Machine Learning

Introducción a la Inteligencia Artificial - LCC - 2024

Nicolas Soncini

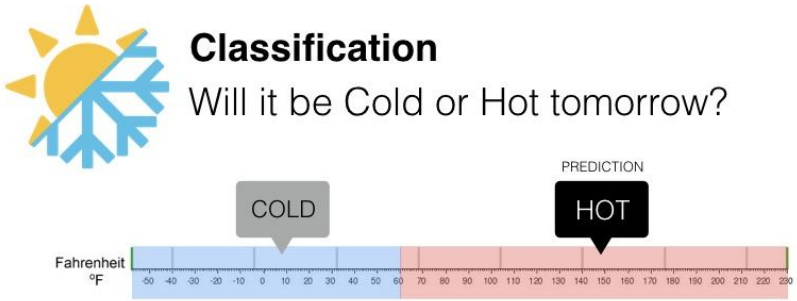
Scikit-Learn

scikit-learn.org

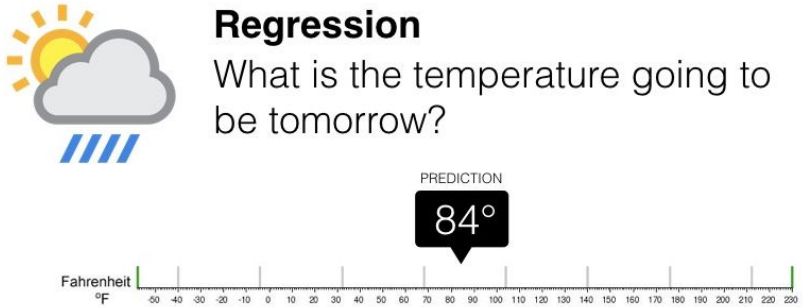
- Análisis de Datos Predictivo
  - Classificación
  - Regresión
  - Clustering
  - y más...
- Desarrollado con:
  - NumPy
  - SciPy
  - Matplotlib

Multi-Layer Perceptron

Clasificación
(sklearn.neural_network.MLPClassifier)

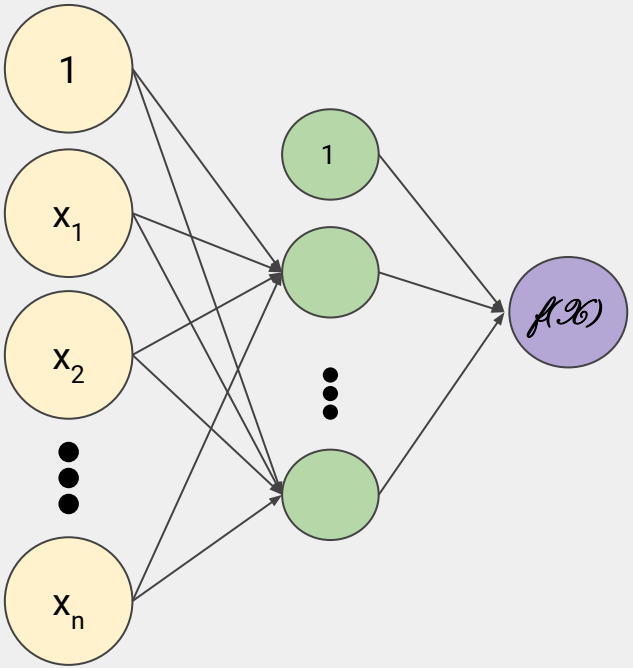


Regresión
(sklearn.neural_network.MLPRegressor)

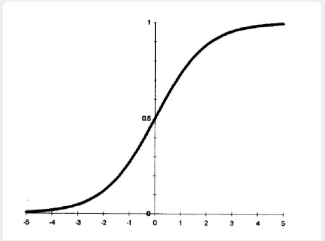


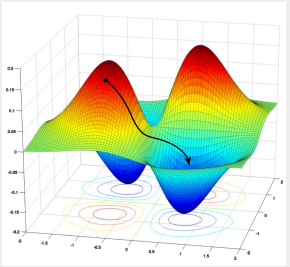
Multi-Layer Perceptron

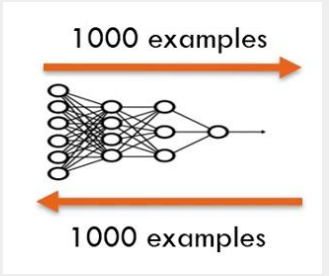
Parámetros Comunes a Ambos
(no los tocamos, pero veamos qué son)



