hw_4

Matthew Jensen

2024-11-16

Part 1

Problem 1

a.

When a point breaks into the margin, it means that it is either found on the hyperplane or is misclassified. ξ is the measure of how far a point x_i has broken into the margin. If the point is on the margin but still correctly classified, then ξ is greater than 0, but still less than 1. If the point is misclassified (on the wrong side of the hyperplane), then ξ will be greater than 1.

Therefore, when a training example breaks into the margin:

- $0 < \xi_i \leq 1$ if the example lies within the margin but is correctly classified.
- $\xi_i > 1$ if the example is misclassified.

b.

When the training example x_i stays on or outside the margin, it means that the example either lies correctly outside the margin or exactly on the margin, satisfying the constraint $y_i(w^T x_i + b) \geq 1 - \xi_i$. In this case, the slack variable ξ_i will be 0 because the example does not violate the margin. This means the training example is correctly classified and does not contribute any slack.

Therefore, when a training example stays on or outside the margin,

- $\xi_i = 0$, indicating that no slack is required and the example adheres to the margin constraint.

c.

The term $C \sum_i \xi_i$ is incorporated into the objective function to penalize slack variables, which represent violations of the margin. This term helps to control the trade-off between maximizing the margin (minimizing $\frac{1}{2} w^T w$) and allowing some training examples to break into the margin, which is necessary when the data is not linearly separable. The parameter C serves as a regularization factor, adjusting the importance of penalizing margin violations. A larger value of C increases the penalty for margin violations, encouraging the model to prioritize a larger margin with fewer violations. Conversely, a smaller C allows more margin violations, which may result in better generalization for noisy data.

If we were to remove the term $C \sum_i \xi_i$ from the objective function, the optimization problem would no longer penalize margin violations. As a result, the model would focus solely on maximizing the margin (minimizing $\frac{1}{2} w^T w$) without accounting for the misclassifications or margin violations. This could lead to an overly strict margin, especially when the data is not linearly separable, and could result in a model that is unable to generalize well to unseen data, potentially causing poor performance on noisy or complex datasets.

Problem 2

Step i: Primal Problem Formulation

The primal optimization problem for the soft margin SVM is formulated as follows:

$$\min_{w,b,\{\xi_i\}} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$$

subject to the constraints:

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i = 1, 2, \ldots, N$$

$$\xi_i \geq 0 \quad \forall i = 1, 2, \ldots, N$$

Step ii: Lagrange Multiplier Method

To derive the dual, we first introduce Lagrange multipliers for the constraints. We define the Lagrange multiplier $\alpha_i \geq 0$ for each constraint $y_i(w^T x_i + b) \geq 1 - \xi_i$, and the Lagrange multiplier $\lambda_i \geq 0$ for the non-negativity constraint on the slack variables $\xi_i \geq 0$.

The Lagrangian for the primal problem is:

$$L(w, b, \xi_i, \alpha_i, \lambda_i) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + b) - (1 - \xi_i)] - \sum_{i=1}^{N} \lambda_i \xi_i$$

Step ii: Derivatives/ Stationarity Conditions

To find the optimal solution, we compute the partial derivatives of the Lagrangian with respect to w, b, and ξ_i, and set them to zero.

- **Partial derivative with respect to w:**

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

- **Partial derivative with respect to b:**

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{N} \alpha_i y_i = 0$$

- **Partial derivative with respect to ξ_i:**

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i = 0 \quad \Rightarrow \quad \alpha_i = C - \lambda_i$$

Since $\lambda_i \geq 0$ and $\alpha_i \geq 0$, we must have $0 \leq \alpha_i \leq C$ for each i.

Step iv: Substituting into the Lagrangian

Substituting $w = \sum_{i=1}^{N} \alpha_i y_i x_i$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$ into the Lagrangian gives us:

$$L(w, b, \xi_i, \alpha_i, \lambda_i) = \frac{1}{2} \left(\sum_{i=1}^{N} \alpha_i y_i x_i \right)^T \left(\sum_{i=1}^{N} \alpha_i y_i x_i \right) - \sum_{i=1}^{N} \alpha_i$$

Step v: Dual Problem

The objective is to maximize the Lagrangian with respect to the Lagrange multipliers α_i, subject to the constraints $\sum_{i=1}^{N} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$.

Thus, the dual optimization problem is:

$$\min_{\{\alpha_i\}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{N} \alpha_i$$

subject to:

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C \quad \forall i = 1, 2, \ldots, N$$

Problem 3

a.

In the dual form of the SVM, If an example stays outside the margin, it must be correctly classified, meaning it lies on the side of the hyperplane where the decision rule $)y_i(w^T x_i + b) \geq 1$ holds.

The optimal Lagrange multipliers α_i that correspond to such an example will be strictly positive but not necessarily equal to C

Therefore, for an example x_i to stay outside the margin: $0 < \alpha < C$

b.

