

Taller de Aprendizaje Automático

Clasificación y regresión

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería



UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

- ① Medidas de desempeño
- ② Análisis de error
- ③ Gradiente descendente
- ④ Curvas de aprendizaje
- ⑤ Regularización

Clasificador binario

5 0 4 1 9 2 1 3 1 4
3 5 3 6 1 7 2 8 6 9
4 0 9 1 1 2 4 3 2 7
3 8 6 9 0 5 6 0 7 6
1 8 7 9 3 9 8 5 9 3
3 0 7 4 9 8 0 9 4 1
4 4 6 0 4 5 6 1 0 0
1 7 1 6 3 0 2 1 1 7
8 0 2 6 7 8 3 9 0 4
6 7 4 6 8 0 7 8 3 1

MNIST dataset

```
y_train_5 = (y_train == 5)  
y_test_5 = (y_test == 5)
```

```
from sklearn.linear_model import SGDClassifier  
sgd_clf = SGDClassifier(random_state=42)  
sgd_clf.fit(X_train, y_train_5)
```

```
>>> some_digit = X[0]  
>>> sgd_clf.predict([some_digit])  
array([True])
```



Evaluar desempeño

- **accuracy**: tasa de acierto

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5,
                    cv=3, scoring="accuracy")
array([0.96355, 0.93795, 0.95615])
```

clasificador *tonto*: nunca 5

```
from sklearn.base import BaseEstimator

class Never5Classifier(BaseEstimator):
    def fit(self, X, y=None):
        pass
    def predict(self, X):
        return np.zeros((len(X), 1),
                        dtype=bool)
```

no sirve para datos desbalanceados!

(solo 10% de los datos son 5)

```
>>> never_5_clf = Never5Classifier()
>>> cross_val_score(never_5_clf, X_train, y_train_5,
                    cv=3, scoring="accuracy")
array([0.91125, 0.90855, 0.90915])
```

Matriz de confusión

TP verdaderos positivos

TN verdaderos negativos

FP falsos positivos

FN falsos negativos

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

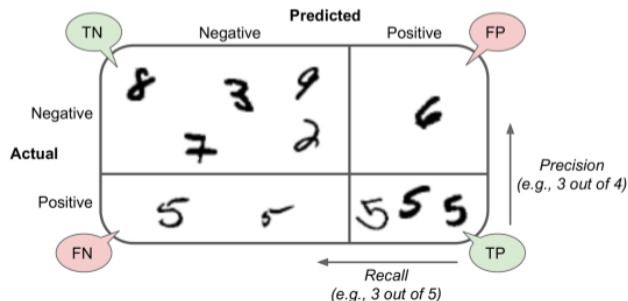
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F_1 = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
```

```
y_train_pred = cross_val_predict(sgd_clf, X_train,
                                  y_train_5, cv=3)
```

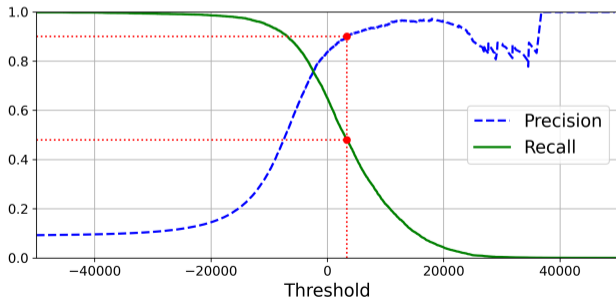
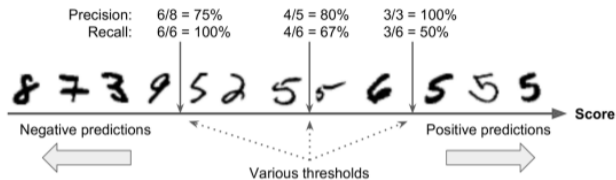
```
>>> confusion_matrix(y_train_5, y_train_pred)
array([[53057, 1522],
       [1325, 4096]])
```



Compromiso Precision-Recall

```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
>>> precision_score(y_train_5, y_train_pred)
0.7290850836596654
>>> recall_score(y_train_5, y_train_pred)
0.7555801512636044
```

```
>>> y_scores =
    sgd_clf.decision_function([some_digit])
>>> y_scores
array([2412.53175101])
>>> threshold = 0
>>> y_some_digit_pred = (y_scores > threshold)
array([True])
>>> threshold = 8000
>>> y_some_digit_pred = (y_scores > threshold)
array([False])
```



Curva Precision vs Recall

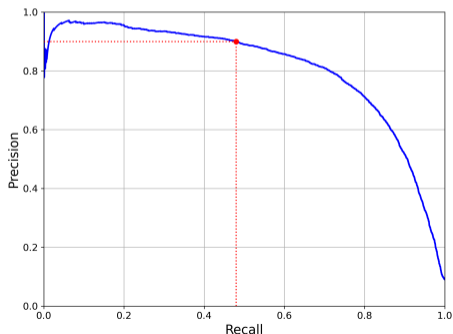
```
y_train_pred_90 = (y_scores >=
    thresholds[np.argmax(precisions >= 0.90)])
```

```
>>> precision_score(y_train_5, y_train_pred_90)
```

```
0.9000380083618396
```

```
>>> recall_score(y_train_5, y_train_pred_90)
```

```
0.4368197749492714
```



