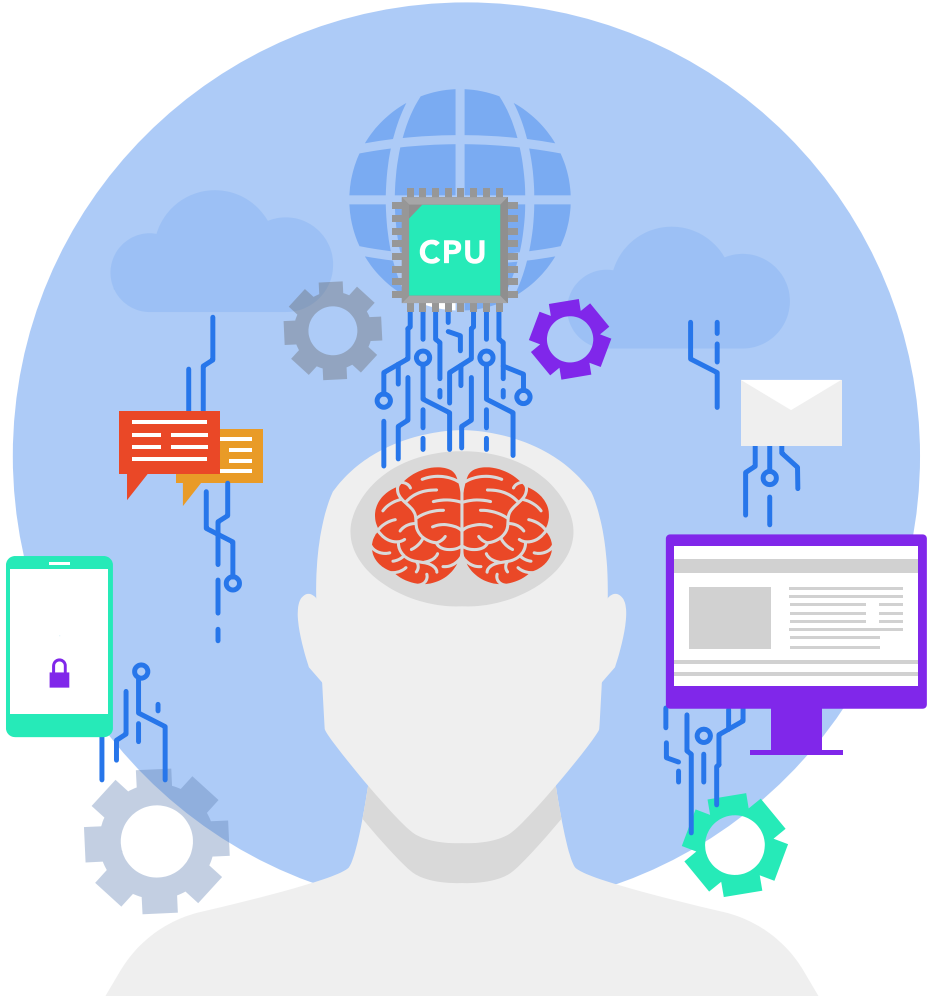
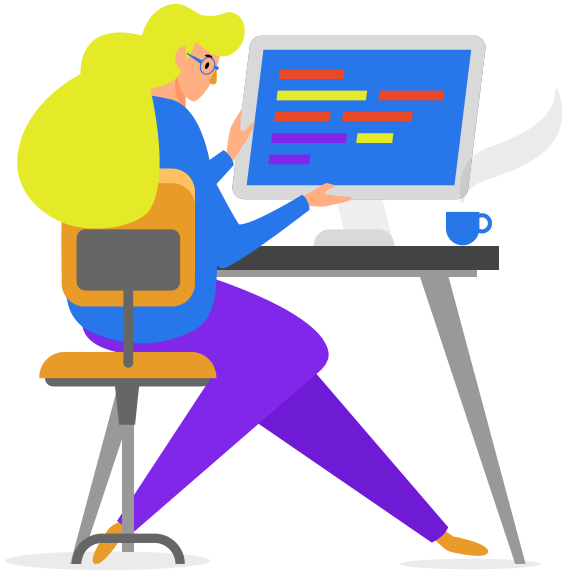


