Assignment 1

Aidan Fischer

2/11/2022

Homework assignments will be done individually: each student must hand in their own answers. Use of partial or entire solutions obtained from others or online is strictly prohibited. Electronic submission on Canvas is mandatory.

1. **Maximum Likelihood estimator** (10 points) Assuming data points are independent and identically distributed (i.i.d.), the probability of the data set given parameters: μ and σ^2 (the likelihood function):

$$P(\mathbf{x}|\mu,\sigma^2) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu,\sigma^2)$$

Please calculate the solution for μ and σ^2 using Maximum Likelihood (ML) estimator.

$log(P(\mathbf{x}|\mu,\sigma^2) = \sum_{n=1}^{N} \mathcal{N}(x_n|\mu,\sigma^2)$

$= -\frac{N}{2}log(\sigma^2) - \frac{N}{2}log(2\pi) - \frac{1}{\sigma^2}(\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2)$

MLE = -$\frac{N}{2}log(\sigma^2) - \frac{N}{2}log(2\pi) - \frac{1}{2}\frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2) = -\frac{N}{2}log(\sigma^2) - \frac{N}{2}log(2\pi) - \sum_{n=1}^{N} -\frac{1}{2}\frac{1}{\sigma^2}(x_n-\mu)^2)$

$\frac{\partial MLE}{\partial \mu} = \sum_{n=1}^{N}\frac{1}{\sigma^2}(x_n-\mu) = \frac{1}{\sigma^2}(\sum_{n=1}^{N} x_n) - N\frac{1}{\sigma^2}\mu$

$0 = \frac{1}{\sigma^2}(\sum_{n=1}^{N} x_n) - N\frac{1}{\sigma^2}\mu$

$N\frac{1}{\sigma^2}\mu = \frac{1}{\sigma^2}(\sum_{n=1}^{N} x_n)$

$N\mu = \sum_{n=1}^{N} x_n$

$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$

$\frac{\partial MLE}{\partial \sigma^2} = -\frac{N}{2\sigma^2} - (\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2)(-\frac{1}{(\sigma^2)^2})$

$= \frac{\partial MLE}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + (\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2)(\frac{1}{(\sigma^2)^2})$

$= \frac{\partial MLE}{\partial \sigma^2} = \frac{1}{2\sigma^2}(\frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - N)$

$0 = \frac{\partial MLE}{\partial \sigma^2} = \frac{1}{2\sigma^2}(\frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - N) \implies 0 = \frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - N$, if we assume $\sigma^2 \neq 0$

$N = \frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2$

$N\sigma^2 = \sum_{n=1}^{N}(x_n-\mu)^2$

$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)^2$

Final Solution: $\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$, $\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)^2$

2. **Maximum Likelihood** (10 points) We assume there is a true function $f(\mathbf{x})$ and the target value is given by $y = f(x) + \epsilon$ where ϵ is a Gaussian distribution with mean 0 and variance σ^2. Thus,

$$p(y|x, w, \beta) = \mathcal{N}(y|f(x), \beta^{-1})$$

where $\beta^{-1} = \sigma^2$.

Assuming the data points are drawn independently from the distribution, we obtain the likelihood function:

$$p(\mathbf{y}|\mathbf{x}, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(y_n|f(x), \beta^{-1})$$

Please show that maximizing the likelihood function is equivalent to minimizing the sum-of-squares error function.

From previous problem, this form of likelihood problem has a log-likelihood function of

-$\frac{N}{2}log(\sigma^2) - \frac{N}{2}log(2\pi) - \frac{1}{2}\frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n - \mu)^2)$

Since we know that $\beta^{-1} = \sigma^2$, y_n takes the role of the data point in this problem, and $f(x)$ takes the role of μ in this problem, we can replace the variables in the old expression to get:

$\frac{N}{2}log(\beta) - \frac{N}{2}log(2\pi) - \frac{1}{2}\beta\sum_{n=1}^{N}(y_n - f(x_n))^2)$

Since we only have control over f(x) through the w parameter, as β is fixed by the dataset, minimizing this expression is equivalent to minimizing this last term, in other words, minimizing the sum-of-squares loss function.

