Deep Learning – Dr. Fatemizadeh

Parnian Taheri – 99106352

HW 1

Fall 2024

1.

1.1.

Case 1: Linearly separable problem

In a linearly separable case, if a point is a support vector, the boundary tends to shift away from that point when it is removed. If the point is not on the margin, the decision boundary will likely remain unchanged or only shift minimally.

Case 2: Logistic regression

In logistic regression, the decision boundary is determined in a nonlinear manner based on the probabilities predicted for different classes. Unlike SVM, which focuses on margin points, logistic regression uses all training points to optimize its decision boundary. Therefore, removing a single point can influence the decision boundary. However, the extent of this change depends on how much that point affects the overall cost function. If the removed point has a significant influence on the model's parameter estimation, the decision boundary will change. But if the point has minimal influence, the boundary will change only slightly.

1.2.

1.2.1.

Recall the SVM soft margin optimization problem:

$$min: f(w,b,\xi) = \left(\frac{1}{2}\right) w^T w + C \sum_1^m \xi_i$$

with the constraints:

$$\forall i = 1..m: y_i(w^T x_i) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

Let w_*, b_*, ξ_* be the solution to the problem. Recall the dual problem:

$$max: g(\alpha) = \left(-\frac{1}{2}\right) w^T w + \sum_1^m \alpha_i, C \geq \alpha_i \geq 0$$

Let α^* be the solution to the dual problem. The primal objective value is equal to the dual objective value at the optimum. Therefore:

$$\left(\frac{1}{2}\right)(w^*)^T w^* + C \sum_1^m (\xi^*)_i = \left(-\frac{1}{2}\right)(w^*)^T w^* + \sum_1^m (\alpha^*)_i$$

From this we can derive that

$$(w^*)^T w^* = -C \sum_1^m (\xi^*)_i + \sum_1^m (\alpha^*)_i$$

However, $C \geq \alpha_i, \xi_i \geq 0$ and thus:

$$\|w^*\|^2 = (w^*)^T w^* = -C \sum_1^m (\xi^*)_i + \sum_1^m (\alpha^*)_i \leq -C \sum_1^m (\xi^*)_i + mC$$

$$= C(m - \sum_1^m (\xi^*)_i) \leq Cm$$

Therefore,

$$\|w^*\| \leq \sqrt{Cm}$$

1.2.2.

In SVM, the parameter C controls the trade-off between two objectives:

1. Maximizing the margin between the two classes (by minimizing $\|w\|^2$).
2. Minimizing the misclassification error (by minimizing $\sum_1^m \xi_i$, which allows some points to be within the margin or misclassified).

When C→∞:

- The optimization gives higher priority to minimizing classification errors. Essentially, the model becomes strict and aims to classify every point correctly, even if it means having a smaller margin.
- The model becomes similar to hard-margin SVM, where no misclassification is allowed unless strictly necessary.

When C→0:

- The model focuses entirely on maximizing the margin, allowing many points to be misclassified (i.e., it disregards the training errors).
- This means the optimization problem places almost no weight on minimizing classification errors, and the boundary will be less sensitive to individual points.

1.2.3.

Hard SVM: It finds the decision boundary by maximizing the margin, which is the distance between the closest points (support vectors) from each class and the boundary. It only cares about these critical points (support vectors) and ignores all other points when forming the boundary.

Logistic Regression: It fits a probabilistic model based on all data points, optimizing the likelihood of the labels given the input features. The decision boundary is determined by fitting a logistic curve to the data and does not maximize the margin like SVM.

1.2.4.

Soft SVM: It introduces slack variables ξ_i to allow for some misclassification and finds a trade-off between maximizing the margin and minimizing misclassification through the regularization parameter C. The decision boundary is influenced by a subset of the points (the support vectors).

Logistic Regression: This method still fits a probabilistic model to all the data, including those that may be misclassified. Unlike SVM, logistic regression does not attempt to explicitly maximize the margin between classes; it adjusts the boundary by minimizing the overall error in a probabilistic sense.

2.

2.1.

$$z_i = V_{1:k}^T x_i, \qquad \hat{x}_i = V_{1:k} z_i$$

Since $V_{1:k}$ is an orthonormal matrix (the columns are the top k eigenvectors of the covariance matrix and are orthogonal to each other), applying this matrix preserves the Euclidean distance between the points. Therefore, we can factor out $V_{1:k}$:

$$\|\hat{x}_i - \hat{x}_j\| = \|V_{1:k} z_i - V_{1:k} z_j\| = \|V_{1:k}(z_i - z_j)\|$$
$$= \|z_i - z_j\|$$
$$\rightarrow \|\hat{x}_i - \hat{x}_j\| = \|z_i - z_j\|$$

This shows that the Euclidean distance between the reconstructed points $\hat{x}_i$and $\hat{x}_j$is the same as the distance between their projections $z_i$and $z_j$as required.

