3. (a)

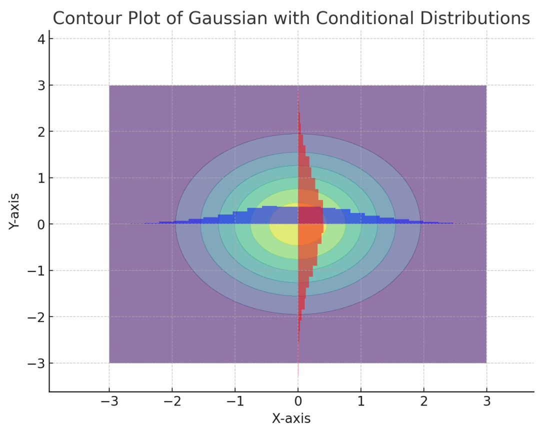
- a(i)
  - L(x) is the loss funciton
  - x is a matrix of parameters representing the linear relation between each content and the rating. The vector of dimension would be 1x6.
  - A = $[p_m, p_n, p_b, p_f, p_c, p_a]$ is the number of hours for each content type, a metrix of dimenstion would be 120x6. Each row correspond to one month of historical data.
  - y is a vector representing each month's score from audience during 10 years, and vector of dimension would be 120x1.
  - The outcome will be the coefficient for the linear relation $\hat{x}$, which is the best fit of this dataset $
- a(ii)
  - The biggest issue of this case is not a linear relation. linear regression cannot solve this problem.
  - Residual analysis. we can examine the difference between the model's predicted and actual values. If the residuals show a non-random pattern, this may indicate that the model is not capturing some key features in the data. Also, if the model is good, it should be very small.
- b(i)
  - 见答案
- b(ii)
  - Beacause the model is complexible, gradient descent is likely to be the most efficient optimisation in rhis case.
  - The set of parameters $\theta = [b, \alpha_m, \beta_m, \mu_m, ..\alpha_a, \beta_a, \mu_a]$
  - the issue can be represented as :

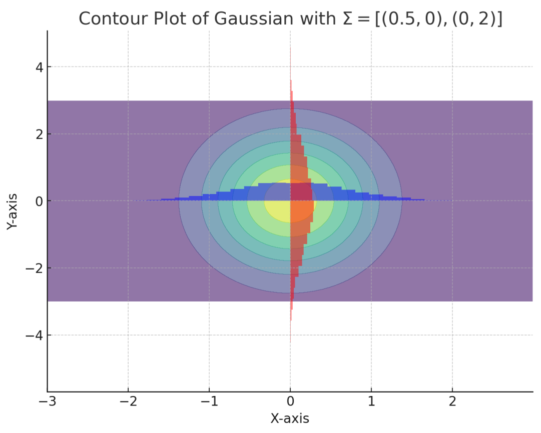
$$\argmin_\theta L(\theta) = \sum_p ||\hat{r}(p) - r(p)||_2^2$$

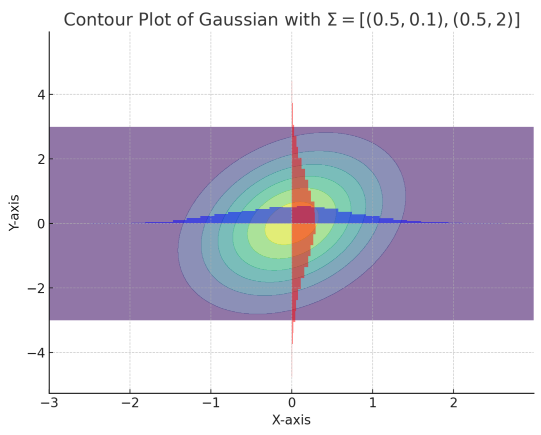
1.(a)

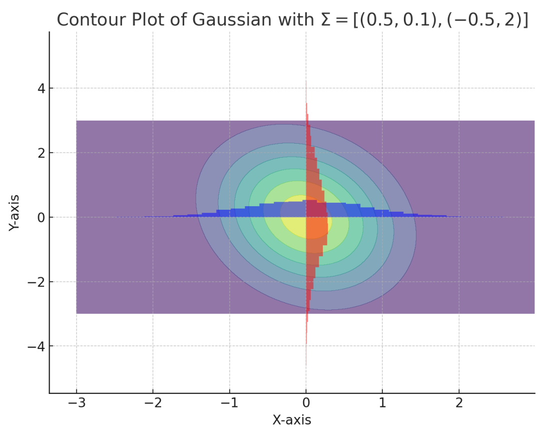
- (i)
  - 在这个问题中，我选择词袋模型BoW，每一个文档被表示为一个向量。向量中的每个元素是该词汇出现的次数。向量的维度取决于词汇表的长度。
  - In this case, I would like to choose the BoW model, and every document will be represented as a vector. Each element in the vector is the number of times the word appears. Moreover, the dimension of each vector depends on the length of vocabulary list.
- (ii)
  - L_0 norm is defined as the number of non-zero element in the vector. Here, there are the number of occurrences of the word.
- (iii)
  - $L_p(\mathbf{d}, \mathbf{d}') = \left(\sum_{i=1}^{n} |d_i - d_i'|^p\right)^{\frac{1}{p}}$
- (iv)
  - When p=2 , this is L_2 norm, which is the square root of the sum of the difference of the vector, and it would measure the similarity between the current vector and given vector.