3. **MAP estimator** (15 points) Given input values $\mathbf{x} = (x_1, ..., x_N)^T$ and their corresponding target values $\mathbf{y} = (y_1, ..., y_N)^T$, we estimate the target by using function $f(x, \mathbf{w})$ which is a polynomial curve. Assuming the target variables are drawn from Gaussian distribution:

$$p(y|x, \mathbf{w}, \beta) = \mathcal{N}(y|f(x, \mathbf{w}), \beta^{-1})$$

and a prior Gaussian distribution for $\mathbf{w}$:

$$p(\mathbf{w}|\alpha) = (\frac{\alpha}{2\pi})^{(M+1)/2} \exp(-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w})$$

Please prove that maximum posterior (MAP) is equivalent to minimizing the regularized sum-of-squares error function. Note that the posterior distribution of $\mathbf{w}$ is $p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \alpha, \beta)$. **Hint: use Bayes' theorem.**

By Bayes' theorem, $p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \alpha, \beta) \propto p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$

We know both $p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \beta)$ and $p(\mathbf{w}|\alpha)$

$p(\mathbf{w}|\alpha)$ is given a closed form, but we must take the likelihood function for $p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \beta)$, which is equal to $\prod_{n=1}^{N} \mathcal{N}(y_n|f(x, w), \beta^{-1})$.

Taking the logarithm of the product, which is equivalent to adding the logarithms of each term, we get

$ln(p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)) = ln(p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \beta)) + ln(p(\mathbf{w}|\alpha))$

Since $ln(p(\mathbf{y}|\mathbf{x}, \mathbf{w}, \beta))$ is the log-likelihood function, we can copy the computations from the previous problem, and find it is $\frac{N}{2}ln(\beta) - \frac{N}{2}ln(2\pi) - \frac{1}{2}\beta \sum_{n=1}^{N}(y_n - f(x_n, w))^2)$

Now, the other term.

$ln(p(\mathbf{w}|\alpha)) = ln((\frac{\alpha}{2\pi})^{(M+1)/2} \exp(-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}))$

$= ln((\frac{\alpha}{2\pi})^{(M+1)/2}) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}$

$= \frac{M+1}{2}ln(\frac{\alpha}{2\pi}) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}$

$= \frac{M+1}{2}(ln(\alpha) - ln(2\pi)) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}$

Finally, adding the terms gives $\frac{N}{2}ln(\beta) - \frac{N}{2}ln(2\pi) - \frac{1}{2}\beta \sum_{n=1}^{N}(y_n - f(x_n, w))^2) + \frac{M+1}{2}(ln(\alpha) - ln(2\pi)) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}$

As in the previous problem, the only variable we have control over is w, our parameter vector for the estimation function f. Thus, maximizing this expression is the same as maximizing what we have control over. Taking out terms constant with respect to w, we get

$-\frac{1}{2}\beta \sum_{n=1}^{N}(y_n - f(x_n, w))^2) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}$

Maximizing an expression is equivalent to minimizing the negative of that expression, so now can minimize

$\frac{1}{2}\beta \sum_{n=1}^{N}(y_n - f(x_n, w))^2) + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}$

which is the regularized sum-of-squares error function, and we have shown what we want to show.

4. **Linear model** (20 points) Consider a linear model of the form:

$$f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^{D} w_i x_i$$

together with a sum-of-squares error/loss function of the form:

$$L_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2$$

Now suppose that Gaussian noise ϵ_i with zero mean and variance σ^2 is added independently to each of the input variables x_i. By making use of $\mathbb{E}[\epsilon_i] = 0$ and $\mathbb{E}[\epsilon_i \epsilon_j] = \delta_{ij}\sigma^2$ where $\delta_{ii} = 1$, show that minimizing L_D averaged over the noise distribution is equivalent to minimizing the sum-of-squares error for noise-free input variables with the addition of a weight-decay regularization term, in which the bias parameter w_0 is omitted from the regularizer.

Since ϵ_i is added independently to each x_i, then the loss function becomes

$L_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n + \epsilon_n, \mathbf{w}) - y_n\}^2$

Working this out...

