CSE 6740: Homework 1

Mingzheng Michael Huang

September 14, 2023

Contents

1 Linear Regression [30 pts]

1.1 Derivation of $E[\hat{w}]$ [6 pts]

Step 1: Starting with the normal equation, we have

$$\hat{w} = (X^T X)^{-1} X^T Y$$

Step 2: Taking the expectation on both sides

$$E[\hat{w}] = E\left[(X^T X)^{-1} X^T Y\right]$$

Step 3: Since X is fixed, it can be moved out of the expectation

$$E[\hat{w}] = (X^T X)^{-1} X^T E[Y]$$

We also know that $Y = Xw + \epsilon$, so $E[Y] = E[Xw + \epsilon] = Xw$:

$$E[\hat{w}] = (X^T X)^{-1} X^T X w$$

Simplifying, we get:

$$E[\hat{w}] = w$$

Thus, $\hat{w}$ is an unbiased estimator for w.

1.2 Derivation of Var$[\hat{w}]$ [6 pts]

The initial expression for the variance of $\hat{w}$ is:

$$\text{Var}[\hat{w}] = (X^T X)^{-1} X^T (\sigma^2 I) X (X^T X)^{-1}$$

Step 1: Use the Identity Matrix

The term $\sigma^2 I$ is essentially scaling the identity matrix by σ^2. Multiplying any matrix by the identity matrix leaves it unchanged. Thus, when we multiply $X^T(\sigma^2 I)X$, it's equivalent to scaling $X^T X$ by σ^2.

Step 2: Simplify $\sigma^2 I$

Using the above reasoning, $X^T(\sigma^2 I)X = \sigma^2 X^T X$.

Step 3: Substitute into the Original Equation

Substituting this simplified term back into the original equation gives:

$$\mathrm{Var}[\hat{w}] = (X^T X)^{-1}\sigma^2 X^T X(X^T X)^{-1}$$

Step 4: Cancel Out $X^T X$ Terms

Notice that $(X^T X)^{-1}X^T X(X^T X)^{-1}$ contains $(X^T X)^{-1}X^T X$, which will cancel out, leaving:

$$\mathrm{Var}[\hat{w}] = \sigma^2(X^T X)^{-1}$$

1.3 Gaussian Distribution of $\hat{w}$ [8 pts]

Step 1: Expression for $\hat{w}$

We start with the estimate $\hat{w}$ in linear regression, which is given by:

$$\hat{w} = (X^T X)^{-1}X^T Y$$

Additionally, we have the true relationship between Y and X:

$$Y = Xw + \epsilon$$

Here, ϵ is the noise term, Gaussian distributed with zero mean and constant variance σ^2.

Step 2: Substitute Y into $\hat{w}$

Next, we substitute Y into the equation for $\hat{w}$:

$$\hat{w} = (X^T X)^{-1}X^T(Xw + \epsilon)$$

Simplifying, we get:

$$\hat{w} = w + (X^T X)^{-1}X^T\epsilon$$

Step 3: Distribution of $\hat{w}$

1. **Linearity of Gaussian Distribution**: A linear combination of Gaussian variables is also Gaussian.

2. ϵ **is Gaussian**: ϵ follows a Gaussian distribution.

Thus, $\hat{w}$, being a linear combination of w (a constant) and ϵ (a Gaussian variable), must also be Gaussian.

Step 4: Mean and Variance of $\hat{w}$

We have derived that the expected value $E[\hat{w}]$ is w. We have also derived that $\text{Var}[\hat{w}] = \sigma^2(X^TX)^{-1}$.

So, under the white noise assumption, which stipulates that ϵ is Gaussian-distributed with zero mean and constant variance σ^2, $\hat{w}$ also follows a Gaussian distribution. Its mean is w and its variance is $\sigma^2(X^TX)^{-1}$.

