

Machine Learning Notes

Andrew Valentini

September - December 2026

Contents

1	Linear Regression	2
2	Cross-Validation	4
3	Bayesian Optimization	6
4	Linear Classification Models	12
4.1	Naive Bayes Classifier	12
4.2	Gaussian Mixture Models	13
5	Exercises	16
5.1	Discriminant Function Derivation	16
5.2	Converting an Observed Data Likelihood into an Objective Function Expression	17
5.3	Expectation Maximization Algorithm Practice	18
6	To Know Cold	19

1 Linear Regression

Want: a model to learn the relationship between data point $\{x_i\}$ and target parameters $\{t_{ij}\}$, where t_{ij} represents the j th target parameter corresponding to the i th data point (your data could be described by any number of parameters which you want a model to learn).

Example: x_i could be a person with features $x_i = \{\text{height, weight, age, sex, etc.}\}$. This representation of the data points x_i is called the feature space $\phi(x_i)$. In this example, target parameters t_{ij} could be a person's probability of cancer, life expectancy, etc. Each of these targets could be captured separately by $t_{i1}, t_{i2}, \dots, t_{iN}$ for N possible targets you wish to map the observed data onto with a learned model.

Combined, $\{(x_i, t_{ij})\}_{i,j=1}^{N,M}$ is called the **feature space** for N distinct samples in your data and M distinct target parameters. With N samples and M desired features, the total feature space can be written as an $N \times M$ space.

$$\Phi = \begin{pmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_N)^T \end{pmatrix} \quad (1)$$

For example, if $\phi(x_i) = x_i$, then

$$\Phi = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1M} \\ x_{21} & x_{22} & \dots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{NM} \end{pmatrix} \quad (2)$$

Linear Regression assumes the output of the model is a linear combination of the input data

$$y(x) = \sum_{i=1}^N w_i x^i \quad (3)$$

The linear regression task can also be written as a matrix product

$$\mathbf{y} = X\mathbf{w} \quad (4)$$

Since $\mathbf{w}$ are free parameters and we wish to minimize our model's error in predicting the target values t_i , we define an objective function,

$$J(\mathbf{w}) = \sum_{i=1}^N \frac{(t_i - y(x_i))^2}{N}, \quad (5)$$

a function which we seek to minimize to find the optimal set of $\mathbf{w}$ to best align the model with the data. Note that we assumed only one target value in this expression for simplicity.

To find the optimal $\mathbf{w}$ for a given linear regression task, we take

$$\frac{\partial J}{\partial W} = 0 \quad \text{for} \quad J(W) = \frac{1}{N}(T - XW)^T(T - XW)$$

Expanding $J(W)$:

$$J(W) = \frac{1}{N} (T^T T - T^T XW - W^T X^T T + W^T X^T XW)$$

Taking the partial derivative with respect to W :

$$\frac{\partial J}{\partial W} = \frac{1}{N} (-2X^T T + 2X^T XW)$$

Setting this to zero:

$$\begin{aligned} -2X^T T + 2X^T XW &= 0 \\ \implies X^T XW &= X^T T \\ \implies \boxed{W^* = (X^T X)^{-1} X^T T} \end{aligned}$$

To prevent overfitting, we can also add a **regularizer** term, which modifies the form of the objective function $J(\mathbf{w})$. One regularizer is called the Ridge regularizer. Adding this to the objective function looks like:

$$J(w) = \frac{1}{N}|T - XW|^2 + \frac{\lambda}{N}|W|^2,$$

which results in a form of the optimal W^* values written like

$$W^* = (X^T X + \lambda \mathbb{1}_n)^{-1} X^T T$$

obtained by the same process as before.

2 Cross-Validation

Goal: Find the optimal model hyperparameters for a particular regression task, which have so far been λ , the strength of regularization, and m , the model order.

To do this, the given data is typically given an 80/20 split into an *operational* and *testing* set, where the operational data is then further split into *training* and *validation*. The validation set is added to evaluate the model's performance for a given set of hyperparameters on unseen data. The optimal model hyperparameters are found by evaluating its performance under some error metric E and finding the set of parameters $X = \{\lambda, m\}$ that give $\frac{\partial E}{\partial X} = 0$ for the *validation* dataset. The error of the model computed for the training data will always decrease due to overfitting, whereas there will always be some minimum in the error of the validation's error, marking the parameter values where the model has begun to overfit on the training data. With the optimal hyperparameter values chosen, the overall performance of the model is then evaluated on the testing set.

Common error metrics for regression:

1. Mean-squared error:

$$\sum_{i=1}^N \frac{(\phi(x_i) - t_i)^2}{N} \quad (6)$$

2. Root mean-squared error:

$$\sqrt{\sum_{i=1}^N \frac{(\phi(x_i) - t_i)^2}{N}} \quad (7)$$

which requires that the differences $\phi(x_i) - t_i$ be positive to keep the overall error positive.