$L_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n + \epsilon_n, \mathbf{w}) - y_n\}^2$

$= \frac{1}{2} \sum_{n=1}^{N} \{w_0 + \sum_{i=1}^{D} w_i(x_i + \epsilon_i) - y_n\}^2$

$= \frac{1}{2} \sum_{n=1}^{N} \{w_0 + \sum_{i=1}^{D} (w_i x_i + w_i \epsilon_i) - y_n\}^2$

$= \frac{1}{2} \sum_{n=1}^{N} \{w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} w_i \epsilon_i - y_n\}^2$

$= \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w}) + \sum_{i=1}^{D} w_i \epsilon_i - y_n\}^2$

Now, taking a small break from that calculation to multiply out the original loss function's square to get the terms in that, to make later calculations easier:

$L_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2$

$= \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w})^2 - 2y_n f(\mathbf{x}_n, \mathbf{w}) + y_n^2\}$

Returning to the original calculation, multiply out the square here as well, then recombine.

Let $E = \sum_{i=1}^{D} w_i \epsilon_i$

$\frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w}) + E - y_n\}^2$

$= \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w})^2 + 2Ef(\mathbf{x}_n, \mathbf{w}) - 2y_n f(\mathbf{x}_n, \mathbf{w}) + E^2 - 2y_n E + y_n^2\}$

Taking the original loss function terms back into a square:

$= \frac{1}{2} \sum_{n=1}^{N} \{(f(\mathbf{x}_n, \mathbf{w}) - y_n)^2 + 2Ef(\mathbf{x}_n, \mathbf{w}) + E^2 - 2y_n E\}$

$= \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2 + \frac{1}{2} \sum_{n=1}^{N} \{2Ef(\mathbf{x}_n, \mathbf{w}) + E^2 - 2y_n E\}$

Making use of $\mathbb{E}(\epsilon_i) = 0$, and that w_i is not a random variable, we can say

$\mathbb{E}(E) = \mathbb{E}(\sum_{i=1}^{D} w_i \epsilon_i) = \mathbb{E}(\sum_{i=1}^{D} w_i \epsilon_i) = \sum_{i=1}^{D} \mathbb{E}(w_i \epsilon_i) = \sum_{i=1}^{D} w_i \mathbb{E}(\epsilon_i) = \sum_{i=1}^{D} w_i \mathbb{E}(\epsilon_i) = \sum_{i=1}^{D} w_i 0 = 0$

Therefore, we can make a best guess for E, and set it to 0 in the loss function. However, since expectation of multiplication is not 0, we cannot assume E^2 is 0.

$\frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2 + \frac{1}{2} \sum_{n=1}^{N} \{2Ef(\mathbf{x}_n, \mathbf{w}) + E^2 - 2y_n E\}$

$\approx \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2 + \frac{1}{2} \sum_{n=1}^{N} \{E^2\}$

$= \frac{1}{2} \sum_{n=1}^{N} \{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2 + \frac{N}{2} E^2$

Now, we need to figure out what E^2 is.

$E^2 = (\sum_{i=1}^{D} w_i \epsilon_i)^2$

$= (w_1\epsilon_1 + w_2\epsilon_2 + ... + w_D\epsilon_D)^2$

$= \sum_{i=1}^{D}(w_i^2\epsilon_i^2) + \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} 2w_i\epsilon_i w_j\epsilon_j$ (from the form of $(a+b+c+...+n)^2$)

Now, we can make use of the last fact we know, $\mathbb{E}[\epsilon_i\epsilon_j] = \delta_{ij}\sigma^2$ where $\delta_{ii} = 1$, we can finish off the calculations for E^2 and return to the loss function.

$\sum_{i=1}^{D}(w_i^2\epsilon_i^2) + \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} 2w_i\epsilon_i w_j\epsilon_j$

$\approx \sum_{i=1}^{D}(\sigma^2 w_i^2) + \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} 2w_i\epsilon_i w_j\epsilon_j$ (from $\delta_{ii} = 1$)

$= \sigma^2 \sum_{i=1}^{D}(w_i^2) + \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} 2w_i\epsilon_i w_j\epsilon_j$

Note, here,
w does NOT contain w_0.

$= \sigma^2 \mathbf{w}^T\mathbf{w} + \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} 2w_i\epsilon_i w_j\epsilon_j$

Now, for the other sum, we can make use of the facts mentioned earlier still, however, we don't know the other δ_{ij}s. For now, let us just simplify through this.