Analysis of Support Vector Machines

SAPD Detectors

23M0764 (Atul Kumar)
23M0765 (Debdoot Roy Chowdhury)
23M0777 (Santanu Sahoo)
23M0800 (Pranab Kumar Paul)

Purpose of the study



01

Evaluation

Evaluate the capabilities and performance of Support Vector Machine (SVM).

02

Explore

Explore the application of SVM, in optical character recognition and in multiclass classification.

03

Analyze

Analyze SVM's adaptability and effectiveness.

04

Provide Insights

Rigorous evaluation of SVM's performance to provide valuable insights into its strengths and limitations.

05

Enhance

Enhance the prediction of two datasets by refining SVM parameters to achieve increased accuracy.

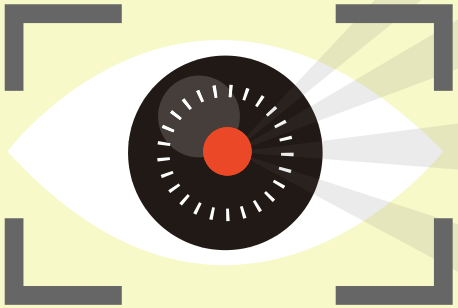
06

Contribute

Contribute to the broader understanding of SVM, advancing t application of machine learning in critical domains.

Implementations / Libraries Used

Some important implementations which are used



- **SVC from sklearn** — SVM Implementations based on libsvm.
- **LinearSVC from sklearn** — SVM Implementations based on liblinear. Used for comparison study.
- **Pytorch** — Used to implement NN models and SVM from scratch
- **Kfold from sklearn** — Utilized for implementing k-fold cross-validation in the study

Dataset Study

Credit Card Fraud Transaction

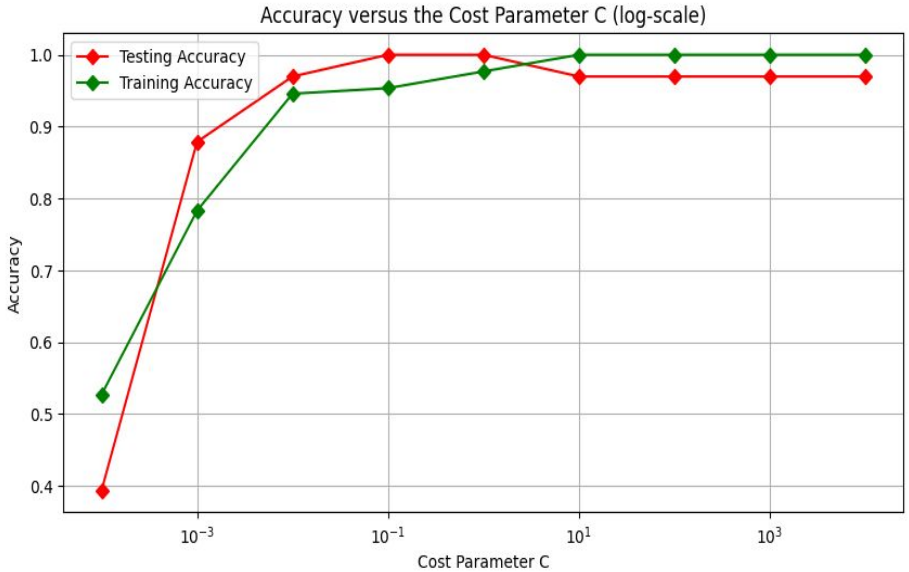
- This dataset is for binary classification.
- This is a highly unbalanced dataset, where we have 492 fraud transactions out of total 284,807.
- There are no ”Null” values.
- It contains only numeric input variables which are the result of a PCA transformation.
- The dataset comprises 30 input features represented as real values, along with an output variable that takes binary values: 0 (indicating ”Not Fraud”) and 1 (indicating ”Fraud”)

Digit Recognizer

- This dataset is for multi-class classification.
- This is a balanced dataset with each class having a total weight of 9-11% out of all the data points.
- Each point is an image of 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total.
- Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker.
- Each pixel of the image is normalized by dividing each value by 255.0, ensuring the pixel values are within the range of 0 to 1.

