Tarea N°1: SVMs

Valentina Paz Campos Olguín

I. Resumen

Este informe evalúa el rendimiento de los clasificadores Support Vector Machine (SVM) con diferentes kernels (RBF, polinomial) con el conjunto de datos de QuickDraw. El conjunto contiene 10 clases de imágenes hechas a mano, cada una con un tamaño de 256 x 256 píxeles. Usando técnicas de reducción de dimensionalidad como PCA y UMAP, se ha logrado optimizar los hiperparámetros usando GridSearch. Los experimentos muestran que SVM con kernel RBF rinden mejor que el kernel polinomial.

PCA, siendo más rápido, mantuvo una mayor precisión comparado a UMAP, el cual mostró menor precisión en los modelos. El mejor rendimiento fue conseguido al ajustar los hiperparámetros SVM, a pesar de que las precisiones de los modelos se mantuvieran moderados, sugiriendo que agregar datos o ingeniería avanzada en características podría mejorar los resultados.

II. Introducción

La clasificación de imágenes, en el contexto de aprendizaje automático, es una tarea bastamente explorada con aplicaciones en diferentes áreas, tales como salud, visión de computadoras, y sistemas autónomos. La meta de este experimento es evaluar el rendimiento de SVMs al clasificar dibujos hechos a mano ocupando el dataset QuickDraw-10. Cada imagen de este conjunto de datos tiene un tamaño de 256x256 píxeles, y la información está organizada en 10 clases, representando diferentes objetos, como por ejemplo ”león”, çastillo”, ”parche curita”, y más.

El objetivo principal de este estudio es comparar la efectividad de los clasificadores SVM con diferentes kernels (RBF y polinomial). Dada la naturaleza de alta dimensionalidad de la información, se emplearon técnicas de reducción de dimensionalidad como PCA (Principal Component Analysis) y UMAP (Uniform Manifold Approximation and Projection).

Este experimento también involucra la búsqueda de hiperparámetros para optimizar los valores de C y gamma, los cuales impactan de forma significativa en la habilidad del modelo de generalizar adecuadamente en los datos no vistos.

III. Desarrollo

III-A. Materiales

El experimento fue ejecutado en una máquina virtual de Google Cloud con las siguientes especificaciones:

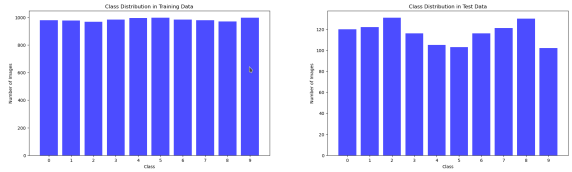
Tabla I: Implementos utilizados

Materiales	Especificación
Máquina Virtual	e2-highmem-4
CPU	4 virtuales
RAM	32GB
Sistema operativo	Linux
Distribución	Debian
Arquitectura	x86-64
Versión	12

La máquina virtual permite escalar recursos, lo que facilita el manejo eficiente de conjuntos de datos grandes y tareas de aprendizaje automático. Durante los experimentos, se monitorearon los recursos (RAM y CPU) utilizando el comando `htop` para asegurar que el sistema no se colapsara, especialmente durante el entrenamiento y ajuste de hiperparámetros de los modelos.

III-B. Preprocesamiento

El conjunto QuickDraw-10 consta de 10 clases, con 9834 imágenes para entrenamiento y 1166 imágenes para evaluación. Se empleó la librería “Scikit-learn”, la cual ofrece una gran variedad de algoritmos de ML [1].



(a) Conjunto de entrenamiento (b) Conjunto de evaluación

Los histogramas muestran una distribución balanceada entre las clases, con aproximadamente 1000 imágenes por clase en el conjunto de entrenamiento y entre 100 y 125 en el conjunto de evaluación. Esto asegura una representación equitativa para cada clase y contribuye a una evaluación confiable y sin sesgo de los modelos.

Las imágenes fueron estandarizadas con `StandardScaler` para asegurar que cada característica tuviera una media de 0 y varianza de 1. Además, se aplicó reducción de dimensionalidad utilizando PCA y UMAP, reduciendo cada imagen a 256 componentes, lo que facilita el procesamiento al reducir la complejidad computacional de las imágenes originales, que tienen 65,536 características.