3. Mean absolute error:

$$\left| \sum_{i=1}^N \frac{\phi(x_i) - t_i}{N} \right| \quad (8)$$

which is ideal when you want your error metric to be unaffected by large errors (MSE's value will increase quadratically as error increases).

Techniques for cross-validation:

1. K -fold cross validation
2. $K \times 2$ cross validation
3. Stratified cross validation

Example of 4-fold cross validation:

First split the operational data into 4 equal subsets of the data $S = S_1 + S_2 + S_3 + S_4$, run a grid search over all combinations of possible parameter values (within a defined range) $m \times \lambda = \{1, \dots, 10\} \times \{0, .001, .01, 1\}$, evaluate an error metric over the model's performance for the training and validation data over each subset, and choose the set of parameters

Drawn out:

For $\lambda_j, m_j \in m \times \lambda = \{1, \dots, 10\} \times \{0, .001, .01, 1\}$, evaluate

1. Training set $\{S_1, S_2, S_3\} \Rightarrow W^{(1)} \Rightarrow \text{MSE of } x_t^{(1)}$
Validation set $\{S_4\}$ under $W^{(1)} \Rightarrow \text{MSE of } x_v^{(1)}$
2. Training set $\{S_1, S_2, S_4\} \Rightarrow W^{(2)} \Rightarrow \text{MSE of } x_t^{(2)}$
Validation set $\{S_3\}$ under $W^{(2)} \Rightarrow \text{MSE of } x_v^{(2)}$
3. Training set $\{S_1, S_3, S_4\} \Rightarrow W^{(3)} \Rightarrow \text{MSE of } x_t^{(3)}$
Validation set $\{S_2\}$ under $W^{(3)} \Rightarrow \text{MSE of } x_v^{(3)}$
4. Training set $\{S_2, S_3, S_4\} \Rightarrow W^{(4)} \Rightarrow \text{MSE of } x_t^{(4)}$
Validation set $\{S_1\}$ under $W^{(4)} \Rightarrow \text{MSE of } x_v^{(4)}$

The total MSE performance of the model under a particular set of hyperparameters $\lambda_j, m_j \in \lambda \times m$ is the average of every fold's MSE $x_t^{(i)}$, $\sum_{i=1}^4 x_t^{(i)} / 4$ under these hyperparameter values. We also take the variance of the validation MSE values $x_v^{(i)}$ for each set of hyperparameters to determine how subject the model is to overfitting under them; $\sqrt{\sum_{i=1}^4 (x_t^{(i)})^2} / 4$, since a model with a strong average MSE performance across all folds with a large variance in the MSE values would indicate the model greatly depends on which subset of data it is trained on (for the particular set of hyperparameters $\{\lambda_j, m_j\}$).

The desired set of hyperparameters $\{\lambda_j, m_j\}$ is therefore that which has a low average MSE across all folds *and* low variance across these folds.

3 Bayesian Optimization

Bayesian optimization offers another method for finding ideal weight values. Again consider the goal of finding the set of W which minimize the objective function $J(W)$. Mathematically, the goal is again to find

$$\arg \min_W \frac{1}{N} \sum_{i=1}^N (t_i - y_i)^2 \quad (9)$$

for an objective function governed by the mean squared error. First note that finding W that minimizes the objective function is the same thing as finding W that maximizes the negative of the objective function:

$$\arg \min_x f(x) = \arg \min_x (-f(x)). \quad (10)$$

If we exponentiate this and multiply by $\frac{1}{\sqrt{2\pi}}$, we will get a form that looks like a Gaussian

$$\begin{aligned} & \arg \max_W \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{1}{N} \sum_{i=1}^N (t_i - y_i)^2 \right) \\ &= \arg \max_W \frac{1}{\sqrt{2\pi}} \prod_{i=1}^N \exp \left(-\frac{1}{N} (t_i - y_i)^2 \right) \\ &= \arg \max_W \prod_{i=1}^N \mathcal{N}(t_i | y_i(W), 1) \end{aligned} \quad (11)$$

where we have used the notation $\mathcal{N}(x|\mu, \sigma^2)$, where x is the random variable we are sampling on the Gaussian distribution of mean μ and variance σ^2 . Note that optimizing this expression will still result in the $W^* = (X^T X)^{-1} X^T T$ form we found earlier; this is just a different way of getting there.

If we instead let our objective function be described by the mean absolute error, we find

$$\begin{aligned} & \arg \max_W \prod_{i=1}^N \exp \left(-\frac{1}{2} |t_i - y_i| \right) \\ & \equiv \arg \max_W \mathcal{L}(t|y, 1), \end{aligned} \quad (12)$$

which is the exact form of a random variable t_i being drawn from the **Laplacian function** $\mathcal{L}(\mu, b)$, whose probability density function looks like

$$f_X(b) = \frac{1}{2b} \exp\left(\frac{-|x - \mu|}{b}\right). \quad (13)$$

Notice under this expression of weight optimization that when using the mean absolute error objective function, t values closer to the mean μ are more likely to be chosen than when using the mean squared error objective function since the Laplacian function is more highly peaked near μ than a Gaussian distribution.