In the SVM dual formulation, examples that “just sit on the margin” are called support vectors. These examples are those for which the decision boundary is exactly on the margin. The support vectors are associated with Lagrange multipliers α_i that are exactly equal to C. This means that the training examples that lie exactly on the margin correspond to those examples for which the Lagrange multipliers are at the upper boundary of the allowed range, i.e., $\alpha_i = C$.

Thus, to identify which examples just sit on the margin: - Look for the training examples where the corresponding Lagrange multipliers α_i are equal to C.

Problem 4

The kernel trick allows SVMs to perform nonlinear classification by mapping the input data into a higher-dimensional feature space where a linear decision boundary can separate the classes. Instead of explicitly transforming the data, we use a kernel function $K(x_i, x_j)$ to compute the inner product between the data points in this higher-dimensional space. Common kernels include the linear kernel $K(x_i, x_j) = x_i^T x_j$, the polynomial kernel $K(x_i, x_j) = (x_i^T x_j + c)^d$, and the Gaussian (RBF) kernel $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$.

To use the kernel trick, replace the dot product $x_i^T x_j$ in the SVM optimization problem with the kernel function $K(x_i, x_j)$. In the dual problem, this results in the following formulation:

$$\max_{\alpha_i} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \right]$$

subject to:

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C \quad \forall i = 1, 2, \ldots, N$$

Here, $K(x_i, x_j)$ replaces the dot product, allowing SVM to learn a nonlinear decision boundary without the need to explicitly compute the higher-dimensional mapping.

Problem 5

The objective function for SVM is

$$J(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b))$$

Initial Values

Learning rates = [.01,.005,.0025] w = [0,0,0] b = 0 C = 1

Calculations

Steps for X_1:

x_1 = [.5, -1, .3] y_1 = 1

$w^T x_1 + b$ = [0,0,0] + 0 = **0**

$max(0, 1 - y_i(w^T x_1 + b))$ = max(0, 1 - 1(0)) = **1**

$\frac{1}{2}||w||^2$ = $\frac{1}{2}(0^2 + 0^2 + 0^2)$ = 0

$$J(x_1) = \frac{1}{2}(0) + 1(1) = 1$$

Weight and Bias Updates

$\eta_1 = .01$

For the first step, since $y_1(w^T x_1 + b) = 1(0) = 0$, and is less than 1, this point is misclassified and thus, the derivatives are:

$$\nabla_w J(w) = w - C \cdot y_1 \cdot x_1 = [0, 0, 0] - 1(1)([.5, -1, .3]) = [-.5, 1, -.3]$$

$$\nabla_w J(w) = -C \cdot y_1 = -1(1) = -1$$

$$w = w - \eta \cdot \nabla_w J(w) = [0, 0, 0] - .01([-.5, 1, -.3]) = \mathbf{[0.005, -0.01, 0.003]}$$

$$b = b - \eta \cdot \nabla_b J(w) = 0 - .01(-1) = \mathbf{.01}$$

Steps for X_2:

x_2 = [-1, -2, -2] y_2 = - 1 w = [0.005,−0.01,0.003] b = .01 C = 1

$w^T x_2 + b = [0.005, -0.01, 0.003]^T \cdot [-1, -2, -2] + .01$= -.005 + .02 - .006 + .01 = **.019**

$max(0, 1 - y_2(w^T x_2 + b))$ = max(0, 1 + 1(.019)) = 1.019

$\frac{1}{2}||w||^2$ = $\frac{1}{2}(-1^2 + -2^2 + -2^2)$ = $\frac{9}{2}$

Since $y_2(w^T x_2 + b) < 1$, this point is also misclassified.

Weight and Bias Updates

$$\nabla_w J(w) = w - C \cdot y_2 \cdot x_2 = [0.005, -0.01, 0.003] - 1(-1)([-1, -2, -2]) = \mathbf{[-0.995, -2.01, 2.003]}$$

$$\nabla_w J(w) = -C \cdot y_2 = -1(-1) = 1$$

$$w = w - \eta \cdot \nabla_w J(w) = [0.005, -0.01, 0.003] - .005([-0.995, -2.01, 2.003]) = \mathbf{[0.009975, 0.00005, -0.007015]}$$

$$b = b - \eta \cdot \nabla_b J(w) = .01 - .005(1) = \mathbf{.005}$$

Steps for X_3:

x_3 = [1.5, .2, -2.5] y_3 = 1 w = [0.009975,0.00005,−0.007015] b = .005

$w^T x_3 + b = [0.009975, 0.00005, -0.007015]^T \cdot [1.5, .2, -2.5] + .005$ = **.03751**

Since $y_3(w^T x_3 + b) < 1$, this point is also misclassified.

Weight and Bias Updates

$$\nabla_w J(w) = w - C \cdot y_3 \cdot x_3 = [0.009975, 0.00005, -0.007015] - 1(1)([1.5, .2, -2.5]) = \mathbf{[-1.490025, -0.19995, 2.492985]}$$

$$\nabla_w J(w) = -C \cdot y_3 = -1(1) = -1$$

$$w = w - \eta \cdot \nabla_w J(w) = [0.009975, 0.00005, -0.007015] - .0025([-1.490025, -0.19995, 2.492985]) = [\mathbf{0.0137000625, 0.000549875, -0.01}$$

$$b = b - \eta \cdot \nabla_b J(w) = .005 - .0025(1) = \mathbf{.0075}$$

Final Parameters

$$w = [\mathbf{0.0137000625, 0.000549875, -0.0132474625}]$$

$$b = \mathbf{.0075}$$

Problem 6

2a.