III-C. Diseño del modelo e implementación

Se implementaron tres clasificadores SVM con diferentes funciones de kernel: RBF (Radial Basis Function) y polinomial. Los modelos fueron entrenados con

GridSearchCV, donde se ajustaron los hiperparámetros de C y gamma.

Los hiperparámetros incluyeron:

1. **C**: Parámetro de regularización, el cual balancea la maximización del margen y minimizar los errores de clasificación. Se seleccionaron los rangos de valores desde 1 a 100 para asegurar que se han explorado los escenarios de underfitting y overfitting. Valores más bajos de C llevan a tener una regularización más alta, lo que puede prevenir el sobreajuste pero puede bajo-ajustar la información. Valores de C más grandes permiten al modelo adaptarse al conjunto de entrenamiento pero puede llevar a sobreajuste. El rango escogido ayuda a balancear estos dos extremos [2].
2. **Gamma**: Define la influencia de cada punto individual de los datos. Un valor más pequeño resulta en un límite de decisión más suave, mientras que un valor más grande resulta en una línea de decisión más compleja. Los valores de gamma fueron escogidos en el rango desde 0.001 a 0.1 (incluyendo 'scale' siendo una opción que se ajusta automáticamente en función del n° de características) para probar tanto límites de decisión suaves como complejas.
3. **Degree**: Controla la complejidad de la línea de decisión para el kernel polinomial. Mientras más alto sea el valor del grado, más complejos serán los límites de decisión. Y si son bajos, por ejemplo considerando grado igual a 1, se producen límites de decisión lineales. Para el kernel polinomial, se ha seleccionado el de grado 3 (kernel cúbico) como default, ya que mantiene un buen balance entre la complejidad y el rendimiento del modelo.

Se usó GridSearchCV con una validación cruzada de 5 *folds* para evaluar el rendimiento de cada modelo con diferentes configuraciones de C y gamma, y los mejores parámetros fueron seleccionados en base a la precisión (accuracy).

III-D. Estructura de código

El código está organizado de modo que permite una fácil modificación de hiperparámetros, kernels y conjuntos de características. Los pasos más importantes del código son los siguientes:

1. Cargar los datos y preprocesamiento: Las imágenes son cargadas, las etiquetas son asignadas y la información es estandarizada.
2. Entrenamiento de los modelos: Los modelos SVM son entrenados con diferentes kernels (RBF y polinomial) y técnicas de reducción de dimensionalidad (sin reducción, PCA y UMAP).
3. Ajuste de hiperparámetros: Se emplea GridSearchCV para ajustar C y gamma en cada modelo, con una elección de 100 imágenes por clase.
4. Evaluación: Precisión y matrices de confusión son usados para evaluar el rendimiento de cada modelo.

IV. Resultados Experimentales

IV-A. Vista general del experimento

El objetivo principal de este experimento fue evaluar el rendimiento de los clasificadores SVM con kernels RBF y polinomial de grado 3 en el conjunto de datos QuickDraw-10. Para reducir la complejidad computacional, se aplicaron técnicas de reducción de dimensionalidad como PCA y UMAP, además de utilizar imágenes en crudo como vectores.

Los modelos probados fueron: Kernel RBF con imagen vectorizada, PCA y UMAP, y Kernel Polinomial (grado 3) con las mismas características.

Los resultados de estos modelos fueron analizados basándose en su precisión y matrices de confusión. El impacto de la reducción de dimensionalidad en el rendimiento del modelo también fue estudiado.