We have shown that we can instead think about minimizing objective functions as minimizing probability distributions

$$\arg \min_W J(W) \equiv \arg \min_W P(t|W), \quad (14)$$

which is called the Bayesian interpretation of the objective function.

Extending this new expression of the objective function to include a regularizer term:

$$J(W) = \frac{1}{2} \sum_{i=1}^N (t_i - y_i)^2 + \frac{\lambda}{N} \sum_{j=1}^M w_j^2 \quad (15)$$

$$\begin{aligned} \implies \arg \max_W \exp\left(-\frac{1}{N} \sum_i (t_i - y_i)^2 + \frac{\lambda}{N}\right) \\ \equiv \arg \max_W \prod_i \mathcal{N}(t_i|y_i, 1) \prod_j \exp\left(\frac{\lambda}{N} w_j^2\right) \\ = \arg \max_W \prod_i \mathcal{N}(t_i|y_i, 1) \prod_j \mathcal{N}(w_j|0, \frac{1}{\lambda}) \\ \equiv \arg \max_W P(t|W)P(W) \end{aligned} \quad (16)$$

where we have used $P(W)$ to define the **prior** distribution on the weights and $P(t|W)$ is the likelihood, which is the distribution around t values given a set of weights W . Notice that, as expected, as λ increases, the variance of the prior increases, making the prior dominate the normal distribution which t_i 's are being sampled from, meaning the regularizer term dominates the expression for finding the optimal W values.

We have thus further expressed the optimization problem as

$$\arg \min_W J(W) \equiv \arg \min_W P(t|W)P(W) \quad (17)$$

and we can use Bayes' theorem to say

$$\arg \min_W P(t|W)P(W) = P(t) \arg \min_W P(W|t) \quad (18)$$

$$\implies \arg \min_W P(W|t) = \frac{\arg \min_W P(t|W)P(W)}{P(t)} \quad (19)$$

which is called the **maximum a posteriori algorithm**. This is an especially important technique for finding optimal W values in scenarios when you have limited data and therefore cannot learn trends from the data alone (introducing the prior gets rid of this need).

Example: Consider a case of a limited dataset of coin flips $\{x_i\}_{i=1}^N$ where $x_i = \{1, 1, 1, 0, 0, 0\}$ for example. Say we want to determine which model generated these observations. We would expect this to look like a simple Bernoulli distribution for a fair coin.

First define the probabilities of observing 1 (heads) and 0 (tails) as

$$P(x = 1|\mu) \equiv \mu \text{ \& } P(x = 0|\mu) \equiv 1 - \mu. \quad (20)$$

The data likelihood for any observation x is then given as

$$P(x|\mu) = \mu^x(1 - \mu)^{1-x} \quad (21)$$

We know for a fair coin that $\mu \equiv 1 - \mu = .5$, but let us assume μ to be a free parameter to be learned from the maximum likelihood estimation method.

For the maximum likelihood estimation algorithm, we first define a total data observed likelihood function

$$\begin{aligned} \mathcal{L}^0 &= \prod_{i=1}^N P(x_i|\mu) = \prod_{i=1}^N \mu^{x_i}(1 - \mu)^{1-x_i} \\ &= \mu^{\sum_i x_i} (1 - \mu)^{\sum_i (1-x_i)} = \mu^{\sum_i x_i} (1 - \mu)^{N - \sum_i x_i} \end{aligned} \quad (22)$$

Converting this to a log-likelihood to make computation easier:

$$\begin{aligned} \ln \mathcal{L}^0 &= \ln \left(\mu^{\sum_i x_i} \right) + \ln \left((1 - \mu)^{N - \sum_i x_i} \right) \\ &= \sum_i x_i \ln(\mu) + \left(N - \sum_i x_i \right) \ln(1 - \mu) \end{aligned} \quad (23)$$

Finding the optimal μ value:

$$\frac{\partial \ln \mathcal{L}^0}{\partial \mu} = \sum_i \frac{x_i}{\mu} - \frac{(N - \sum_i x_i)}{1 - \mu} = 0 \quad (24)$$

$$\implies \mu_{\text{MLE}} = \frac{\sum_i x_i}{N} \quad (25)$$

Now using the maximum a posteriori algorithm to find the optimal μ value, we first need a prior distribution. Choose

$$\mu \sim \text{Beta}(a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1} (1-\mu)^{b-1} \quad (26)$$

where $a, b > 0$.