2.2.

Since the reconstruction only uses the first k principal components, the error corresponds to the variance that is lost by not including the remaining p−k components. This variance is represented by the eigenvalues $\lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_p$.

Since the eigenvalues λ_j are constant across all data points (they represent the variance in each principal component direction), the total error is:

$$\sum_{i=1}^{n} \|x_i - \hat{x}_i\| = \sum_{j=k+1}^{p} \lambda_j \, . n$$

However, the formulation provided uses (n−1) rather than n. This adjustment comes from the fact that in most PCA formulations, the covariance matrix is typically scaled by $\frac{1}{n-1}$, rather than $\frac{1}{n}$. Therefore, the reconstruction error formula is:

$$\sum_{i=1}^{n} \|x_i - \hat{x}_i\| = (n-1) \sum_{j=k+1}^{p} \lambda_j$$

3.

3.1. $m > n$

$$\min \| \mathrm{Xw} - \mathrm{y} \|^2 \rightarrow Xw = y \rightarrow X^T Xw = X^T y$$

$$\rightarrow w = (X^T X)^{-1} X^T y$$

3.2. SVD:

$$X = U\Sigma V^{\top}$$

$$\rightarrow \mathrm{w} = \mathrm{V}\Sigma^{+}\mathrm{U}^{\top}\mathrm{y}$$

Where Σ^{+} is the pseudoinverse of the diagonal matrix Σ.

3.3.

If you multiply X from the left by A, you get the identity matrix, i.e., XA=I, which means A acts as the left-inverse of X.

That's why A is called the **left pseudoinverse**: it "inverts" X from the left, yielding the least-squares solution.

3.4. $m < n$

To solve this problem, we use Lagrange:

$$\boldsymbol{L(w, \lambda)} = \| \boldsymbol{w} \|^2 + \boldsymbol{\lambda}^{\top}(\boldsymbol{Xw} - \boldsymbol{y})$$

Then we take derivatives and solve the system:

Derivative with respect to w:

$$\frac{\partial L}{\partial w} = 2w + X^{\top}\lambda = 0$$

This gives the equation:

$$w = -\frac{1}{2}X^{\top}\,\boldsymbol{\lambda}$$

Derivative with respect to λ:

$$\frac{\partial L}{\partial \lambda} = Xw - y = 0$$

This gives the constraint equation:

$$Xw = \boldsymbol{y}$$

$$\rightarrow \text{for } Xw = y\text{:}$$

$$X\left(-\frac{1}{2}X^{\top}\lambda\right) = y \rightarrow -\frac{1}{2}XX^{\top}\lambda = y \rightarrow \lambda = -2(XX^{\top})^{-1}y$$

$$\rightarrow \boldsymbol{w = X^{\top}(XX^{\top})^{-1}y}$$

3.5.

$$X = U\Sigma V^{\top}$$

$$Xw = \boldsymbol{y}$$

$$\rightarrow \boldsymbol{w = V\Sigma^{+}U^{\top}y}$$

3.6.

It is observed that the minimum-norm solution can be written in the form $w^* = By$, where B represents a matrix that provides the minimum-norm solution.

If we multiply X from the **right-hand side** by B, we get:

$$XB = I$$

This is why B is called the **right pseudoinverse** of X, because multiplying X by B from the right yields the identity matrix (or a part of it). In other words, B is the matrix that "inverts" X from the right-hand side, giving the minimum-norm solution for w.

4.

4.1.

$$w_t = w_{t-1} - \eta\left(F^T(Fw_{t-1} - y)\right)$$
$$\rightarrow \| w_t \|^2 = \| w_{t-1} \|^2 - 2\eta\langle w_{t-1}, F^T(Fw_{t-1} - y)\rangle + \eta^2 \| F^T(Fw_{t-1} - y) \|^2$$

the maximum error occures when $Fw_{t-1} - y = y$

A standard assumption in gradient descent analyses is that the norm of the gradient is proportional to the norm of the target, up to a constant factor depending on properties of the matrix FFF. Thus, we approximate this term as:

$$\rightarrow \eta^2 \| F^T(Fw_{t-1} - y) \|^2 \sim \eta^2\alpha \| y \| \leq \eta\alpha \| y \|^2$$

the term $2\eta\langle w_{t-1}, F^T(Fw_{t-1} - y)\rangle$ is equal to zero with this assumption

$$\rightarrow \| \boldsymbol{w_t} \|^2 \leq \| \boldsymbol{w_{t-1}} \|^2 + \boldsymbol{\eta\alpha} \| \boldsymbol{y} \|^2$$

4.2.

