{i=1}^{N} \sum_{k=1}^{K} P(z_i = k) P(x_i \mid z_i = k, \theta)$

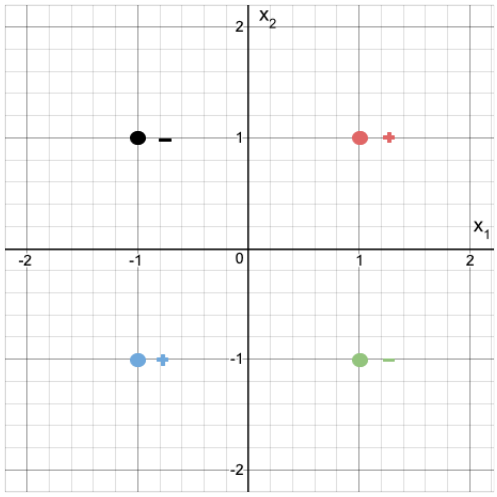
Our points are from a gaussian distribution, thus we have:
$\text{argmax}_\theta \prod_{i=1}^{N} \sum_{k=1}^{K} P(z_i = k) \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-1}{2\sigma^2} ||x_i - \mu_k||_2^2)$
$= \text{argmax}_\theta \prod_{i=1}^{N} \exp(\frac{-1}{2\sigma^2} ||x_i - \mu_k||_2^2)$

Taking the log, we get
$= \text{argmax}_\theta \log \prod_{i=1}^{N} \exp(\frac{-1}{2\sigma^2} ||x_i - \mu_k||_2^2)$
$= \text{argmax}_\theta \sum_{i=1}^{N} \log \exp(\frac{-1}{2\sigma^2} ||x_i - \mu_k||_2^2)$
$= \text{argmax}_\theta \sum_{i=1}^{N} \frac{-1}{2\sigma^2} ||x_i - \mu_k||_2^2$
$= \text{argmin}_\mu \sum_{i=1}^{N} ||x_i - \mu_k||_2^2$

This looks familiar....it's K-means!

3 Kernels

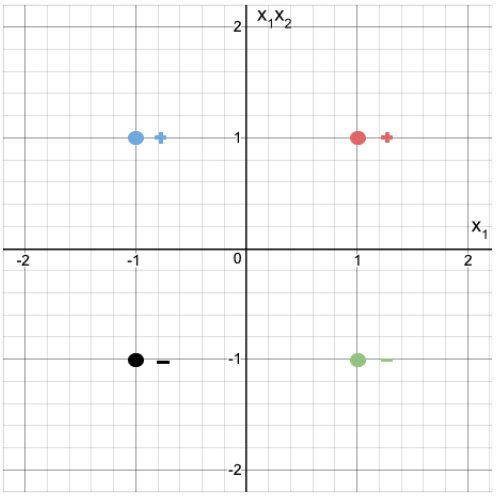
The XOR-problem is a non-linear problem which can be represented by the plot below. For the following questions, consider a feature transformation - $\phi([x_1, x_2]^\top) = [x_1, x_1 x_2]^\top$



1. What is the kernel $k(x, z)$?

$K(x, z) = \phi([x_1, x_2])^\top \phi([z_1, z_2]) = [x_1, x_1 x_2]^\top [z_1, z_1 z_2] = x_1 z_1 + x_1 x_2 z_1 z_2$

2. How is the dataset represented in the transformed space?



3. Is the dataset linearly separable in the transformed space? If so, give the boundary in the original space?

$x_1 x_2 = 0$

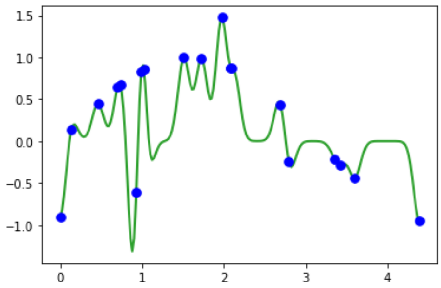
4 Kernel Regression

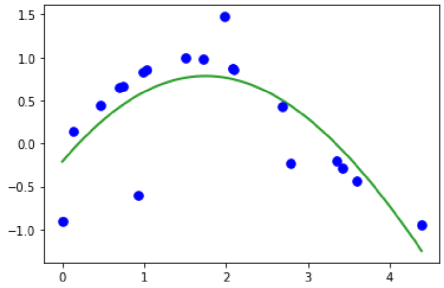
4.1 Conceptual Recap

1. Is kernel regression a parametric or nonparametric model? Explain.

   It is a nonparametric model. The kernel is applied to each data point, so the number of parameters increase as the number of data points increase.

2. Consider the RBF kernel, $k(x, z) = e^{-\gamma\|x-z\|_2^2}$. Match the models below with their corresponding γ values (0.01 or 100). What is the effect of γ on overfitting?





   The left model corresponds to $\gamma = 100$, while the right corresponds to $\gamma = 0.01$. When γ grows too large, the model is likely to overfit.

4.2 Kernelize it!

For this question, consider the box kernel:

$$k(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \|\mathbf{x} - \mathbf{x}'\|_2^2 \leq \frac{1}{2} \\ 0 & otherwise \end{cases}$$

You are given the following 1D data points (x,y): (3,4), (3.7,1), (4.2,-2).

1. Find K and calculate $\boldsymbol{\alpha}$. For simplicity, let $\lambda = 0$. You can use an online inverse matrix calculator.

   K (3x3): $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$, where K^{-1} (3x3): $\begin{bmatrix} 0 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 0 \end{bmatrix}$

   y (3x1): [4, 1, -2]

   With $\gamma = 0$, we get α (3x1): $K^{-1}y = \begin{bmatrix} 0 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 0 \end{bmatrix}$ [4, 1, -2] = [3, 1, -3]

2. Predict $\hat{y}$ for $x = 3.4$.

   $\hat{y} = \sum_{i=1}^{3} \alpha_i k(3.4, x^{(i)})$

   $= \alpha_1 k(3.4, 3) + \alpha_2 k(3.4, 3.7) + \alpha_3 k(3.4, 4.2)$

   = 3*1 + 1*1 + (-3)*0

   = 4