$\sigma^2 \mathbf{w}^T\mathbf{w} + \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} 2w_i\epsilon_i w_j\epsilon_j$

$= \sigma^2 \mathbf{w}^T\mathbf{w} + \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} 2w_i w_j\epsilon_i\epsilon_j$

$\approx \sigma^2 \mathbf{w}^T\mathbf{w} + \sum_{i=1}^{D-1}\sum_{j=i+1}^{D} 2w_i w_j\delta_{ij}\sigma^2$

$= \sigma^2 \mathbf{w}^T\mathbf{w} + 2\sigma^2\sum_{i=1}^{D-1}\sum_{j=i+1}^{D} w_i w_j\delta_{ij}$

Since we don't know the values of δ_{ij} when $i \neq j$, but since $w_i, w_j, \delta_{ij}, \sigma$ are constants, we can collapse the whole sum into an arbitrary accumulation variable α

$= \sigma^2 \mathbf{w}^T\mathbf{w} + \alpha$

$= \sigma^2 \mathbf{w}^T\mathbf{w} + \alpha\frac{\mathbf{w}^T\mathbf{w}}{\mathbf{w}^T\mathbf{w}}$

$= \mathbf{w}^T\mathbf{w}(\sigma^2 + \frac{\alpha}{\mathbf{w}^T\mathbf{w}})$

Collapsing the constants:

$= \alpha\mathbf{w}^T\mathbf{w}$

Returning back to original computation:

$\frac{1}{2}\sum_{n=1}^{N}\{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2 + \frac{N}{2}E^2$

$= \frac{1}{2}\sum_{n=1}^{N}\{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2 + \frac{N}{2}\alpha\mathbf{w}^T\mathbf{w}$

Collapsing constants one last time

$= \frac{1}{2}\sum_{n=1}^{N}\{f(\mathbf{x}_n, \mathbf{w}) - y_n\}^2 + \alpha\mathbf{w}^T\mathbf{w}$

Remember that $\mathbf{w}$ here is without w_0 here because the squaring did not include w_0, thus we have our answer. Since we can't know the value of α, it can be chosen arbitrary depending on the regularization.

5. **Linear regression** (45 points) Please choose **one** of the below problems. You will need to **submit your code**.

   a) UCI Machine Learning: Facebook Comment Volume Data Set

   Please implement a Ridge regression model and use mini-batch gradient descent to train the model on this dataset for predicting the number of comments in next H hrs (H is given in the feature). You do not need to use all the features. Use K-fold cross validation and report the mean squared error (MSE) on the testing data. You need to write down every step in your experiment.

   a) UCI Machine Learning: Bike Sharing Data Set

   Please write a Ridge regression model and use mini-batch gradient descent to train the model on this dataset for predicting the count of total rental bikes including both casual and registered. You do not need to use all the features. Use K-fold cross validation and report the mean squared error (MSE) on the testing data. You need to write down every step in your experiment.

   Choosing problem a. First thing that I needed to do was figure out what the different variants meant. From looking at the linked research paper, they just represent datasets taken starting from a different number of hours after a training post had been posted. For example, variant 1 is taken from 6 hours after a post, and variant 5 was 64 hours.

   Next, choosing which features to use from the training data. I decided to exclude features 5-29, because they are derived from other features, and feature 4, because it encodes a value. I am also excluding feature 34 because it is the difference between 2 prior including features. Finally, I excluded feature 54 as it was a target variable. The specific names of the features I am including are listed in the code file Dataset.py in a comment.

   Step 1: Read in the dataset. Reading in the full dataset takes place in get_full_dataset, which obtains the full training dataset

   Step 2: Split into 10 subsets. The dataset is split into K evenly sized sets

   Step 3: For each set, select that set as validation set and compile the other 9 sets into a training set.

   Step 4: Run minibatch gradient descent on the training set, using Adagrad dynamic learning rate.

   Step 5: Get average squared error from the trained model on the validation set and store it. Repeat steps 3 through 5 on the rest of the sets.

   Step 6: Get set with least error, return it.

   Step 7: Calculate MSE on full data set, using best model, print it.

Please follow the below instructions when you submit the assignment.

1. You are NOT allowed to use packages for implementing the code required in this assignment.
2. Your submission should consist of a zip file named Assignment1_LastName_FirstName.zip which contains:
   - a jupyter notebook file(.ipynb) or a python file (.py). The file should contain the code and the output after execution (in comments if you use python). You should also include detailed comments.
   - a pdf file to show (1) the derivation steps of for questions 1 to 4 and (2) experiment design and results (plots, tables, etc) for question 5.