IV-B. Precisión de modelo

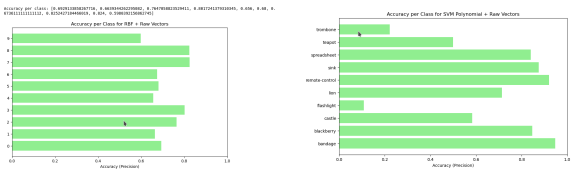
Tabla II: Precisión de cada modelo

Kernel	Raw Vectors	PCA	UMAP
RBF	0.717	0.771	0.298
Polinomial	0.229	0.625	0.277

Los resultados indican que el kernel RBF tiene un mejor rendimiento en general, especialmente con PCA, que alcanzó la mejor precisión con un 77.10 %. Este rendimiento superior se debe a cómo PCA reduce la dimensionalidad, manteniendo las características más importantes, lo que permite que el kernel RBF, que captura relaciones no lineales, trabaje más eficientemente en un espacio de menor dimensión. En comparación, el kernel polinomial mostró un rendimiento significativamente más bajo, especialmente con UMAP, que mostró una baja precisión debido a su enfoque en preservar relaciones locales a expensas de las características globales. El uso de Raw Vectors, aunque preciso, fue computacionalmente costoso debido a la gran cantidad de características.

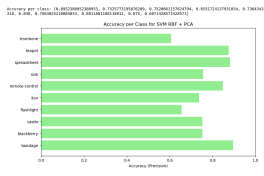
IV-C. Precisión de cada modelo por clase

La precisión de cada modelo se evaluó también para cada clase individual del conjunto de datos. Clases como "bandage" y "remote-control" tuvieron un buen rendimiento en la mayoría de los modelos, mientras que clases como "flashlight" y "trombone" tuvieron dificultades, especialmente con PCA y UMAP. La siguiente figura muestra la precisión por clase para los modelos más representativos:

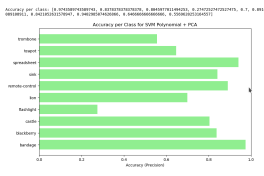


(a) RBF + Raw Vectors (b) Poly + Raw Vectors

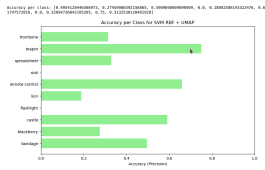
UMAP mostró el peor rendimiento en comparación con PCA y Raw Vectors debido a su enfoque en preservar relaciones locales, lo que limita su capacidad para capturar



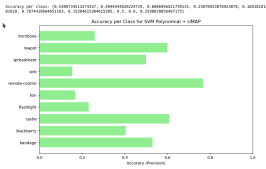
(a) RBF + PCA



(b) Poly + PCA



(a) RBF + UMAP

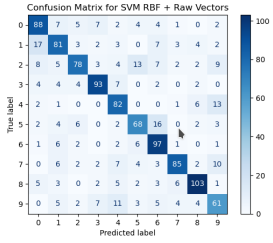


(b) Poly + UMAP

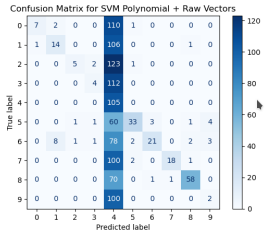
patrones globales importantes para la clasificación. Los resultados sugieren que PCA, al reducir la dimensionalidad de manera más controlada, ofrece una mejor compensación entre precisión y eficiencia computacional, especialmente para el kernel RBF.

IV-D. Matrices de confusión

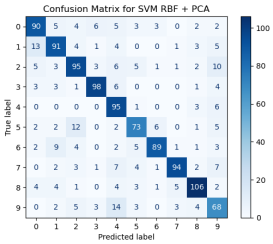
Las matrices de confusión fueron usadas para analizar de mejor forma las predicciones de los modelos.



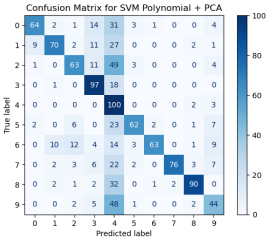
(a) RBF + Raw Vectors



(b) Poly + Raw Vectors



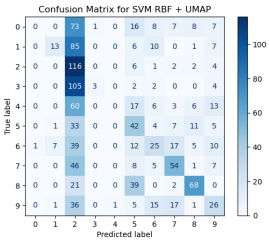
(a) RBF + PCA



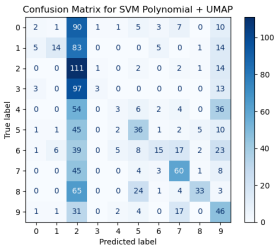
(b) Poly + PCA

Las matrices de confusión muestran que los modelos SVM con kernel RBF tienen un mejor rendimiento en la mayoría de las clases, especialmente con Raw Vectors y PCA, donde clases como “bandage”, “remote-control” y “sink” fueron clasificadas con alta precisión. Sin embargo, clases como “trombone” y “flashlight” siguen siendo desafiantes, con muchas clasificaciones incorrectas, especialmente en PCA y UMAP, donde se pierden características clave.