Analysis on Credit Card Dataset

1. Linear Kernel



Observations

- **C Parameter Impact:** Lower values of C (<1) in the linear SVM model yield higher accuracy, indicating better performance.
- **Optimal Regularization (C=0.1):** Achieves the highest testing accuracy (96.97%), striking a balance between capturing training patterns and generalizing to new data.
- **Overfitting and Complexity:** High regularization (C>1) leads to overfitting, with 100% training accuracy but a decline in testing accuracy, emphasizing the importance of balancing model complexity.
- **Recommendation - Optimal C Value (C=0.1):** C=0.1 is identified as the optimal regularization parameter, providing high testing accuracy without overfitting. Further exploration within a narrow range and experimentation with alternative kernels are suggested for potential enhancement.

Analysis on Credit Card Dataset

2. Polynomial Kernel

Param C	degree	train_acc	test_acc
0.0100	2.0	0.527132	0.393939
0.0100	3.0	0.713178	0.606061
0.0100	4.0	0.736434	0.636364
0.1000	2.0	0.775194	0.757576
0.1000	3.0	0.775194	0.757576
0.1000	4.0	0.775194	0.757576
1.0000	2.0	0.875969	0.909091
1.0000	3.0	0.852713	0.878788
1.0000	4.0	0.790698	0.787879
10.0000	2.0	1.000000	0.939394
10.0000	3.0	0.992248	0.969697
10.0000	4.0	0.922481	0.909091
100.0000	2.0	1.000000	0.878788
100.0000	3.0	1.000000	0.969697

Observations

- **Effect of Hyperparameter C:** Increasing C in the polynomial kernel enhances model accuracy, indicating improved performance on the credit card dataset.
- **Optimal C for Accuracy:** The accuracy consistently rises with increasing C values up to a point, suggesting a trade-off between fitting the training data well and overfitting.
- **Overfitting Concerns:** Beyond a certain threshold, further increases in C lead to diminishing returns and declining accuracy, signaling the onset of overfitting.
- **Recommendation:** Careful selection of C is crucial to balance model complexity. Further exploration within the observed range and experimentation with alternative kernels are advised for optimal performance.

Analysis on Credit Card Dataset

3. Gaussian Kernel

C	gamma	train_acc	test_acc
0.1000	0.001	0.527132	0.393939
0.1000	0.010	0.837209	0.909091
0.1000	0.100	0.930233	0.969697
1.0000	0.001	0.837209	0.909091
1.0000	0.010	0.945736	0.969697
1.0000	0.100	0.992248	1.000000
1.0000	1.000	1.000000	0.636364
10.0000	0.001	0.953488	0.969697
10.0000	0.010	0.976744	1.000000
10.0000	0.100	1.000000	1.000000
10.0000	1.000	1.000000	0.666667
10.0000	10.000	1.000000	0.545455
100.0000	0.001	0.961240	1.000000
100.0000	0.010	1.000000	0.969697

Observations

- **Optimal Hyperparameters for RBF Kernel:** Increasing hyperparameter C in the RBF kernel enhances accuracy, peaking at approximately C = 1.
- **Impact of Regularization (C) and Kernel Coefficient (Gamma):** Low C and Gamma lead to a simplistic model with low accuracy, indicating underfitting. Moderate values of C and Gamma strike a balance, improving accuracy without overfitting. High C and Gamma result in overfitting, with perfect training accuracy but declining testing accuracy.
- **Optimal Parameter Combination:** The best performance is achieved with C = 10 and Gamma = 0.01, demonstrating a delicate balance between complexity and generalization.
- **Influence of Gamma:** High gamma values make the decision boundary intricate, hindering generalization.
- **Generalization vs. Complexity:** The Gaussian kernel's ability to capture intricate patterns requires careful consideration to prevent overfitting and ensure robust generalization.
- **Recommendations for Hyperparameter Tuning:** Optimal performance often lies with moderate values of C and gamma; fine-tuning within this range may provide further improvements.