Setting up the observed likelihood for this MAP algorithm:

$$\mathcal{L}^0 = \left(\prod_{i=1}^N P(x_i | \mu) \right) \left(\prod_{j=1}^M P(\mu_j) \right) \quad (27)$$

where since we only have one parameter to optimize, $M = 1$. Using the same steps as before on the product of terms with sums in the exponential terms for $P(x_i | \mu)$, we get

$$\mathcal{L}^0 = \mu^{\sum_i x_i} (1-\mu)^{N-\sum_i x_i} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1} (1-\mu)^{b-1} \quad (28)$$

Since the term with the Gamma functions is just a constant, we can write

$$\mathcal{L}^0 = C \mu^{\sum_i x_i + a - 1} (1-\mu)^{N - \sum_i x_i + b - 1} \quad (29)$$

which is just the form of another Beta distribution with its constants reparameterized:

$$a' = \sum_i x_i + a \text{ \& } b' = N - \sum_i x_i + b \quad (30)$$

giving us

$$\mathcal{L}^0 = C \mu^{a'-1} (1-\mu)^{b'-1}. \quad (31)$$

Note that whenever a posterior $P(y|t)$ is proportional to the prior, we call this a **conjugate prior relationship**.

$$\implies \ln \mathcal{L}^0 = \ln C + (a' - 1) \ln(\mu) + (b' - 1) \ln(1 - \mu) \quad (32)$$

$$\begin{aligned} \implies \frac{\partial \mathcal{L}^0}{\partial \mu} &= \frac{a' - 1}{\mu} + \frac{b' - 1}{1 - \mu} = 0 \\ \implies \mu_{\text{MAP}} &= \frac{\sum_i x_i + a - 1}{N + a + b - 2} \end{aligned} \quad (33)$$

This μ_{MAP} value is also data-driven, but it also now encodes additional information from the prior (Beta) distribution on a, b . Notice that if there was no prior information, the form of μ_{MAP} would reduce to μ_{MLE} .

We now wish to explicitly show that representing the objective function as a probability distribution in the Bayesian interpretation yields the same result for the optimal parameters W^* that we found from linear regression. Formally, we will construct and then minimize an objective function of a general Gaussian distribution and show this reduces to the solution $W^* = (X^T X)^{-1} X^T T$ found from linear regression.

Writing random variable vector $X = [x_1, \dots, x^k]$ being sampled from a general Gaussian distribution with mean μ and variance Σ

$$\mathcal{N}(X|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X - \mu)^T \Sigma^{-1} (X - \mu)\right) \quad (34)$$

where Σ is the **covariance matrix**, with terms

$$\Sigma_{ij} = \text{cov}[X_i, X_j] := \mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])] \quad (35)$$

which physically measures the joint variability of two variables X_i and X_j .

Notice that if $X_i = X_j$,

$$\begin{aligned} \Sigma_{ii} &= \mathbb{E}[(X_i - \mathbb{E}[X_i])^2] = \mathbb{E}[X_i^2 - 2X_i\mathbb{E}[X_i] + \mathbb{E}[X_i]^2] \\ &= \mathbb{E}[X_i^2] - 2\mathbb{E}[X_i]\mathbb{E}[X_i] + \mathbb{E}[\mathbb{E}[X_i]^2] \\ &= \mathbb{E}[X_i^2] - 2\mathbb{E}[X_i]\mathbb{E}[X_i] + \mathbb{E}[X_i]^2 \\ &= \mathbb{E}[X_i^2] - \mathbb{E}[X_i]^2 \equiv \sigma_i^2. \end{aligned} \quad (36)$$

Setting $\Sigma = 1$ for our case of a Gaussian with unity variance, we get

$$\ln \mathcal{N}(t|y, 1) = \frac{1}{2} \ln\left(\frac{1}{\sqrt{2\pi}}\right) - \frac{1}{2}|t - y|^2 \quad (37)$$

after applying the log.

Setting the partial with respect to W equal to 0:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W} &= \frac{\partial}{\partial W} \left(\frac{1}{2} \ln \left(\frac{1}{\sqrt{2\pi}} \right) - \frac{1}{2} |t - XW|^2 \right) = 0 \\ \implies W^* &= (X^T X)^{-1} X^T t\end{aligned}\tag{38}$$

which is the same solution we found with normal linear regression, showing that representing objective functions as probability distributions to be maximized under the Bayesian interpretation, $\arg \max_W \mathcal{N}(t|y, 1)$, is equivalent to the methodology developed in linear regression.

This connection also allows us to find optimal W values very quickly by using information developed in linear regression. If we are instead maximizing the product of two Gaussians:

$$\arg \max_W \mathcal{N}(t|y, 1) \mathcal{N}(W|0, 1/\lambda),\tag{39}$$

we can now immediately recognize the solution to this to be equivalent to the linear regression task with the ridge regularizer:

$$W^* = (X^T X + \lambda \mathbf{1})^{-1} X^T t\tag{40}$$

4 Linear Classification Models

4.1 Naive Bayes Classifier

Goal: Assign data to distinct classes. In other words, find the probability of a data point x_i belonging to a class C_k .

