Práctica Final: Máquinas de vectores soporte

Alumno: Pablo Alonso

1)

La implementación usada de svm fue libsvm, y se descargó de esta dirección: https://github.com/cjlin1/libsvm.

A continuación se describen los pasos seguidos para evaluar en 10 10 folds el dataset heladas:

1. Se dividió el dataset en las 2 clases.
2. Se dividió cada clase en 10 subconjuntos lo más uniformes posibles en tamaño.
3. Para crear el fold i, unimos el subconjunto i de la clase 0 con el subconjunto i de la clase 1.

Para cada fold i:

1. El fold i se lo llama test
2. Reunimos el resto de los datos para entrenamiento en un conjunto train.
3. Se extrae aleatoriamente un 20% de los datos del conjunto train y lo llamamos validación y el 80% restante lo llamamos train2.
4. Se optimiza las constantes necesarias entrenando sobre train2 y evaluando con validación.
5. Se recupera el conjunto inicial train haciendo la unión entre train2 y validación.
6. Se entrena un modelo usando las constantes optimizadas obtenidas sobre train.
7. Se evalua test.

Se usa la siguiente tabla para mostrar los resultados obtenidos:

Método	Errores obtenidos en cada fold
SVM - Kernel Lineal	31.38,15.69,19.61,29.42,12.00,22.00,18.37,10.21,24.49,24.49
SVM - Kernel Polinómico	23.53,15.69,15.69,23.53,12.00,26.00,18.37,14.29,20.41,22.45
SVM - Kernel Gaussiano	31.38,19.61,21.57,23.53,14,26,18.37,12.25,24.49,24.49
Árboles	29.4,21.6,15.7,19.6,16.0,32.0,26.5,18.4,28.6,24.5
Naive Bayes	31.37,17.64,19.60,29.41,12,22,20.4,14,26.53,20.40

La siguiente tabla resume las medias y desviaciones estándar en las mediciones hechas:

Método	Promedio	Desv. Est.
SVM - Kernel Lineal	20.7	6.97
SVM - Kernel Polinómico	19.1	4.67
SVM - Kernel Gaussiano	21.5	5.7
Árboles	23.23	5.81
Naive Bayes	21.6	6.2

SVM con kernel polinómico logra los mejores resultados debido a que logra un promedio menor de error y tiene además una baja desv. estándar con respecto al resto de los métodos. SVM con kernel lineal obtiene los mejores resultados después de este. En general, a menos que se sepa que los datos son linealmente separables, usar SVM con kernel polinómico es una mejor idea que usar un kernel lineal puesto que el espacio de soluciones de un kernel lineal está contenido por el espacio de soluciones del kernel polinómico y este es capaz de mejorar esas soluciones buscando otros hiperplanos en espacios de más dimensiones. Obviamente si sabemos que los datos son linealmente separables, SVM con kernel lineal va a ser mejor que cualquier método puesto que este encuentra el hiperplano óptimo que separa las clases. El peor método observado resultan ser los árboles de decisión, los cuales crean soluciones agrupando los datos en hipercubos.

2)

Hipótesis nula: SVM - Polinómico y Árboles de decisión son modelos con igual performance.

Calculamos la media de las diferencias entre los resultados de ambos métodos:

$\bar{d} = 4.034$

Calculamos la desviación estándar de las diferencias entre los resultados de ambos métodos:

$\bar{S_d} = 3.77$

Calculamos t:

$t = (4.034)/(3.77/sqrt(10)) \approx 3.38$

Dado que $t > 2.26$, se puede rechazar la hipotésis nula de que los 2 algoritmos tienen igual performance con 95% de confidencia y como se vio que la esperanza del error de SVM - KL tiende a ser menor tiene mejor performance para este problema.

Hipótesis nula: SVM- KL y SVM - Polinómico son modelos con igual performance

Calculamos $\bar{d}, \bar{S_d}$ y t:

$\bar{d} = 1.57$

$\bar{S_d} = 3.95$

$t = (1.57)/(3.95/sqrt(10)) \approx 1.25$

Dado que $t < 2.26$, no se puede rechazar con 95% de confidencia la hipótesis nula. Es decir, para dicho intervalo de confidencia, el test no es lo suficientemente fuerte para probar que ambos modelos tienen distinta performance.

3)

Se eligió como dataset spambase (se puede encontrar en https://archive.ics.uci.edu/ml/datasets.php/spamba el cual se confeccionó en 1998. La idea era crear un filtrador de spam y este dataset fue uno de los más mencionados tanto en papers como en blogs o foros de discusión de machine learning dando la idea de que la calidad de los datos es buena, además la cantidad de datos que posee (4601 mails, 40% de spam) se asemeja a la cantidad de datos que se manejó durante el curso de la materia, los features miden porcentajes por lo que son números reales entre 0 y 100. Además miden cosas interesantes como la frecuencia de palabras más comunes, frecuencia de carácteres y frecuencia de letras capitales. Posiblemente para un filtrador de spam actual sea necesario considerar otros features como por ejemplo en que horario se mandó el correo,

Para obtener un valor representativo sobre el error de los modelos utilizados se usó la técnica 10-folds. Luego se toma el promedio de todos los resultados obtenidos como valor representativo, como se hizo en el ejercicio 1.

Para las redes neuronales se usó la siguiente configuración:

- 10 nodos en la capa intermedia
- Learning rate: 0.001
- Momentun: 0.1

Para los modelos que necesitan validación se usó el 20% del conjunto de datos de entrenamiento en cada iteración de 10-fold.

A continuación, una tabla con los errores de cada modelo en cada iteración del fold:

Modelo	Errores
SVM con kernel lineal	18.87,19.08,14.31,20.43,16.52,15.21,18.47,16.73,20.04,15.68
Redes Neuronales	76.35,75.05,84.38,75.65,80.21,75, 85,83.91,83.44,78.21
Naive Bayes	19.09,18.66,21.48,18.92,22.61,19.14,21.09,21.74,39.44,18.31
Árboles de decisión	6.1,7.2,6.3,7.8,7.2,8.3,8.7,5.7,6.3,5.9
K nearest neighbours	18.65,19.30,17.35,19.34,16.52,20.86,19.56,17.60,20.91,16.77

La siguiente tabla muestra el error promedio y la desviación estándar en los errores:

Modelo	Error promedio	Desviación estándar
SVM con kernel lineal	17.53	2.12
Redes Neuronales	20.28	4.15
Naive Bayes	22.04	6.29
Árboles de decisión	6.95	1.05
K nearest neighbours	18.68	1.58

Claramente el modelo que mejor resuelve este problema son los árboles de decisión, tanto por su error promedio como por la baja desviación estándar en los errores medidos.

Por los resultados observados, SVM resuelve con más precisión el problema de si un mail es spam o no que RD,NB y KNN salvo C4.5. En términos de bias inductivo, se puede encontrar un hiperplano que separa los datos y que permite clasificar de forma más eficaz que el resto de los modelos y una forma de precisar aún más esta clasificación es la división en hipercubos llevada a cabo por C4.5.