Analysis on Credit Card Dataset

Overall Analysis

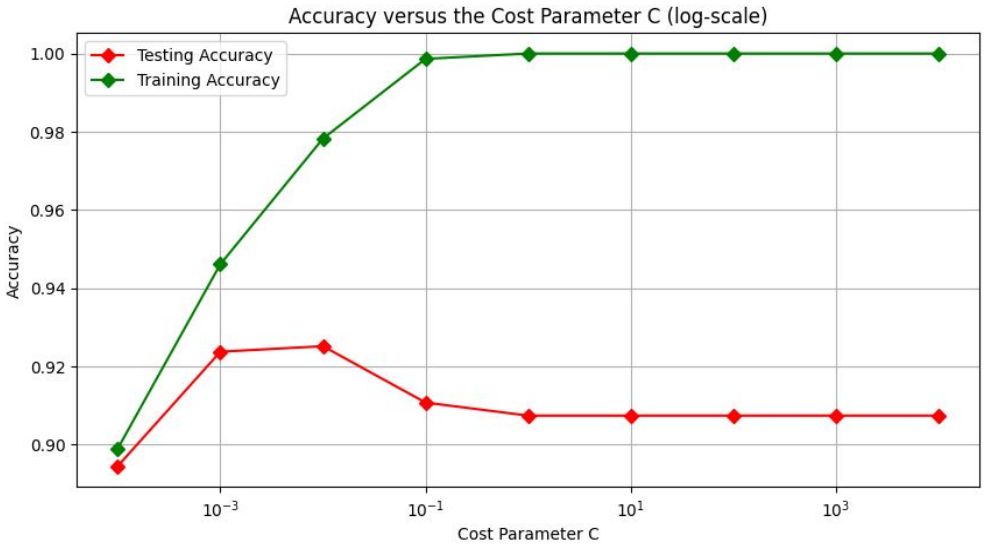
Kernel	Parameters	Train_acc	Test_acc
Linear	C = 1000	0.976	1.000
Polynomial	C = 10 , Degree = 3	0.992	0.969
Gaussian	C = 10 gamma = o.1	1.000	1.000

Observations

- Effect of Small C: Small C allows more misclassifications, leading to underfitting.
- Linear Kernel Strengths: Ideal for well-separated classes in datasets, like credit card fraud detection.
- Computational efficiency and scalability suit large transaction volumes.
- Decision Boundary Clarity: Linear kernel offers a clear, interpretable decision boundary. Gaussian kernel provides flexibility but sacrifices interpretability.

Analysis on Digit Recognizer Dataset

1. Linear Kernel



Observations

- **Optimal C Range:** Optimal C lies in the lower range, particularly around 0.01. Beyond this point, increasing C doesn't significantly enhance accuracy.
- **Effect of Regularization (C):** Smaller C values, indicating higher regularization, result in better generalization. Larger C values may lead to overfitting, emphasizing correct classification on training data.
- Linear Kernel Behavior: In linear kernel SVM, lower C values (e.g., around 0.01) demonstrate improved generalization. Optimal C strikes a balance between fitting training data and preventing overfitting.