1.(b)

- (i)
  - $f(x) = (x - \mu)^T \Sigma^{-1} (x - \mu)$ 是多变量高斯（或正态）分布中的二次型部分。这个公式是多变量高斯分布的概率密度函数（PDF）的核心组成部分。μ是均值向量，Σ是协方差矩阵
  - 协方差矩阵, 非对角线元素为0，则他们之间互相独立。
  - 步骤:
    - 定义高斯分布参数μ, Σ
    - 计算高斯分布的值`rv = np.random.multivariate_normal(mu, sigma)`
    - 绘制等高线图`contourf(x, y, np.exp(-0.5 * (x**2 + y**2))`
    - 使用 plt.hist 分别绘制沿X轴和Y轴的条件分布直方图。
    - `plt.hist(rv[:, 0], bins=30, density=True, alpha=0.5,color='blue')`
  - 



- 



- (ii)
  - $\Sigma = [(1,0),(0,1)]$这是一个标准正态分布，其中两个变量相互独立且方差相同。在这种情况下，两个主成分的方差是相同的。因此，使用PCA简化为单维分布会丢失一半的信息。
  - This is a standard normal distribution in which two variables are independent of each other and have the same variance. In this case, the variances of the two principal components are the same. Therefore, using PCA to simplify to a unidimensional distribution loses half of the information.
  - $\Sigma = [(0.5,0),(0,2)]$ 这个矩阵有不同的方差值。第二个变量的方差较大。在这种情况下，大部分方差集中在第二个变量上，因此可以用PCA简化为单维分布，且不会损失太多信息。
  - This matrix has different values of variance. The second variable has a higher variance. In this case, most of the variance is concentrated in the second variable, so it can be reduced to a unidimensional distribution using PCA without losing much information.
  - $\Sigma = [(0.5,0.1),(0.5,2)]$这个矩阵显示两个变量之间有轻微的正相关性。虽然有轻微的相关性，但由于第二个变量的方差仍然较大，大部分信息可能集中在一个方向上，因此PCA可能仍然能有效地将其简化为单维分布。
  - This matrix shows a slight positive correlation between the two variables.Although there is a slight correlation, PCA may still be effective in reducing it to a one-dimensional distribution since the variance of the second variable is still large and most of the information may be concentrated in one direction.
  - $\Sigma = [(0.5,0.1),(-0.5,2)]$这个矩阵在数学上是正定的,且两个方差差距较大，适用于PCA。特征值近似为方差，且行列式大于0，判断正定。This matrix is mathematically positive definite,and the difference between the two variances is large for PCA.The eigenvalues are approximated by the variances and the determinant is greater than 0. Positive definite is judged.

1.(c)

- (i)

1. **输入实例 Input Instances**：
   - The input features of the training samples are represented as $x_i \in \mathbb{R}^n$， Here, n is dimension of feature vector。
   - The known output value for each sample is represented as $y_i \in \mathbb{R}$。
2. **参数向量 Parameter Vectors**：
   - The parameters of the linear regression model are given by the weight vector $w \in \mathbb{R}^n$ and the bias term $b \in \mathbb{R}$。

- For each training sample i, the predicted output of the model can be expressed as $\hat{y_i} = w^T x_i + b$。

3. **平方损失函数 Squared Loss Function**:

   For each training instance i, the squared loss function is denoted as

$$L_i(w, b) = \frac{1}{2}(\hat{y_i} - y_i)^2$$

   Here $\frac{1}{2}$ is used to simplify the computation in the subsequent derivation:

4. **梯度计算 Gradient Calculation**:

   The core of stochastic gradient descent is to update the parameters w and b at each step and compute the gradient of the loss function $L_i(w, b)$ with respect to the parameters w and b. The loss function $L_i(w, b)$ is used to calculate the gradient of the loss function.