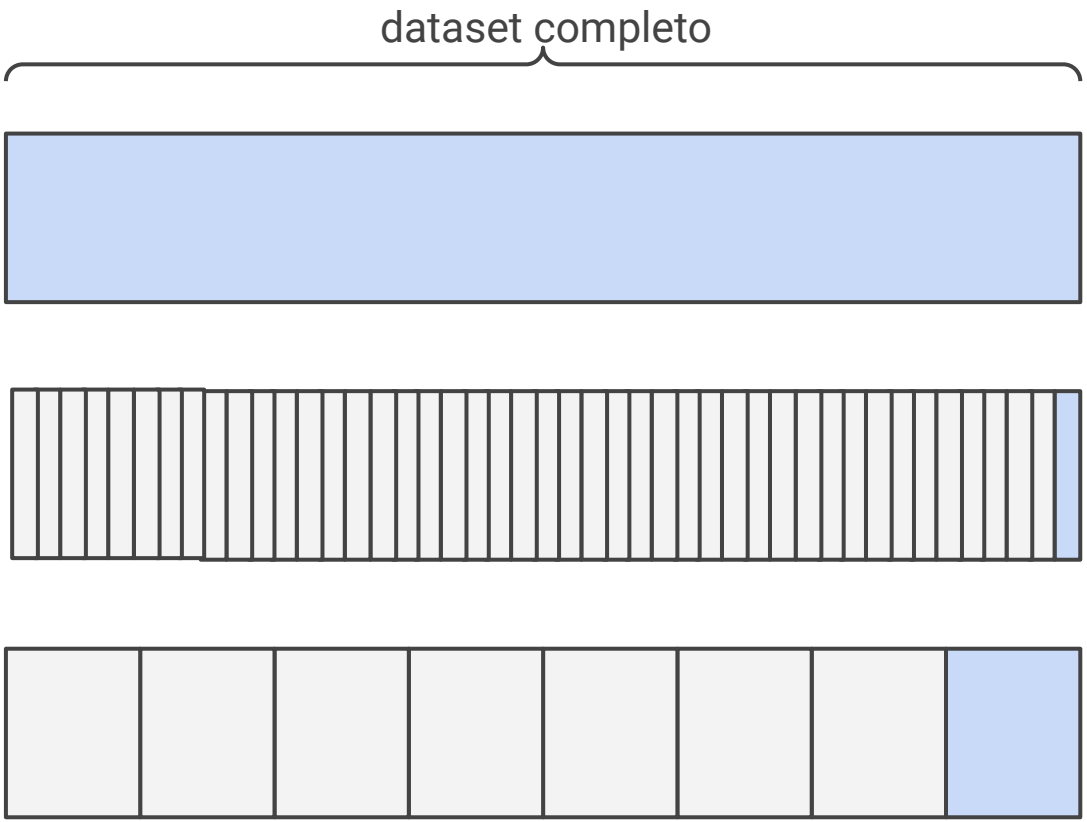
- hidden_layer_sizes = (K,)
- activation = ‘sigmoid’
- solver = ‘sgd’
- batch_size = BATCH_SZ
- max_iter = MAX_ITER_EP







Epochs y Batches



Batch Gradient Descent

Tamaño de **Batch** = N
Iters. por **Epoch** = 1

Stochastic Gradient Descent

Tamaño de **Batch** = 1
Iters. por **Epoch** = N

Mini-Batch Gradient Descent

Tamaño de **Batch** = N/k
Iters. por **Epoch** = k

Ej. 1 - Capacidad de Modelado

Entrene redes neuronales para resolver el problema de **clasificación** de las ***espirales anidadas***.

Use un número creciente de neuronas en la capa intermedia: 2, 10, 20, 40.
Valores posibles para los demás parámetros de entrenamiento:

- learning rate = 0.1
- momentum = 0.9
- 600 datos para ajustar los modelos
  - separar 20% de ese conjunto al azar para conjunto de validación
- 2000 para testear
- 1000 evaluaciones del entrenamiento, cada una de 20 épocas.

Para cada uno de los cuatro modelos obtenidos, graficar en el plano *xy* las clasificaciones sobre el conjunto de test. Comentar.

Ej. 2 - Mínimos Locales

Baje el dataset **dos-elipses** de la descargas y realice varios entrenamientos con los siguientes parámetros:

- 6 neuronas en la capa intermedia
- 500 patrones en el training set
  - 400 se usan para entrenar y 100 para validar el modelo (sacados del .data)
- 2000 patrones en el test set (del .test)
- 300 evaluaciones del entrenamiento, cada una de 50 épocas.