Analysis on Digit Recognizer Dataset

2. Polynomial Kernel

Parameters	Best C and Degree	Training Accuracy(%)	Test Accuracy(%)
C = [0.01,0.1,1] Degree = [2]	{C: 1, Degree: 2}	93.34	90.74
C = [0.01,0.1,1] Degree = [3]	{C: 1, Degree: 3}	86.63	89.09
C = [0.01,0.1,1] Degree = [4]	{C: 1, Degree: 3}	58.85	66.07
C = [5,10,15] Degree = [3]	{C: 10, Degree: 2}	94.57	95.21
C = [5,10,15] Degree = [4]	{C: 15, Degree: 3}	94.66	95.16
C = [5,10,15] Degree = [4]	{C: 15, Degree: 4}	89.34	90.46

Conclusion: Comprehensive analysis of SVM with a polynomial kernel on MNIST dataset, highlighting the delicate balance between complexity and generalization. Recommendations guide optimal hyperparameter combinations and suggest avenues for further refinement. The SVM demonstrates versatility in capturing intricate patterns with judicious hyperparameter selection.

Observations

Combination 1: C=1, Degree=2: Balanced model, high accuracy (Training: 0.9335, Test: 0.9074), well-regulated decision boundary.

Combination 2: C=1, Degree=3: Lower accuracy (Training: 0.8664, Test: 0.8909), potential overcomplexity, less effective generalization.

Combination 3: C=1, Degree=4: Significant accuracy drop (Training: 0.5886, Test: 0.6608), overfitting due to high polynomial degree.
Combination 4: C=10, Degree=2: High accuracy (Training: 0.9457, Test: 0.9522), good balance between complexity and generalization.

Combination 5: C=15, Degree=3: Robust to degree choice, higher C (Training: 0.9466, Test: 0.9517), generalizes well without sacrificing accuracy.

Recommendations: Use polynomial of degree 2 or 3 to avoid overfitting. Focus on C=10 or C=15 for consistent performance across degrees. Explore additional hyperparameter values or alternative kernels for optimization.

Analysis on Digit Recognizer Dataset

3. Gaussian Kernel

Parameters	Best Parameter	Training Accuracy	Test Accuracy
{C : [0.1], Gamma:[0.001,0.01,0.1,1]}	C : 0.1, Gamma: 0.001	0.838	0.910
{C : [1], Gamma:[0.001,0.01,0.1,1]}	C : 1, Gamma: 0.001	0.922	0.933
{C : [1], Gamma:[0.001,0.01,0.1,1]}	C : 10, Gamma: 0.001	0.929	0.941
{C : [1], Gamma:[0.001,0.01,0.1,1]}	C : 0.1, Gamma: 5	0.119	0.107
{C : [1], Gamma:[0.001,0.01,0.1,1]}	C : 0.1, Gamma: 5	0.119	0.107
{C : [1], Gamma:[0.001,0.01,0.1,1]}	C : 1, coef0 : 0.0, Gamma: 0.001	0.929	0.941

Observations

- **Optimal Parameter Combination:** Best performance is observed with moderate regularization (C=10) and small gamma (0.01).
- Achieves high accuracy on both training (100%) and test sets (96.4%).
- **Impact of C and Gamma:** Higher C values result in complex decision boundaries, fitting training data closely. Lower C values encourage simpler decision boundaries, promoting generalization.
- **Lower gamma values** lead to smoother decision boundaries, aiding generalization. Higher gamma values introduce more localized decision boundaries, risking overfitting.

- **Overfitting and Underfitting:** High gamma (10) and low C (0.1) combinations exhibit overfitting (Test Accuracy: 13%). Low gamma (0.01) with low C (0.1) results in underfitting, leading to poor performance (Test Accuracy: 13%).
- **Generalization Challenges:** High gamma values (5) in some combinations hinder generalization. Consistency in optimal parameters is observed with low gamma (0.01) and higher C (10).
- **Recommendations:** Emphasize finding a balance between C and gamma for optimal model complexity and generalization. Perform a detailed search for hyperparameters and consider preprocessing or feature engineering. Consistency in optimal parameters (C=10, gamma=0.01) underscores their robustness for the MNIST dataset.