- **w gradient**：{\nabla}

$$\nabla_w L_i(w, b) = (\hat{y_i} - y_i) x_i$$

  $\hat{y_i} - y_i$ is the difference between the predicted and actual values.

- **b gradient**：

$$\nabla_b L_i(w, b) = \hat{y_i} - y_i$$

5. **更新规则 Update Rule**:

   Using the gradient calculated above, we can update the weights and bias. Let the learning rate be α and the update rule is as follows:：

- Update w：$w \leftarrow w - \alpha \nabla_w L_i(w, b)$
- Update b：$b \leftarrow b - \alpha \nabla_b L_i(w, b)$

  These updates are typically performed over multiple training **epochs**, each of which randomly selects samples across the training set to update the parameters.

6. **输出实例 Output Instances**：
   - testing vector is x_t final parameters are $\hat{w}, \hat{b}$
   - final output is $y_t = \hat{w}^T x_t + \hat{b}$

- (ii)

  - use the polynomial instead of the linear regression.
  - For example, it's a simple linear regression model $y = w_0 + w_1 x$, and we can extend it by increasing the high powers of x, such as $y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$
  - For alleviating this issue, we can use normalisation into loss function to limit the complexity of the model,such as L1 or L2 normalisation.

- Moreover, we also would use cross-validation to assess the generalisation ability of the model and choose an appropriate polynomial number of times. This can help find a model with a trade-off between bias and variance.
- Finally, we additionally use some feature selection methods to remove unimportant features.

- (iii)

    - I perfer to choose a) method. Due to with increase of t, the learning rate α would be small. It can help the algorithm to move forward quickly in the early stages and make finer adjustments in the later stages.

Database

(a)

Each record of S has size: 4B + 54B + 2B = 60B. Thus, the block factor of S will be :

$$\lfloor \frac{512\mathrm{B} - 10\mathrm{B}}{60\mathrm{B}} \rfloor = 8$$

and the number of block of S: $\mathrm{n_s} = \lceil \frac{1000}{8} \rceil = 125$

Each record of P has size: 4B + 8B + 8B + 4B= 24B. Thus, the block factor of P will be :

$$\lfloor \frac{512\mathrm{B} - 10\mathrm{B}}{24\mathrm{B}} \rfloor = 20$$

and the number of block of P: $\mathrm{n_p} = \lceil \frac{100000}{20} \rceil = 5000$

(b)(i)

Scan S at outer loop, and do binary search in inner loop P for each value $\lceil \log_2(\mathrm{n_p}) \rceil = 13$

For each tuple in S, perform the following steps: Use the join attribute from the current tuple in S to perform a binary search in P. If the binary search finds a matching tuple in P, join the current tuple from S with this matching tuple from P. Similar to the nested-loop join, store the join results in the output block until it is full. Once full, flush and continue.

$$\mathrm{r'_s} = \mathrm{NDV(ID, Seller)} = 200$$

with 100,000 records for 1000 sellers , each seller id will appeaer in 100 records in P on average. With 24B per record in P and 512B blocks, all 100 records whould be in 5 blocks. Thus, $\mathrm{n'_p} = 13 + 5 = 18$

The cost of this join is then: $\mathrm{n_s} + \mathrm{r'_s} \times \mathrm{n'_p} = 125 + 200 \times 18 = 3725$

(b)(ii)

Scan P at out loop, and do nested-loop join, so we should full scan S: $\mathrm{n'_p} = \mathrm{NDV(ID, P)} \times 5 + 13 = 1013$

Since the outer loop in the nested-loop join scans over relation P, then we need to:

1. Scan all blocks of P, nB−2 blocks at a time.
2. For each such block, scan all blocks of S for matches and produce join results.

3. Store join results in the output block until full; flush and continue.

The cost of this join is then:

$$n'_p + n'_s \times \lceil \frac{1013}{n_B - 2} \rceil = 1013 + 125 \times 51 = 7388$$

1. Computational linear algebra and optimisation

(a)

- (i)
  - Reviewing the data in Tabel 1, since the magnitude and distribution of each dimension varies widely(e.g., the mean value of dimension 3 is 78.1, while the mean value of dimension 8 is -159.3)
  - In this case, the aim is computing the similarity bewteen the tracks, so it's better to choose L2 norm to solve this problem.
  - For example, when we compute the distance of two feature vectorsl in non-normalisation situation, some features may contribute too much thee overall distance due to their large magnitude, which may mask the effect of other features.
  - L2 norm can be defined as $x_{norm} = \frac{x_i}{\sqrt{\sum_i^n x_i^2}}$
  - (ii)

```
x_upload_norm = x_upload / np.linalg.norm(x_upload)
X_dataset_norm = X_dataset / np.linalg.norm(X_dataset, axis=1,
keepdims=True)

distances = np.linalg.norm(X_dataset_norm - x_upload_norm,
axis=1)
closest_match_index = np.argmin(distances)
```