We do this by assigning probability distributions on the observed data for points to belong to a given class. Usually these class probability distributions are just Gaussians:

$$P(x_i|C_k) = \mathcal{N}(x_i|\mu_k, \Sigma_k) \quad (41)$$

Where $P(x_i|C_k)$ is the likelihood; the probability that a random data point x_i would be pulled from class C_k . If we want to find the probability of each class existing in the total dataset, we use Bayes Theorem to convert the likelihood into a posterior:

$$P(C_k|x_i) = \frac{P(x_i|C_k)P(C_k)}{P(x_i)} \quad (42)$$

where $P(C_k)$, the prior, determines how frequently instances from class C_k are drawn from the full dataset (how present C_k is in the dataset). $P(x_i)$ is the normalization factor to keep $P(C_k|x_i) \in [0, 1]$, which is a sum over classes $\sum_j P(x_i|C_k)P(C_k)$.

The “naive” assumption here comes from assuming that the data features features are independent from each other. In other words:

$$P(X|C_k) = \prod_{i=1}^N P(x_i|C_k) \quad (43)$$

In this formatting of the classification problem, we have introduced two new hyperparameters to be chosen, which are the forms of the likelihood and priors. Using this method, we determine a datapoint x_i to belong to class C_k for which the posterior probability is maximized:

$$P(C_1|x_i) \leq_{x_i \in C_1}^{x_i \in C_2} P(C_2|x_i) \quad (44)$$

and in general for k classes:

$$x_i \in C_k \quad \text{if} \quad k = \arg \max P(C_k|x_i) \quad (45)$$

We can also use this Bayesian design of the classification problem to define a closed form **discriminant** function $g(\mathbf{x})$ that clearly distinguishes the regions where class membership is defined in the feature space.

We motivate the construction of $g(\mathbf{x})$ by first considering a two class classification problem, where $x_i \in C_1$ which implies

$$\begin{aligned} P(C_1|x_i) > P(C_2|x_i) &\implies \ln\left(\frac{P(C_1|x_i)}{P(C_2|x_i)}\right) > 0 \\ &\implies g(\mathbf{x}) \equiv \ln\left(\frac{P(C_1|x_i)}{P(C_2|x_i)}\right) \end{aligned} \quad (46)$$

$$\implies g(\mathbf{x}) = \ln(P(C_1|x_i)) - \ln(P(C_2|x_i)) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x}) > 0 \quad (47)$$

meaning that regions in the feature space where the above expression is greater than zero is where our model is classifying data points to belong to class C_1 . All regions in the data where the discriminant function $g(\mathbf{x}) < 0$ is then the region in the data where our model is defining all points with C_2 membership to exist. Independently, $g_1(\mathbf{x})$ defines the space in the data where class membership is to class C_1 .

4.2 Gaussian Mixture Models

This is an unsupervised classification method which is especially useful for the existence of multiple clusters in data and when the exact class descriptions may not be known. In this model, the likelihood becomes a weighted sum of K Gaussians:

$$P(x_i|\theta) = \sum_{k=1}^K \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k) \quad (48)$$

where the π_k are the prior probabilities that a sample belongs to class k , subject to the constraints $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$.

Just as before, to find the optimal parameter values $\theta_k = \{\pi_k, \mu_k, \Sigma_k\}$ for this classification problem, the total data likelihood is constructed over all samples;

$$\mathcal{L} = \prod_{i=1}^N \sum_{k=1}^K \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k), \quad (49)$$

where the usual process would then be to take $\frac{\partial \ln \mathcal{L}}{\partial \theta} \Big|_{\theta^*} = 0$. This has now become an intractable problem, however, since we are taking the product of

a sum. The method for dealing with this is called the **Expectation Maximization Algorithm (EM)**. This algorithm introduces a latent variable z_{ik} which represents the belongingness of datapoint i to class k (a one-hot vector where the only non-zero entry is for the class k that sample i actually belongs to). We can then write the likelihood as

$$\mathcal{L} = \prod_{i=1}^N \prod_{k=1}^K \left(\pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k) \right)^{z_{ik}} \quad (50)$$

since when a sample i belongs to class k , the only remaining term in the products will be $\pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k)$. This is equivalent to a discrete Dirac delta function, but is placed in the exponent so that the log is easier to take.

When the log of this expression is taken, we get

$$\ln \mathcal{L} = \sum_{i=1}^N \sum_{k=1}^K z_{ik} \left(\ln(\pi_k) + \ln \left(\mathcal{N}(x_i | \mu_k, \Sigma_k) \right) \right) \quad (51)$$

This is now the quantity we want to use to find the optimal $\theta_i = \{\pi_i, \mu_i, \Sigma_i\}$, but z_{ik} is still an unknown latent variable, so we can average over its contributions using an expectation value of the previous expression, giving us the EM objective function:

$$Q(\theta, \theta^{(t)}) = \mathbb{E}_{z_i} \left[\sum_{i=1}^N \sum_{k=1}^K z_{ik} \left(\ln(\pi_k) + \ln \left(\mathcal{N}(x_i | \mu_k, \Sigma_k) \right) \right) \middle| X, \theta^{(t)} \right] \quad (52)$$

(given the full dataset X and initial parameter values $\theta^{(t)}$, we update the parameters according to this algorithm).