Analysis on Digit Recognizer Dataset

Overall Analysis

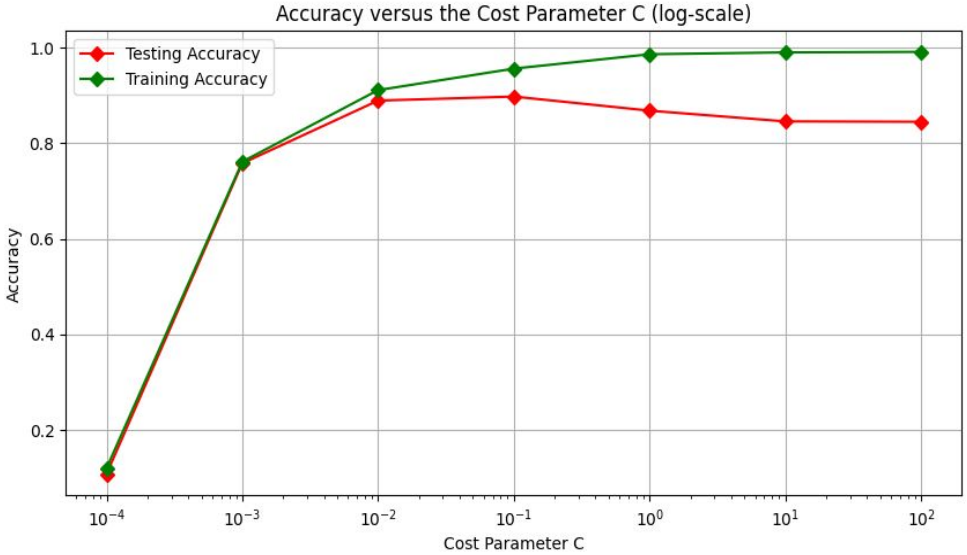
Kernel	Parameters	Train_acc	Test_acc
Linear	C = [5]	0.909	0.907
Polynomial	C = 100 Deg = 3	1.0	0.954
Gaussian	C = 10.00 Gamma = 0.01	1.0	96.4

Limitations of Linear Kernels: Linear kernels may struggle to capture non-linear relationships in MNIST, as they are more effective for linearly separable problems.
Less expressive for complex image datasets like MNIST.
Polynomial Kernel Complexity: Polynomial kernels introduce additional complexity through the degree parameter. Determining the optimal degree is challenging and may lead to overfitting or underfitting

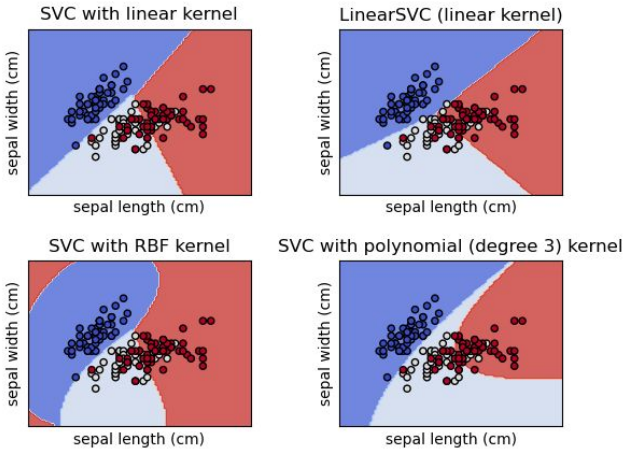
Observations

- **Gaussian Kernel Suitability:** MNIST's high-dimensional images (28x28 pixels) benefit from the Gaussian kernel's effectiveness in capturing relationships between pixels at different positions.
  Well-suited for recognizing complex patterns in handwritten digits due to its ability to model non-linear patterns.
- **Robustness to Noise:** Gaussian kernels can be robust to noise in MNIST data, advantageous when dealing with potential pixel-level noise.
  The ability to smooth decision boundaries aids in handling noise effectively.