(b)

- (i)
  - the dimension of X is 101750 x 15
  - the dimension of w is 1 x 15
  - the dimension of y is 101750 x 1
- (ii)
  - A common approach is to use linear least squares. The goal of the least squares method is to minimise the sum of squares of the prediction errors, which can find the w.
  - First step is performing a singular value decomposition of the matrix X. $X = U\Sigma V^T$
  - Second step is computing $X^+ = V\Sigma^+U^T$ 其中，Σ^+ 是Σ的伪逆，即将Σ的非零元素替换为它们的倒数，零元素保持为零。
  - Finally, we can directly get w by $w = X^+y$

(c)

- (i)

- We can use the PCA to finish this 3D transformer.

```
# 计算每个特征的均值和标准差
  mean = np.mean(X, axis=0)
  std = np.std(X, axis=0)

  # 标准化数据
  X_normalized = (X - mean) / std

  # Step 2: 计算协方差矩阵
  cov_matrix = np.cov(X_normalized.T)

  # Step 3: 计算协方差矩阵的特征值和特征向量
  eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

  # Step 4: 对特征向量进行排序，并选择主成分
  # 对特征值从大到小排序，获取排序后的特征值的索引
  sorted_indices = np.argsort(eigenvalues)[::-1]

  # 选择前k个特征向量，k是你想要的主成分数量
  k = 3
  principal_components = eigenvectors[:, sorted_indices[:k]]

  # Step 5: 将原始数据转换到新的特征空间
  X_pca = X_normalized.dot(principal_components)
```

- (ii)
  - Figure 1 shows the eigenspectrum of the covariance of the matrix X. From this eigen-spectrum, it can be seen that the two largest eigenvalues are much larger than the other eigenvalues, which indicates that the variance in the dataset is mainly concentrated on these two eigenvectors.
  - To more efficiently, we can use the PCA method to decrease the dimension of data, to remove the not main features.
  - In this case, 2D will be more better, becuase two largest eigenvalues are much larger than the other eigenvalues.

(d)

- SGD can solve this issue
- For each training sample i, the predicted output of the model can be expressed as $\hat{g_i} = x_i w^T + b$。
- For each training instance i, the squared loss function is denoted as

$$L_i(w, b) = ||\hat{g_i} - g_i||_2^2$$

- **w gradient**：

$$\nabla_w L_i(w, b) = (\hat{g_i} - g_i)x_i$$

- **b gradient**：

$$\nabla_b L_i(w, b) = \hat{g_i} - g_i$$

-
  - Update w： $w \leftarrow w - \alpha \nabla_w L_i(w, b)$
  - Update b： $b \leftarrow b - \alpha \nabla_b L_i(w, b)$
  - α is the learning rate

2 Probabilities & Bayes rule

(a)

- (i_1)
  - $P(D|T) = \frac{P(T|D)P(D)}{P(T)}$ $P(T|D) = \frac{28}{28+3}$ $P(D) = \rho$
  - $P(T) = P(T|D)P(D) + P(T|D^-)P(D^-) = \frac{28}{28+3}\rho + \frac{12}{12+89}(1-\rho)$
  - $P(D|T) = \frac{\frac{28}{28+3}\rho}{\frac{28}{28+3}\rho + \frac{12}{12+89}(1-\rho)}$
- (i_2)
  - $P(D|T^-) = \frac{P(T^-|D)P(D)}{P(T^-)}$ $P(T^-|D) = \frac{3}{28+3}$ $P(D) = \rho$
  - $P(T^-) = P(T^-|D)P(D) + P(T^-|D^-)P(D^-) = \frac{3}{28+3}\rho + \frac{89}{12+89}(1-\rho)$
  - $P(D|T^-) = \frac{\frac{3}{28+3}\rho}{\frac{3}{28+3}\rho + \frac{89}{12+89}(1-\rho)}$
- (ii)
  - $S_p = \frac{TN}{TN+FP} = \frac{89}{89+28} = 76$
  - Because of the high sensitivity S_p, it is appropriate for regular testing of people working with vulnerabel population.
  - the rate of true diseased people is 24%， This is not the case when the treatment has serious side effects, and only 24 per cent of people actually need to be treated.
  - As for applying to the whole population to find all diseasd people is not suitable, due to high rate of healthy negative, and it's diffcult for goverment to treat these individuals.
- (iii)
  - 联立方程组 得ρ =%
  - $$0.7 = \frac{P(T|D)\rho}{P(T)} \; 0.01 \qquad = \frac{P(T^-|D)\rho}{P(T^-)} \; P(T) = P(T|D)\rho + (1 - P(T|D))(1-\rho)$$