1.4 Weighted Linear Regression [10 pts]

In weighted linear regression, the objective function $J(w)$ to be minimized is given by:

$$J(w) = (Y - Xw)^T\Sigma^{-1}(Y - Xw)$$

Step 1: Taking the Derivative

To find the value of w that minimizes $J(w)$, we take the derivative of $J(w)$ with respect to w and set it to zero. Mathematically, this is expressed as:

$$\frac{\partial J(w)}{\partial w} = 0$$

First, let's expand the objective function:

$$J(w) = Y^T\Sigma^{-1}Y - Y^T\Sigma^{-1}Xw - w^TX^T\Sigma^{-1}Y + w^TX^T\Sigma^{-1}Xw$$

Now, taking the derivative of $J(w)$ with respect to w yields:

$$\frac{\partial J(w)}{\partial w} = -2X^T\Sigma^{-1}Y + 2X^T\Sigma^{-1}Xw$$

Step 2: Finding the Estimator $\hat{w}$

Setting the derivative to zero gives:

$$-2X^T\Sigma^{-1}Y + 2X^T\Sigma^{-1}X\hat{w} = 0$$

Simplifying, we obtain:

$$X^T\Sigma^{-1}Y = X^T\Sigma^{-1}X\hat{w}$$

$$\Rightarrow \hat{w} = (X^T\Sigma^{-1}X)^{-1}X^T\Sigma^{-1}Y$$

This is the estimator $\hat{w}$ in weighted linear regression, accounting for the different variances σ_i^2 of each data point.

2 Ridge Regression [15 pts]

Step 1: Ridge Regression Estimate

The ridge regression estimate $\hat{\mathbf{w}}_{\text{ridge}}$ is:

$$\hat{\mathbf{w}}_{\text{ridge}} = (X^\top X + \lambda I)^{-1}X^\top Y$$

Step 2: Bayesian Posterior Mean Estimate

In a Bayesian framework, the mean of the posterior $\hat{\mathbf{w}}_{\text{posterior}}$ is:

$$\hat{\mathbf{w}}_{\text{posterior}} = (\tau^{-2}X^\top X + \sigma^{-2}I)^{-1}\tau^{-2}X^\top Y$$

Step 3: Comparison and Relationship

By setting λ such that $\lambda = \frac{\sigma^2}{\tau^2}$, we can establish:

$$\hat{\mathbf{w}}_{\text{ridge}} = \hat{\mathbf{w}}_{\text{posterior}}$$

Thus, the ridge regression estimate is the posterior distribution's mean under the Gaussian prior.

Step 4: Explicit Relation between λ, σ^2, and τ^2

The explicit relationship is:

$$\lambda = \frac{\sigma^2}{\tau^2}$$

3 Lasso Estimator

3.1 Equivalent Quadratic Function

The LASSO regression problem can be formulated as

$$\min_{w\in\mathbb{R}^d} \sum_{i=1}^{m} (w^\top x_i - y_i)^2 \quad \text{subject to } \|w\|_1 \leq \lambda.$$

A transformation involves representing the original w as $w^+ - w^-$, where w^+ and w^- are the positive and negative components of w, respectively.

The new, equivalent function $Q(w^+, w^-)$ is

$$Q(w^+, w^-) = \sum_{i=1}^{m} ((w^+ - w^-)^\top x_i - y_i)^2 + \lambda \left(\|w^+\|_1 + \|w^-\|_1\right).$$

3.2 Stochastic Gradient Descent Algorithm

To calculate the gradient for a single data point (x_i, y_i), we have:

$$\begin{aligned}
\frac{\partial Q}{\partial w_j^+} &= 2((w^+ - w^-)^\top x_i - y_i) \cdot x_{ij} + \lambda, \\
\frac{\partial Q}{\partial w_j^-} &= -2((w^+ - w^-)^\top x_i - y_i) \cdot x_{ij} + \lambda.
\end{aligned}$$

After obtaining the gradient, the update rules for w_j^+ and w_j^- are:

$$\begin{aligned}
w^+_{\text{new}} &= w^+_{\text{old}} - \alpha \frac{\partial Q}{\partial w_j^+}, \\
w^-_{\text{new}} &= w^-_{\text{old}} - \alpha \frac{\partial Q}{\partial w_j^-}.
\end{aligned}$$

3.3 Orthonormal X and SGD [10 pts]

Explanation

First, while the orthonormality simplifies the optimization problem, it doesn't ensure that SGD will find the sparse solution w with k non-zero elements. Second, factors like learning rate and initial conditions play a role in the convergence of SGD, but they don't necessarily guide it toward a sparse solution.