Comparison of SVC with LinearSVC of sklearn



Accuracy plot vs. C for Digit Recognizer Dataset



Predicted Class Distribution for IRIS Dataset using different implementations of SVM

Comparison of SVC with LinearSVC of sklearn

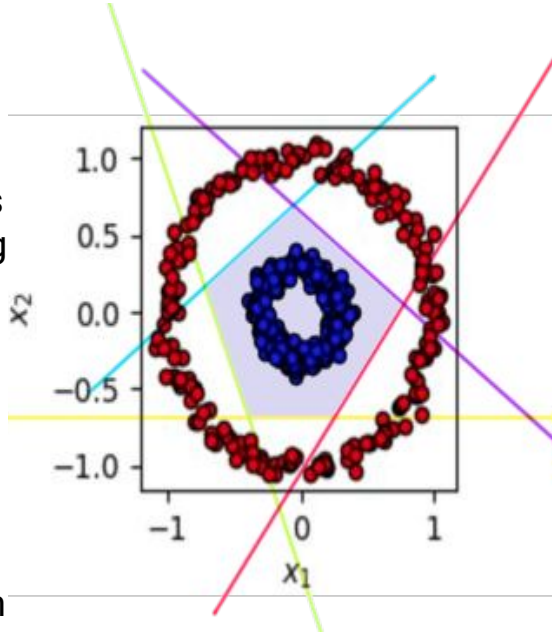
- **LinearSVC vs. SVC in scikit-learn:** LinearSVC is similar to SVC with a linear kernel but implemented with liblinear, offering more flexibility in penalties and loss functions.
- **LinearSVC for Linearly Separable Problems:** LinearSVC is designed specifically for linear kernel SVMs, making it suitable for linearly separable problems and utilizing the liblinear library.
- **Efficiency and Scalability:** LinearSVC's liblinear solver is efficient for linear problems but may not scale as well to very large datasets.
- **LIBSVM Solver in SVC:** SVC uses the LIBSVM library based on the SMO algorithm, allowing it to handle non-linear problems effectively.
- **Multi-Class Classification Strategies:** LinearSVC uses the one-vs-the-rest (OvR) scheme, fitting N models, while SVC supports both OvR and one-vs-one (OvO) schemes, fitting N * (N - 1) / 2 models depending on configuration.
- **Default Loss and Customization:** By default, LinearSVC minimizes the squared hinge loss, while SVC minimizes the regular hinge loss. Customization allows manually defining a 'hinge' string for the loss parameter in LinearSVC.

SVM implementation using PyTorch

- **Unified Model Implementation:** Implemented a unified model for Linear SVM using pyTorch, training with gradient descent on Hinge Loss and hyperparameter C.
- **Handling Imbalanced Data:** Addressed dataset imbalance through under-sampling, ensuring an equal number of instances for both classes and updating class labels.
- **Exploding Gradient Challenge:** Faced exploding gradient issue after 7 epochs, resolved by normalizing feature values and experimenting with learning rate, epochs, and C values.
- **Individual SVMs and Ensemble Approach:** Attempted an ensemble approach with 10 SVMs using varying parameters, but it didn't significantly improve accuracy.
- **Optimizing Hyperparameter C:** Started with C=1, observed overfitting (81% training, 51% test accuracy), gradually reduced C, achieving the best result at C=0.1 with a test accuracy of 62%.
- **Challenges and Outcome:** Despite efforts, the final accuracy remained at 62%, prompting exploration of different strategies for improvement.