(b)

- (i) 见卷子

Database

(a)

Each record of S has size : 8B + 40B + 2B = 50B Thus, the block factor of S will be :

$$\lfloor \frac{512B - 10B}{50B} \rfloor = 10$$

and the number of block of S : $n_b = \frac{1000}{10} = 100$

Each record of M has size : 8B + 2B + 2B + 8B= 20B Thus, the block factor of M will be :

$$\lfloor \frac{512B - 10B}{20B} \rfloor = 25$$

and the number of block of S : $n_b = \frac{100000}{25} = 4000$

(b)

In this case, we use the binary search and nested-loop join algorithm, there are two strategies which are:

- (i) Outer loop S, inner loop M , using binary search.
- (ii) Outer loop M , inner loop S, using nested-loop join algorithm.

For (i): Scan S at the outer loop, do binary search on M for each value $\lceil \log_2(n_M) \rceil = 12$

with 100,000 records for 1,000 students , each studentID will appear in 100 records in M on average.

With 20B per record in M and 512B blocks, all 100 records should be in 4 block on average.

The cost of this join is then: $n_S + r'_S \times 16 = 100 + 200 * 16 = 3300$

For (ii): Scan M at the outer loop， full scan on S for each value.

$$n_{M'} = NDV(ID, M) \times 4 + 12 = 200 \times 4 + 12 = 812$$

The cost of this join is then:

$$n'_M + n'_S \times \lceil \frac{n'_M}{n_B - 2} \rceil = 812 + 100 \times 41 = 4912$$

Database

(a) (i)

Each reocrd has siez: 1B + 51B + 8B = 60B
Thus, the block factor will be: $\frac{512\text{B}-24\text{B}}{60\text{B}} = 8$ records per block.
and the number of block $n_b = \frac{1000}{8} = 125$ blocks.

(a)(ii)

- for (a) is a heap file， it would search all of blocks, so the cost of block accesses is then: cost = 125 block accesses.
- for (b) is a sequential file, order by primary key, it would use the binary search to solve the problem, the best cost = 2 $andtheworstcost$ =\log_2(b) = 7$, so the average cost is 4.5 block accesses.
- for (c) is hash file, the cost would be 1 block accesses.

(b)(i)

Join D with E

- Join intermediate (I), $I = D \bowtie E$
- $js_{I=D\bowtie E} = \frac{1}{\max(NDV(Mgr_SSN,D),NDV(SSN,E))} = \frac{1}{1000}$
- $jc_{I=D\bowtie E} = js_{I=D\bowtie E} \times |E||C| = \frac{1}{1000} * 1000 * 10 = 10$

In this case, we use the nested-loop join algorithm, there are two strategies, whcih are :

- (a) Outer loop D, inner loop E
- (b) Outer loop E, inner loop D

For (a):

- Outer loop D, and do the nested-loop join algorithm with E
- The cost of this join is then:

$$n_D + n_E \times \lceil \frac{n_D}{n_B - 2} \rceil = 50 + 100 \times 5 = 550$$

For (b):

- Outer loop E, and do the nested-loop join algorithm with D
- The cost of this join is then:

$$n_E + n_D \times \lceil \frac{n_E}{n_B - 2} \rceil = 100 + 50 \times 10 = 600$$

As you can see it, the cost of strategy (a) is better than (a), and the total cost is then:
$\text{cost} = 550 + \frac{jc_{I=D\bowtie E}}{bfr_{RS}} = 550 + \frac{10}{10} = 551$.

(b)(ii)

In this case, we use the two index-based nested-loop join strategies, which are :

- (a) Outer loop D, inner loop E
- (b) Outer loop E, inner loop D
-

Additionally, we ignore the cost of storage to disk.

For (a):

- Outer loop E, and do the two index-based nested-loop joinalgorithm with D
- The cost of this join is then:

$$r_E \times t = 1000 \times 3 = 3000$$

For (b):

- Outer loop D, and do the two index-based nested-loop joinalgorithm with E
- The cost of this join is then:

$$r_D \times t = 10 \times 3 = 30$$

As you can see the cost of (a) is larger than (b). Thus, Table E would be more subitable using these indexs.