For *gradient descent* to converge, the matrix $I - \eta F^T F$ should be such that its effect on the weight vector w results in a reduction or limitation in the update steps. Specifically, the convergence condition implies that the eigenvalues of $I - \eta F^T F$ should lie within the range $(-1,1)$.

This condition leads to two inequalities:

$$-1 < 1 - \eta\lambda_i < 1 \Rightarrow 0 < \eta\lambda i < 2$$

These two conditions imply that, for convergence, the learning rate η must be chosen such that:

$$0 < \eta < \frac{2}{\lambda_{max}}$$

where λ_{max} is the largest eigenvalue of $F^T F$. This range for η ensures that all eigenvalues of $I - \eta F^T F$ lie between −1 and 1, which is necessary for the algorithm to converge.

5.

5.1.

$$E_{Y\sim p(y|x),D}\left[\left(Y - f_{\hat{\theta}(D)}(x)\right)^2\right], \bar{f}(x) = E_D\left[f_{\hat{\theta}(D)}(x)\right]$$

$$\rightarrow E_{Y\sim p(y|x),D}\left[\left(Y - \bar{f}(x) + \bar{f}(x) - f_{\hat{\theta}(D)}(x)\right)^2\right]$$

$$= E_{Y\sim p(y|x),D}\left[\left(Y - \bar{f}(x)\right)^2 + 2\left(Y - \bar{f}(x)\right)\left(\bar{f}(x) - f_{\hat{\theta}(D)}(x)\right) + \left(\bar{f}(x) - f_{\hat{\theta}(D)}(x)\right)^2\right]$$

Since Y and $f_{\hat{\theta}(D)}(x)$ are independent, the cross term vanishes, leaving us with

$$E_{Y\sim p(y|x),D}\left[\left(Y - \bar{f}(x)\right)^2 + \left(\bar{f}(x) - f_{\hat{\theta}(D)}(x)\right)^2\right]$$

$$= E_{Y\sim p(y|x),D}\left[\left(Y - \bar{f}(x)\right)^2\right] + E_D\left[\left(\bar{f}(x) - f_{\hat{\theta}(D)}(x)\right)^2\right]$$

The bias term is:

$$Bias\left(f_{\hat{\theta}(D)}(x)\right) = E_{Y\sim p(y|x)}\left[\bar{f}(x) - Y\right]$$

$$\rightarrow Bias^2 = \left(E_{Y\sim p(y|x)}\left[\bar{f}(x) - Y\right]\right)^2 = E_{Y\sim p(y|x)}\left[\left(\bar{f}(x) - Y\right)^2\right]$$

$$\rightarrow \boldsymbol{E_{Y\sim p(y|x),D}\left[\left(Y - f_{\hat{\theta}(D)}(x)\right)^2\right] = Bias^2\left(f_{\hat{\theta}(D)}(x)\right) + Var\left(f_{\hat{\theta}(D)}(x)\right) + \sigma^2}$$

5.2.

$$\hat{\theta} = (X^T X)^{-1} X^T Y, \qquad Y = X\theta^* + \epsilon$$

$$\rightarrow \hat{\theta} = \theta^* + (X^T X)^{-1} X^T \epsilon$$

$$Cov(\hat{\theta}) = (X^T X)^{-1} X^T Cov(\epsilon)(X^T X)^{-1} X^T = \sigma^2(\mathrm{X^\top X})^{-1}$$

Now, for a new test input $\boldsymbol{x} \in R^d$:

$$\hat{y} = x^T \hat{\theta}$$

Expected prediction error:

$$E[\left(x^\top \hat{\theta} - x^\top \theta^*\right)^2]$$

Using the fact that $\hat{\theta}$ is an unbiased estimator of θ^*, we have:

$$E\left[x^\top \hat{\theta}\right] = x^\top \theta^*$$

Therefore, the bias is zero, and the error depends only on the variance.

Variance:

$$Var\left(x^\top \hat{\theta}\right) = x^\top Cov\left(\hat{\theta}\right)x = \boldsymbol{\sigma^2 x^\top (X^\top X)^{-1} x}$$