Comparison with Neural Network (Historical Significance)

- **Evolution and Importance:** SVMs were crucial for high-dimensional challenges before the prominence of neural networks in the mid-2010s, particularly in text classification.
- **Decision Boundary Modeling:** While logistic regression struggles with non-linear relationships, both SVMs and neural networks excel in non-linear tasks, leveraging appropriate kernels or activation functions.
- **Parametric Similarities:** SVMs and neural networks are parametric, with SVMs involving soft-margin (C) and kernel parameters, and neural networks requiring more parameters like layer size, number of layers, training epochs, and learning rate.
- **Performance and Preference:** Both SVMs and NNs handle similar classification problems with comparable accuracy. NNs tend to outperform SVMs with ample training and computational power.
- **Structural and Procedural Differences:** SVM parameters increase linearly with input size, while NNs, especially multi-layered ones, allow limitless complexity. SVMs use support vectors, requiring fewer observations, while NNs depend on batch order and demand processing the entire dataset. SVMs train quickly, and restarting training is feasible, whereas NNs, especially large ones restarting is expensive due to sensitivity to weight initialization and gradient descent.



A Non-Linear classification task done by neural network, dividing the two class forming a pentagon

Related Work

1. Yue, Shihong, Ping Li, and Peiyi Hao. "SVM classification: Its contents and challenges." Applied Mathematics-A Journal of Chinese Universities 18 (2003): 332-342.

2. Jakkula, Vikramaditya. "Tutorial on support vector machine (svm)." School of EECS, Washington State University 37.2.5 (2006): 3.

3. Chandra, Mayank Arya, and S. S. Bedi. "Survey on SVM and their application in image classification." International Journal of Information Technology 13 (2021): 1-11.

4. Kurani, Akshit, et al. "A comprehensive comparative study of artificial neural network (ANN) and support vector machines (SVM) on stock forecasting." Annals of Data Science 10.1 (2023): 183-208.

5. Liu, Zhijie, et al. "Study on SVM compared with the other text classification methods." 2010 Second international workshop on education technology and computer science. Vol. 1. IEEE, 2010.

Conclusion

- **The analysis of Support Vector Machines (SVMs)** reveals their robustness and versatility in various applications, solidifying their status as one of the most powerful machine learning tools to date.
- **Versatility and Robustness:** SVMs showcase robustness and adaptability in various applications, making them a versatile tool.

- **Optimal Applications:** Excel in scenarios requiring clear class distinctions, evident in tasks like credit card fraud detection and precise digit recognition.

- **Strengths:** Effective in high-dimensional spaces, capable of handling both linear and non-linear relationships through kernel methods. Well-suited for tasks with limited data, exhibiting resilience to overfitting.

- **Enduring Significance:** SVMs strike a balance between simplicity and complexity, contributing significantly to artificial intelligence and data science. One of the most powerful and enduring machine learning tools due to their timeless efficacy.

Work Split among Members

- **23M0764 - Atul Kumar** - SVM Implementation using PyTorch, training it Credit-Card Fraud Dataset and analysis of the results.
- **23M0765 - Debdoot Roy Chowdhury** - Analysis of the datasets, comparison of SVC model of sklearn with LinearSVC, and comparison of SVM with Neural Network.
- **23M0777 - Santanu Sahoo** - Training and Testing of SVM with Digit Recognizer Dataset, and analysis of the results.
- **23M0800 - Pranab Kr Paul** - Training and Testing of SVM with Credit Card Fraud Detection Dataset, and analysis of the results.

References

- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- https://www.baeldung.com/cs/svm-vs-neural-network
- https://stats.stackexchange.com/questions/510052/are-neural-networks-better-than-svms
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
- https://dmkothari.github.io/Machine-Learning-Projects/SVM_with_MNIST.html
- https://github.com/nathanlem1/SVM_PyTorch/
- https://www.kaggle.com/code/faressayah/support-vector-machine-pca-tutorial-for-beginner
- https://www.kaggle.com/code/fareselmenshawii/svm-from-scratch
- https://www.researchgate.net/profile/Ding-Xuan-Zhou/publication/238652514_Analysis_of_support_vector_machine_classification/links/59ba3765aca27241618ee2b9/Analysis-of-support-vector-machine-classification.pdf
- https://www.econstor.eu/bitstream/10419/27334/1/576821438.PDF
- https://www.kaggle.com/code/sabasiddiqi/svm-classification-parameter-selection-0-968