Los modelos con Raw Vectors, particularmente con el kernel RBF, manejan mejor estas clases complejas, aunque a un costo computacional mayor. En cambio, el kernel



(a) RBF + UMAP



(b) Poly + UMAP

polinomial presenta más dificultades, especialmente con PCA y UMAP, debido a su naturaleza más compleja que no generaliza bien con características reducidas. UMAP, al enfocarse en relaciones locales, tiene un peor rendimiento, especialmente en clases difíciles de distinguir, debido a la pérdida de información importante.

IV-E. Tiempos de entrenamiento por modelo

Otro factor relevante para evaluar los modelos fue su eficiencia computacional, particularmente durante la fase de entrenamiento. Ya que el conjunto de datos contiene imágenes con alta dimensionalidad, técnicas de reducción de dimensionalidad como PCA y UMAP ayudaron a mejorar el tiempo de entrenamiento y su respectiva eficiencia.

Tabla III: Tiempos en segundos

Kernel	Raw Vectors	PCA	UMAP
RBF	1832 [s]	5.4 [s]	5.6 [s]
Polinomial	2757 [s]	8.4 [s]	3 [s]

En cuanto al tiempo de entrenamiento, el kernel polinomial + Raw Vectors tardan significativamente más que el kernel RBF + Raw Vectors, el cuál podría tener relación con la naturaleza del propio kernel polinomial. Este involucra el cálculo a pares de interacciones de características, los cuales aumentan la complejidad computacional, particularmente cuando el espacio de características es grande. Esto quiere decir que RBF + Raw Vectors se beneficia de su habilidad de mapear los datos eficientemente en dimensiones más grandes sin la necesidad de computar explícitamente las interacciones, haciéndolo más rápido.

En modelos como RBF + PCA y RBF + UMAP, donde la reducción de dimensionalidad es aplicada, el kernel RBF también exhibe tiempos de entrenamiento mucho más rápidos. Esto es porque tanto PCA como UMAP reducen la complejidad de la información a través de la disminución de la cantidad de características, por ende, disminuyendo los costos computacionales. El kernel RBF se beneficia de esta reducción ya que está mejor equipado para manejar con datos de alta dimensionalidad, haciéndolo más rápido cuando se trabaja con características derivadas de estas técnicas de reducción.

Considerando todo lo anterior, el rendimiento y costos de tiempo sugieren que el kernel RBF es una opción más eficiente cuando se trabaja con PCA. Mientras que el kernel

polinomial, siendo capaz de entregar un gran rendimiento en algunos contextos, tiene problemas con la alta dimensionalidad y características reducidas, llevando tanto a una baja precisión como un mayor tiempo de entrenamiento.

IV-F. Ajuste de hiperparámetros

Tabla IV: Precisión de cada modelo con ajuste

Kernel	Raw Vectors	PCA	UMAP
RBF	0.617	0.597	0.263
Polinomial	0.175	0.258	0.299

Tabla V: Tiempos de ajuste

Kernel	Raw Vectors	PCA	UMAP
RBF	1104 [s]	4.3 [s]	2.2 [s]
Polinomial	1031 [s]	2.7 [s]	27 [s]

Tabla VI: Hiperparámetros usados para cada modelo (C y gamma)

Model	**C**	**Gamma**
RBF + Raw Vectors	10	'scale'
RBF + PCA	10	'scale'
RBF + UMAP	100	0.1
Polynomial + Raw Vectors	1.0	0.001
Polynomial + PCA	1.0	0.001
Polynomial + UMAP	1.0	'scale'