Furthermore, the inherent randomness in stochastic methods like SGD introduces an additional layer of uncertainty in finding the exact sparse solution. While LASSO's ℓ_1-norm constraint does promote sparsity, the primary aim of SGD is to minimize the loss function. This could theoretically allow it to converge to w, but there's no practical guarantee.

In summary, despite the favorable conditions and the sparsity-promoting ℓ_1-norm, SGD is not tailored to find sparse solutions. Therefore, there's no guarantee it will converge arbitrarily close to w.

4 Logistic Regression

4.1 Log-Odds as a Linear Function of X [6 pts]

The log-odds of success, or the logit, is defined as:

$$\ln\left(\frac{P(Y=1|X=x)}{P(Y=0|X=x)}\right)$$

Given that

$$P(Y=1|X=x) = \frac{\exp(w_0 + w^T x)}{1 + \exp(w_0 + w^T x)}$$

and

$$P(Y=0|X=x) = 1 - P(Y=1|X=x) = \frac{1}{1 + \exp(w_0 + w^T x)}$$

Substituting these into the log-odds equation yields:

$$\ln\left(\frac{\exp(w_0 + w^T x)}{1 + \exp(w_0 + w^T x)} \times \frac{1 + \exp(w_0 + w^T x)}{1}\right)$$

Simplifying, we get:

$$\ln(\exp(w_0 + w^T x)) = w_0 + w^T x$$

Thus, the log-odds of success is a linear function of X.

4.2 Convexity of Logistic Loss [9 pts]

To show that the logistic loss function $L(z) = \log(1 + \exp(-z))$ is convex, we take its second derivative.

First derivative $L'(z)$:

$$L'(z) = \frac{-\exp(-z)}{1 + \exp(-z)}$$

Second derivative $L''(z)$:

$$L''(z) = \frac{\exp(-z)}{(1 + \exp(-z))^2}$$

Since $L''(z)$ is always positive, $L(z)$ is a convex function.

5 Programming: Recommendation System [40 pts]

5.1 Derivation of Update Formula [10 pts]

Partial Derivatives

To minimize $E(U, V)$, we employ the gradient descent optimization technique, which necessitates the computation of partial derivatives of $E(U, V)$ with respect to each element in U and V.

Partial Derivative with respect to $U_{v,k}$

The partial derivative of $E(U, V)$ concerning $U_{v,k}$ can be computed as follows:

$$\frac{\partial E(U, V)}{\partial U_{v,k}} = -2 \sum_{i=1}^{m} (M_{v,i} - U_v V_i^\top) V_{i,k}$$

Partial Derivative with respect to $V_{j,k}$

Similarly, the partial derivative of $E(U, V)$ concerning $V_{j,k}$ is:

$$\frac{\partial E(U, V)}{\partial V_{j,k}} = -2 \sum_{u=1}^{n} (M_{u,j} - U_u V_j^\top) U_{u,k}$$

Update Rule for U

$$U_{v,k} \leftarrow U_{v,k} + 2\mu \sum_{i=1}^{m} (M_{v,i} - U_v V_i^\top) V_{i,k}$$

Update Rule for V

$$V_{j,k} \leftarrow V_{j,k} + 2\mu \sum_{u=1}^{n} (M_{u,j} - U_u V_j^\top) U_{u,k}$$

where μ is the learning rate, a hyperparameter that governs the magnitude of each update step.