Weights: [-0.39393161 0.02255433 -0.26397128 -0.09836142]

Bias: 0.5701525632643287

C = 0.1145475372279496: Train Error = 0.1206, Test Error = 0.1263 C = 0.572737686139748: Train Error = 0.2560, Test Error = 0.2445 C = 0.8018327605956472: Train Error = 0.3605, Test Error = 0.3627

Training with C=0.1145475372279496 Training with C=0.572737686139748 Training with C=0.8018327605956472

2b.

Weights: [-0.35410777 -0.19070304 -0.23062686 -0.01136551]

Bias: 0.6526475784113652

C = 0.1145475372279496: Train Error = 0.0620, Test Error = 0.0862 C = 0.572737686139748: Train Error = 0.0241, Test Error = 0.0261 C = 0.8018327605956472: Train Error = 0.0241, Test Error = 0.0180

2c.

Comparing the results in part 2a and 2b, the weights and bias are actually very similar. However, the model in part 2b did significantly better on the test data. There's also an interesing pattern where in the first model, as the value of C increased, the test error increased, but for the second model the test error decreased.

3a.

Training Dual SVM with C=0.1145475372279496 C = 0.1145475372279496: Train Error = 0.0482, Test Error = 0.0541 Weight vector (w): [-9.06350971 -3.40151892 -5.16724364 1.3550582] Bias (b): 2.8306744237884534

Training Dual SVM with C=0.572737686139748 C = 0.572737686139748: Train Error = 0.1114, Test Error = 0.1283 Weight vector (w): [-19.6005867 -15.81161638 -14.70560601 6.5254179] Bias (b): 6.629107253222535

Training Dual SVM with C=0.8018327605956472 C = 0.8018327605956472: Train Error = 0.1458, Test Error = 0.1703 Weight vector (w): [-22.26967441 -23.28302632 -16.94624079 8.24237037] Bias (b): 5.633365259023655

Results for 3a:

C = 0.1145475372279496: Train Error = 0.0482, Test Error = 0.0541 C = 0.572737686139748: Train Error = 0.1114, Test Error = 0.1283 C = 0.8018327605956472: Train Error = 0.1458, Test Error = 0.1703

ANSWER FOR 3A : Comparing the results to part 2, the training and test errors are very similar. The model for 2b performs the best, and the model for 3 was notably computationally slower than the models for part 2. There's considerations to account for when choosing a model, such as number of features/ samples in the dataset, and in this case, the primal SVM performed much better on the dataset. However,both model types seemed to have converged, which is good.

3b.

C = 100/873

-Learning rate = .01 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 868

-Learning rate = .5 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 825

-Learning rate = 1 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 807

-Learning rate = 5 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 423

-Learning rate = 100 -Train Error = 0.4466 -Test Error = 0.4429 -Number of Support Vectors: 42

C = 500/873

-Learning rate = .01 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 868

-Learning rate = .5 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 731

-Learning rate = 1 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 558

-Learning rate = 5 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 207

-Learning rate = 100 -Train Error = 0.4466 -Test Error = 0.4429 -Number of Support Vectors: 16

C = 800/873

-Learning rate = .01 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 867

-Learning rate = .5 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 694

-Learning rate = 1 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 530

-Learning rate = 5 -Train Error = 0.5534 -Test Error = 0.5571 -Number of Support Vectors: 207

-Learning rate = 100 -Train Error = 0.4466 -Test Error = 0.4429 -Number of Support Vectors: 14

3B answer:

Compared to the Linear SVM, the Gaussian Kernel performed a lot worse. Something could've been wrong with my algorithm as the training and test error were the same across all values of C and only changed between learning rates when gamma was set to 100. It was opposite of the linear SVM where as C got closer to zero, the test error got lower, but for the Gaussian model the higher the C, the lower the errors. Considering the time it took to run these as well (25-35 minutes each), the best combination was C = 800/873 and Learning rate = 100. This was computationally the fastest and also had the best test error(though it was still really bad).

3c.

Support Vectors Overlap for C=500/873 and Consecutive Gamma Values:

Overlap between learning rate = 0.01 and learning rate = 0.5: 750 support vectors Overlap between learning rate = 0.5 and learning rate = 1: 600 support vectors Overlap between learning rate = 1 and learning rate = 5: 300 support vectors Overlap between learning rate = 5 and learning rate = 100: 16 support vectors

3c answer:

Generally, as the learning rate gets larger, the more support vectors are created by the model. Thus, there are more in-common vectors percentage wise for smaller learning rate values and as the learning rate gets larger, there is less vectors in common.