Since the expectation value only acts on terms that contain dependence on z_{ik} , we get

$$Q(\theta, \theta^{(t)}) = \sum_{i=1}^N \sum_{k=1}^K \mathbb{E} \left[z_{ik} \middle| X, \theta^{(t)} \right] \left(\ln(\pi_k) + \ln \left(\mathcal{N}(x_i | \mu_k, \Sigma_k) \right) \right) \quad (53)$$

To further express the expectation value, consider a two-class problem where sample $x_i \in C_1$, meaning $\mathbb{E}[z_{ik} | x_i, \theta^{(t)}] = 1 \cdot P(z_{ik} = 1 | x_i, \theta) + 0 \cdot P(z_{ik} = 0 | x_i, \theta)$

$0|x_i, \theta) = P(z_{ik} = 1|x_i, \theta)$, which we can further express using Bayes theorem:

$$\begin{aligned}
P(z_{ik} = 1|x_i, \theta) &= \frac{P(x_i|z_{ik} = 1, \theta)P(z_{ik} = 1)}{P(x_i)} \\
&= \boxed{\frac{\mathcal{N}(x_i|\mu_k^{(t)}, \Sigma_k^{(t)})\pi_k^{(t)}}{\sum_{j=1}^K \mathcal{N}(x_i|\mu_j^{(t)}, \Sigma_j^{(t)})\pi_j^{(t)}} \equiv \gamma_{ik}} \quad (54)
\end{aligned}$$

We then have the following form of the objective function:

$$\boxed{Q(\theta, \theta^{(t)}) = \sum_{i=1}^N \sum_{k=1}^K \gamma_{ik} \left(\ln(\pi_k) + \ln \left(\mathcal{N}(x_i|\mu_k, \Sigma_k) \right) \right)} \quad (55)$$

5 Exercises

5.1 Discriminant Function Derivation

Problem 1: Find the discriminant function $g_1(\mathbf{x})$ for a 2 class *equal-prior* classification problem where

$$C_1 \sim \mathcal{N}(\mu_1, \Sigma_1) \quad \text{with} \quad \mu_1 = \begin{pmatrix} 3 \\ 6 \end{pmatrix} \quad \& \quad \Sigma_1 = \begin{pmatrix} 1/2 & 0 \\ 0 & 2 \end{pmatrix} \quad (56)$$

(find the function that defines where in $\mathbf{x} = (x_1, x_2)$ the model should define a point x_i to belong to class C_1).

This tells us we want to find the exact form of

$$g_1(\mathbf{x}) = \ln(P(C_1|\mathbf{x})) = \ln\left(\frac{P(\mathbf{x}|C_1)P(C_1)}{P(\mathbf{x})}\right) \quad (57)$$

and we can drop the $P(\mathbf{x})$ term since this is just a normalization factor both discriminant functions $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$ will contain, and we can drop the $P(C_1)$ term since this problem considers an equal-prior case, where samples are equally-likely to be pulled from either C_1 or C_2 , meaning neither term will help us distinguish between the two classes, and we can drop their dependence in the discriminant function, leaving us with

$$g_1(\mathbf{x}) \sim \ln(P(\mathbf{x}|C_1)) \quad (58)$$

Adding the 2-dimensional Gaussian form of the likelihood and plugging the given μ_1 and Σ_1 in:

$$\begin{aligned} g_1(\mathbf{x}) &\sim \ln\left(\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)\right) \\ &= -\frac{d}{2} \ln(2\pi) - \frac{1}{2} \ln(|\Sigma|) - \frac{1}{2} \left((\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right) \end{aligned} \quad (59)$$

$$|\Sigma| = 1 \quad \& \quad \Sigma^{-1} = \begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix} \quad (60)$$

$$\begin{aligned} \implies g_1(\mathbf{x}) &\sim -\ln(2\pi) - \frac{1}{2} \left((x_1 - 3 \quad x_2 - 6) \begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} x_1 - 3 \\ x_2 - 6 \end{pmatrix} \right) \\ &\sim -\ln(2\pi) - \frac{1}{2} \left(2(x_1 - 3)^2 + \frac{1}{2}(x_2 - 6)^2 \right) = g_1(\mathbf{x}) \end{aligned} \quad (61)$$

Note that you would carry out the same method for $g_2(\mathbf{x})$ if you were given an exact form for the likelihood on C_2 , $P(\mathbf{x}|C_2)$.

5.2 Converting an Observed Data Likelihood into an Objective Function Expression

Problem 1: Convert the given observed data likelihood:

$$\mathcal{L} = \left(\prod_{i=1}^N \frac{1}{\sqrt{2\pi}} e^{-(t_i - y_i)^2/2} \right) \left(\prod_{j=1}^M \frac{1}{2b} e^{-|w_j|/b} \right) \quad (62)$$

into an optimization problem for linear regression.

In the language of observed data likelihoods, we want to maximize a posterior probability distribution to find the optimal model parameters, $\arg \max_W \mathcal{L}$, but in linear regression optimization, we want the model parameters which minimize the error of some objective function, $\arg \min_W J(W)$, so the task is to convert this from an $\arg \max \rightarrow \arg \min$ problem and transform $\mathcal{L} \rightarrow J(W)$.