5.2 Regularized Objective Function [5 pts]

To improve the robustness of the matrix-factorization model against overfitting, we introduce regularization terms into the objective function $E(U, V)$. The new regularized objective function is as follows:

$$E(U,V) = \sum_{u=1}^{n}\sum_{i=1}^{m}\left(M_{u,i} - \sum_{k=1}^{d} U_{u,k}V_{i,k}\right)^2 + \lambda\sum_{u,k} U_{u,k}^2 + \lambda\sum_{i,k} V_{i,k}^2$$

Derivation of Partial Derivatives

To minimize this objective function, we need to compute the gradient with respect to each element in the matrices U and V. The partial derivatives with respect to $U_{v,k}$ and $V_{j,k}$ are derived as follows:

Partial Derivative with respect to $U_{v,k}$

$$\frac{\partial E(U,V)}{\partial U_{v,k}} = -2\sum_{i=1}^{m}\left(M_{v,i} - \sum_{k=1}^{d} U_{v,k}V_{i,k}\right)V_{i,k} + 2\mu U_{v,k}$$

Partial Derivative with respect to $V_{j,k}$

$$\frac{\partial E(U,V)}{\partial V_{j,k}} = -2\sum_{u=1}^{n}\left(M_{u,j} - \sum_{k=1}^{d} U_{u,k}V_{j,k}\right)U_{u,k} + 2\lambda V_{j,k}$$

Update Rules via Gradient Descent

Utilizing these partial derivatives, the update rules for U and V using gradient descent with step size μ are as follows:

Update Rule for U

$$U_{v,k} \leftarrow U_{v,k} - \mu\left(-2\sum_{i=1}^{m}\left(M_{v,i} - \sum_{k=1}^{d} U_{v,k}V_{i,k}\right)V_{i,k} + 2\lambda U_{v,k}\right)$$

Update Rule for V

$$V_{j,k} \leftarrow V_{j,k} - \mu \left(-2 \sum_{u=1}^{n} \left(M_{u,j} - \sum_{k=1}^{d} U_{u,k} V_{j,k} \right) U_{u,k} + 2\lambda V_{j,k} \right)$$

These update rules incorporate ℓ_2 regularization to mitigate the risk of overfitting by penalizing large values in U and V.

5.3 Python Implementation and Evaluation [15 pts]

```python
import numpy as np

def my_recommender(rate_mat, lr, with_reg):

    max_iter = 250
    learning_rate = 0.0001
    reg_coef = 2 if with_reg else 0

    n_user, n_item = rate_mat.shape

    U = np.random.rand(n_user, lr)
    V = np.random.rand(n_item, lr)

    mask = np.ma.masked_where(rate_mat != 0, rate_mat).mask

    for i in range(max_iter):
        U -= learning_rate * 2 * np.matmul(
            np.multiply(np.matmul(U, np.transpose(V)) - rate_mat, mask),
        V -= learning_rate * 2 * np.matmul(
            np.multiply(np.transpose(np.matmul(U, np.transpose(V)) - rate

        U -= learning_rate * 2 * reg_coef * U
        V -= learning_rate * 2 * reg_coef * V

    return U, V
```

5.4 Report [10 pts]

5.4.1 Performance (RMSE) on Training and Test Sets with Varied Low Rank

Method	Low Rank	Training RMSE	Test RMSE	Time (s)
SVD-noReg	1	0.9319	0.9609	14.28
SVD-withReg	1	0.9368	0.9657	13.05
SVD-noReg	3	0.9175	0.9556	18.32
SVD-withReg	3	0.9231	0.9596	19.60
SVD-noReg	5	0.9072	0.9536	22.44
SVD-withReg	5	0.9127	0.9558	25.66
SVD-noReg	7	0.9003	0.9564	21.48
SVD-withReg	7	0.9046	0.9563	16.13
SVD-noReg	9	0.8942	0.9569	15.96
SVD-withReg	9	0.8968	0.9548	20.76

5.4.2 Observations with Varied Low Rank

An evident trend is observed where the RMSE decreases as the `lowRank` value increases, indicating improved model performance. However, the marginal gains in RMSE reduction appear to taper off as the rank grows, suggesting a point of diminishing returns beyond which increasing the rank doesn't contribute significantly to performance improvement.

5.4.3 Hyperparameter Selection

- Learning Rate (μ): Chosen as 0.0001 after experimenting with multiple values to ensure quick convergence without oscillations.
- Regularization Coefficient (λ): Set to 2 when regularization is applied. This value was empirically determined to safeguard against overfitting without compromising the predictive accuracy.