Durante el ajuste de hiperparámetros para los modelos SVM, se utilizaron 100 imágenes por clase, las cuales fueron seleccionadas de forma aleatoria desde el conjunto de entrenamiento. Los resultados indican una varianza significativa en la precisión basados en la combinación del tipo de kernel y la técnica de reducción de características. El kernel RBF con características PCA lograron una mejora razonable, con una precisión de evaluación del 59 %. Sin embargo, el tiempo de ajuste fue relativamente corto con 4 segundos, sugiriendo que la habilidad del kernel RBF de manejar eficientemente el conjunto de características reducidas de PCA. Mientras que el kernel polinomial requirió un tiempo de ajuste de hiperparámetros de 2.7 segundos con las características PCA, pero su precisión fue considerablemente menor, reflejando los desafíos del kernel polinomial al tratar con espacios de dimensiones reducidas. El rendimiento tanto para PCA como UMAP fue bajo, especialmente para el kernel polinomial, el cual resultó en una precisión del 17.5 % cuando es usado con Raw Vectors y 16.7 % para PCA.

El ajuste con el kernel RBF en Raw Vectors y características UMAP fue más exitoso. La mejor precisión alcanzada fue de 61.66 % para el kernel RBF con Raw Vectors, mientras que UMAP logró un 26.33 %. El ajuste para UMAP requirió menos tiempo, pero su representación de características reducidas en UMAP llevó a que se obtuvieran pérdidas en cuanto a la precisión. Este rendimiento bajo podría deberse al comportamiento propio de UMAP: rápido, pero pierde información esencial.

Los resultados sugieren que el modelo con Raw Vectors provee un mejor espacio de características para el kernel RBF que para UMAP, mientras que PCA rinde razonablemente bien para el kernel RBF, pero con el kernel polinomial se complica tanto con técnicas de reducción como con Raw Vectors.

V. Conclusiones

En este estudio, varios modelos SVM con diferentes kernels (RBF y polinomial) fueron evaluados con el conjunto de datos de QuickDraw, el cual contiene 10 clases de imágenes. Los modelos fueron entrenados con y sin técnicas de reducción de dimensionalidad, tales como PCA y UMAP, para entender el impacto en el rendimiento y eficiencia computacional.

Los resultados indican que el kernel RBF es mejor que el kernel polinomial en términos de rendimiento en distintas configuraciones, especialmente cuando se combina con PCA. Esta ténica consiguió la precisión más alta de todas, con un 77.1 %, superando a raw vectors (71.7 %) debido a su habilidad de retener las características más significativas mientras se reduce la dimensionalidad. Esto permitió que el kernel RBF pudiera mapear la información de forma más efectiva, a pesar de su espacio reducido. Por otro lado, UMAP exhibió el peor rendimiento, especialmente con el kernel polinomial, ya que tuvo problemas al preservar patrones globales necesarios para una clasificación adecuada. El kernel polinomial también tuvo complicaciones con información con alta dimensionalidad y fue costoso computacionalmente hablando, específicamente en raw vectors.

Adicionalmente la eficiencia computacional de los modelos variaron significativamente. Redujeron la dimensionalidad usando PCA Y UMAP, logrando disminuir de forma drástica los tiempos de entrenamiento. Sin embargo, el kernel polinomial, consiguió mejores resultados en algunas configuraciones, requirió de mucho más tiempo de entrenamiento, particularmente cuando se usó raw vectors debido a la gran complejidad de calcular las interacciones de las características.

Pero lo más importante a destacar es que PCA ofrece una compensación entre precisión y costo computacional, especialmente para el kernel RBF. Mientras que UMAP es rápido, su rendimiento se ve limitado por si incapacidad de capturar características globales importantes, llevando a tener las precisiones más bajas de los modelos. Estos hallazgos sugieren que para conjuntos de datos como QuickDraw-10, RBF con PCA ofrece un balance óptimo de eficiencia y precisión.

Referencias

[1] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & otros. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830. Recuperado de https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

[2] Schölkopf, B., Smola, A. J., & Platt, J. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7), 1443-1471. https://doi.org/10.1162/089976601750400736