Pruebe distintos valores de **momentum** y **learning-rate** (valores usuales son 0, 0.5, 0.9 para el momentum y 0.1, 0.01, 0.001 para el learning-rate, pero no hay por qué limitarse a esos valores), para tratar de encontrar el mejor mínimo posible de la función error.
El valor que vamos a usar es el promedio de 10 entrenamientos iguales, dado que los entrenamientos incorporan el azar.
Confeccione una tabla con los valores usados para los parámetros y el resultado en test obtenido (la media de las 10 ejecuciones). Haga una gráfica de MSE de train, validación y test en función del número de épocas para los valores seleccionados (los mejores valores de eta y alfa).

Ej. 3 - Regularización

Baje el dataset **Ikeda** y realice varios entrenamientos usando el 95% del archivo .data para entrenar, y el resto para validar.

Realice otros entrenamientos cambiando la relación a 75%-25%, y a 50%-50%.

En cada caso seleccione un resultado que considere adecuado, y genere gráficas del MSE en train, validación y test. Comente sobre los resultados.

Los otros parámetros para el entrenamiento son:

- learning rate = 0.01
- momentum = 0.9
- 2000 datos para testear
- 400 evaluaciones del entrenamiento, cada una de 50 épocas
- 30 neuronas en la capa oculta

Ej. 4 - Regularización 2

Vamos a usar regularización por penalización, el **weight-decay**.
Hay que tener cuidado con los nombres de los parámetros en este caso: el parámetro que nosotros llamamos **gamma** en la teoría corresponde en MLP de sklearn al parámetro **alpha**, mientras que nosotros usamos **alfa** para el **momentum** en general.

En este tipo de regularización no se usa un conjunto de validación, así que hay que modificar la función que crearon para evaluar el entrenamiento de las redes, para que en lugar del error sobre el conjunto de validación, nos devuelva la suma de los valores absolutos o de los valores al cuadrado de todos los pesos de la red en la época correspondiente, y todo el resto igual que antes.

(continúa…)

Ej. 4 - Regularización 2

Una vez implementado, aplíquelo al dataset **Sunspots** (ssp).
Busque el valor de **gamma** adecuado para conseguir un equilibrio entre evitar el sobreajuste y hacer el modelo demasiado rígido (el valor de gamma se debe variar en varios órdenes de magnitud, por ejemplo empezar en 10^-6 e ir hasta 10^0 (1) de a un orden cada vez).
En este caso todos los registros del archivo .data se deben usar para el entrenamiento, ya que la regularización se realiza con la penalización a los pesos grandes.

Los otros parámetros se pueden tomar:

- learning rate 0.05
- momentum 0.3
- 4000 evaluaciones del entrenamiento, cada una de 20 épocas
- 6 neuronas en la capa intermedia.

Entregue curvas de los tres errores (entrenamiento y test en una figura, penalización en otra figura) para el valor de gamma elegido, y para algún otro valor en que haya observado sobreajuste. Comente los resultados.

Ej. 5 - Dimensionalidad

Repita el punto 4 del Práctico 1, usando ahora redes con 6 unidades en la **capa intermedia**.

Los otros parámetros hay que setearlos adecuadamente, usando como guía los casos anteriores.

Genere una gráfica que incluya los resultados de redes y árboles.

Ej. 6 - OPCIONAL: Multiclase

Busque en los datasets de sklearn los archivos del problema **iris** y entrene una red sobre ellos.
Tome un tercio de los datos como test, y los dos tercios restantes para entrenar y validar.
Ajuste los parámetros de la red de manera conveniente.
Realice curvas de entrenamiento, validación y test.

Baje el dataset **Faces**.
Entrene una red neuronal para resolver dicho problema, eligiendo adecuadamente todos los parámetros (hay que re-escalar los datos al rango [0,1] antes de usarlos).
Muestre curvas de entrenamiento para asegurar la convergencia.
Compare los resultados con los citados en Mitchell (pag. 112).

Ej. 7 - OPCIONAL: Minibatch

La implementación de sklearn permite usar **minibatchs** cambiando el parámetro ***batch_size***.

Realice una comparación de las curvas de aprendizaje para el problema de **Sunspots**, utilizando batches de longitud 1 y otros dos valores (3 curvas). Comente los resultados.

Parámetros del entrenamiento:

- 20% de validación
- learning rate 0.05
- momentum 0.3
- 2000 evaluaciones del entrenamiento, cada una de 200 épocas
- 6 neuronas en la capa intermedia.

Los valores del minibatch se deben elegir para que en cada etapa de entrenamiento haya varias actualizaciones del gradiente, o sea que no debería ser mayor a 40.