To do this, we start with the observed data likelihood optimization expression and first get rid of some unnecessary scaling constants

$$\begin{aligned} \arg \max_W \mathcal{L} &\propto \arg \max_W \left[\left(\prod_{i=1}^N e^{-(t_i - y_i)^2/2} \right) \left(\prod_{j=1}^M e^{-|w_j|/b} \right) \right] \\ \implies \arg \max_W \ln \mathcal{L} &\propto \arg \max_W \left[- \sum_{i=1}^N (t_i - y_i)^2 - \sum_{j=1}^M \frac{|w_j|}{b} \right] \end{aligned} \quad (63)$$

And using the fact that $\arg \max(-f(x)) = \arg \min f(x)$, we get

$$\arg \max_W \mathcal{L} \propto \arg \max_W \left[\sum_{i=1}^N (t_i - y_i)^2 + \sum_{j=1}^M \frac{|w_j|}{b} \right] \quad (64)$$

And this is now in a form which can be written as the minimization of an objective function:

$$\arg \max_W \mathcal{L} \propto \arg \max_W \left(|\mathbf{t} - \mathbf{y}|^2 + \frac{1}{b} |\mathbf{W}| \right) \quad (65)$$

So we see that the $\exp(-|w_j|/b)$ term in the observed data likelihood is equivalent to adding a Lasso regularization to an MSE objective function.

5.3 Expectation Maximization Algorithm Practice

Problem 1: Consider a mixture model with 2 components ($K = 2$), where one component is Gaussian-distributed, $P_1(X|\theta) = \mathcal{N}(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-(x-\mu)^2/2\sigma^2)$, and the other component is Exponential-distributed, $P_2(X|\theta) = E(x|\lambda) = \lambda e^{-\lambda x}$.

Write the complete data likelihood by adding the latent variable z_i :

$$\begin{aligned}\mathcal{L} &= \prod_{i=1}^N \prod_{k=1}^K \left(\pi_k P_k(x|\theta_k) \right)^{z_{ik}} \\ &= \prod_{i=1}^N \left(\pi_1 P_1(x|\theta_1) \right)^{z_{i1}} \left(\pi_2 P_2(x|\theta_2) \right)^{z_{i2}} \\ &= \prod_{i=1}^N \left(\pi_1 \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \right)^{z_{i1}} \left(\pi_2 \lambda e^{-\lambda x} \right)^{z_{i2}}\end{aligned}\tag{66}$$

Now derive the update equation for the parameter λ :

Since we only care for the term that depends on λ , we can drop the objective function's dependence on the $k = 1$ Gaussian term:

$$\begin{aligned}Q(\theta, \theta^{(t)}) &\propto \sum_i \gamma_{i2} \left(\ln(\pi_2) + \ln(\lambda e^{-\lambda x_i}) \right) \\ \Rightarrow \frac{\partial Q(\theta, \theta^{(t)})}{\partial \lambda} &\propto \frac{\partial}{\partial \lambda} \left(\sum_i \gamma_{i2} \ln(\lambda e^{-\lambda x_i}) \right) = 0 \\ &= \frac{\partial}{\partial \lambda} \left(\sum_i \gamma_{i2} (\ln \lambda - \lambda x_i) \right) \\ &= \sum_i \gamma_{i2} \left(\frac{1}{\lambda} - x_i \right) = 0 \\ \Rightarrow \lambda &= \frac{\sum_i \gamma_{i2}}{\sum_i \gamma_{i2} x_i}\end{aligned}\tag{67}$$

6 To Know Cold

1. Objective function for mean-squared error linear regression ($\mathbf{y} = \mathbf{X}\mathbf{w}$ where $\mathbf{X}\mathbf{w} \equiv \mathbf{w}^T \phi(\mathbf{x})$ and $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \dots, \phi_M(\mathbf{x})]^T$):

$$J(W) = \sum_{i=1}^N \frac{(t_i - y_i)^2}{N} \equiv |\mathbf{t} - \mathbf{X}\mathbf{w}|^2 \quad (68)$$

With optimal weights found by

$$\left. \frac{\partial J}{\partial W} \right|_{W^*} = 0 \quad (69)$$

and corresponding optimal solution

$$W^* = (X^T X)^{-1} X^T t \quad (70)$$

2. Derivatives acting on transposed variables:

$$\frac{\partial}{\partial W} \left(W^T X^T X W \right) = (X^T X W)^T + W^T X^T X = 2W^T X^T X \quad (71)$$

3. MLE Optimization Algorithm:

$$\left. \frac{\partial \mathcal{L}_{\text{MLE}}}{\partial \theta} \right|_{\theta^*} = 0 \quad \text{where} \quad \mathcal{L}_{\text{MLE}} = \prod_{i=1}^N P(x_i | \theta) \quad (72)$$

4. MAP Optimization Algorithm:

$$\left. \frac{\partial \mathcal{L}_{\text{MAP}}}{\partial \theta} \right|_{\theta^*} = 0 \quad \text{where} \quad \mathcal{L}_{\text{MAP}} = \left(\prod_{i=1}^N P(x_i | \theta) \right) P(\theta) \quad (73)$$

5. Product Rules

$$\begin{aligned} \prod_{i=1}^N p(1-p)^{x_i-1} &= p^N (1-p)^{\sum_i x_i - N} \\ \prod_{i=1}^N \lambda e^{-\lambda x_i} &= \lambda^N e^{-\lambda \sum_i x_i} \end{aligned} \quad (74)$$

6. 2x2 matrix inverse

$$A^{-1} = \frac{1}{|A|} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \quad (75)$$

where $|A| = ad - bc$

7. Mixtures of various distributions:

- Exponentials:

$$P(X|\theta) = \prod_{i=1}^N \sum_{k=1}^K \pi_k \lambda_k e^{-\lambda_k x_i} \quad (76)$$

- Gaussians:

$$P(X|\theta) = \prod_{i=1}^N \sum_{k=1}^K \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k) \quad (77)$$

- Bernoullis:

$$P(X|\theta) = \prod_{i=1}^N \sum_{k=1}^K \pi_k p_k^{x_i} (1 - p_k)^{1-x_i} \quad (78)$$

for binary data $x_i \in \{0, 1\}$

- Poissons:

$$\prod_{i=1}^N \sum_{k=1}^K \pi_k \frac{\lambda_k^{x_i} e^{-\lambda_k}}{x_i!} \quad (79)$$

for sequences $x_i \in \{0, 1, 2, \dots\}$

8. Naive Bayes Classifier (for two class classification):

$$g(X) = g_1(X) - g_0(X) \quad (80)$$

where

$$g_1(X) \propto \ln(P(X|C_1)) \neq \ln\left(\prod_{i=1}^N P(x_i|C_1)\right) \quad (81)$$

9. Second Parameter update example for $X = \{x_i\}$ and $N^{(t+1)} = |X^{(t+1)}| - |X^{(t)}|$

$$\theta_*^{(t+1)} = \frac{N^{(t+1)} + \alpha^{(t+1)}}{\sum_{i \in X^{(t+1)}/X^t} x_i + \beta^{(t+1)}} \quad (82)$$

10. 4 Methods for preventing overfitting:

(a) Regularization:

$$J'(W) = J(W) + \lambda\Omega(W) \quad (83)$$

where $\Omega(W)$ can be

i. L_2 /Ridge:

$$\Omega(W) = \sum_j w_j^2 \quad (84)$$

ii. L_1 /Lasso:

$$\Omega(W) = \sum_j |w_j| \quad (85)$$

iii. Elastic Net:

$$\Omega(W) = \alpha \sum_j |w_j| + (1 - \alpha) \sum_j w_j^2 \quad (86)$$

(b) Occam's Razor (choose simpler model $\equiv$ model with fewer necessary parameters)

(c) Cross-Validation (ex: K -fold)

(d) Addition of more data

11. Observed data likelihood optimization $\leftrightarrow$ objective function optimization

$$\arg \max_W \mathcal{L} = \arg \min_W J(W) \quad (87)$$

12. Conversion from $\arg \max_W \leftrightarrow \arg \min_W$:

$$\arg \max(-f(x)) = -\arg \max(f(x)) = \arg \min(f(x)) \quad (88)$$

13. Expectation Maximization Algorithm:

(a) Expectation (E) Step:

$$Q(\theta, \theta^{(t)}) = \mathbb{E}_Z \left[\ln \mathcal{L} \middle| X, \theta \right] \quad (89)$$

where

$$\mathcal{L} = \prod_{i=1}^N \prod_{k=1}^K \left(\pi_k P_k(x_i | \theta_k) \right)^{z_{ik}} \quad (90)$$

for example, for Gaussian data likelihoods, $P_k(x_i|\theta_k) = \mathcal{N}(x_i|\mu_k, \sigma_k)$.
This results in

$$Q(\theta, \theta^{(t)}) = \sum_{i=1}^N \sum_{k=1}^K \gamma_{ik} \left(\ln(\pi_k) + \ln \left(P_k(x_i|\theta_k) \right) \right) \quad (91)$$

where

$$\gamma_{ik}^{(t)} \equiv P_k(z_{ik} = 1|x_i, \theta^{(t)}) = \frac{P_k(x_i|\theta_k^{(t)})\pi_k^{(t)}}{\sum_{j=1}^K P_j(x_i|\theta_k^{(t)})\pi_j^{(t)}} \quad (92)$$

(b) Maximization (M) Step:

$$\theta^{(t+1)} = \arg \max_{\theta} Q(\theta, \theta^{(t)}) \equiv \left. \frac{\partial Q(\theta, \theta^{(t)})}{\partial \theta} \right|_{\theta=\theta^{(t+1)}} = 0 \quad (93)$$

14. Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (94)$$