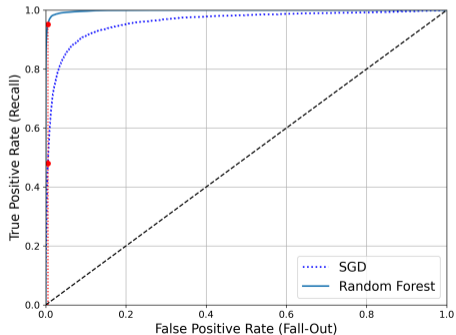
Curva ROC

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_train_5, y_scores)
```

```
from sklearn.metrics import roc_auc_score
```

```
>>> roc_auc_score(y_train_5, y_scores)
```

```
0.9611778893101814
```



Clasificación multi clase

5 0 4 1 9 2 1 3 1 4
3 5 3 6 1 7 2 8 6 9
4 0 9 1 1 2 4 3 2 7
3 8 6 9 0 5 6 0 7 6
1 8 7 9 3 9 8 5 9 3
3 0 7 4 9 8 0 9 4 1
4 4 6 0 4 5 6 1 0 0
1 7 1 6 3 0 2 1 1 7
8 0 2 6 7 8 3 9 0 4
6 7 4 6 8 0 7 8 3 1

MNIST dataset

```
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train, y_train)

>>> some_digit = X[0]
>>> sgd_clf.predict([some_digit])
array([5], dtype=uint8)

>>> cross_val_score(sgd_clf, X_train, y_train,
                    cv=3, scoring="accuracy")
array([0.8489802 , 0.87129356, 0.86988048])

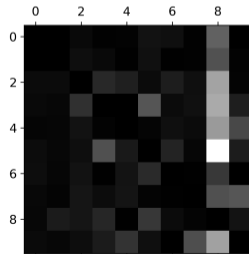
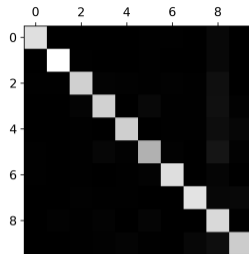
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler()
>>> X_train_scaled = scaler.fit_transform(X_train)
>>> cross_val_score(sgd_clf, X_train_scaled, y_train,
                    cv=3, scoring="accuracy")
array([0.89707059, 0.8960948 , 0.90693604])
```


Análisis de error

```
y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
conf_mx = confusion_matrix(y_train, y_train_pred)
conf_mx
array([[5577,    0,   22,    5,    8,   43,   36,    6,  225,    1],
       [    0, 6400,   37,   24,    4,   44,    4,    7,  212,   10],
       [   27,   27, 5220,   92,   73,   27,   67,   36,  378,   11],
       [   22,   17,  117, 5227,    2,  203,   27,   40,  403,   73],
       [   12,   14,   41,    9, 5182,   12,   34,   27,  347,  164],
       [   27,   15,   30,  168,   53, 4444,   75,   14,  535,   60],
       [   30,   15,   42,    3,   44,   97, 5552,    3,  131,    1],
       [   21,   10,   51,   30,   49,   12,    3, 5684,  195,  210],
       [   17,   63,   48,   86,    3,  126,   25,   10, 5429,   44],
       [   25,   18,   30,   64,  118,   36,    1,  179,  371, 5107]])
```

```
row_sums = conf_mx.sum(axis=1, keepdims=True)
norm_conf_mx = conf_mx / row_sums
```

```
np.fill_diagonal(norm_conf_mx, 0)
plt.matshow(norm_conf_mx, cmap=plt.cm.gray)
plt.show()
```



Análisis de error

```
cl_a, cl_b = 3, 5
X_aa = X_train[(y_train == cl_a) & (y_train_pred == cl_a)]
X_ab = X_train[(y_train == cl_a) & (y_train_pred == cl_b)]
X_ba = X_train[(y_train == cl_b) & (y_train_pred == cl_a)]
X_bb = X_train[(y_train == cl_b) & (y_train_pred == cl_b)]
```

```
plt.figure(figsize=(8,8))
plt.subplot(221); plot_digits(X_aa[:25], images_per_row=5)
plt.subplot(222); plot_digits(X_ab[:25], images_per_row=5)
plt.subplot(223); plot_digits(X_ba[:25], images_per_row=5)
plt.subplot(224); plot_digits(X_bb[:25], images_per_row=5)
plt.show()
```



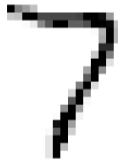
Ejercicio

En el notebook del capítulo 3 del libro* estudiar en qué consiste:

- clasificación multi etiqueta (multilabel classification)
- clasificación multi salida (multioutput classification)



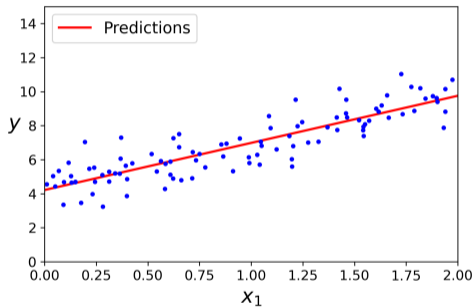
training example



denoised example

* A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition*. O'Reilly Media, Inc., 2022

Regresión lineal

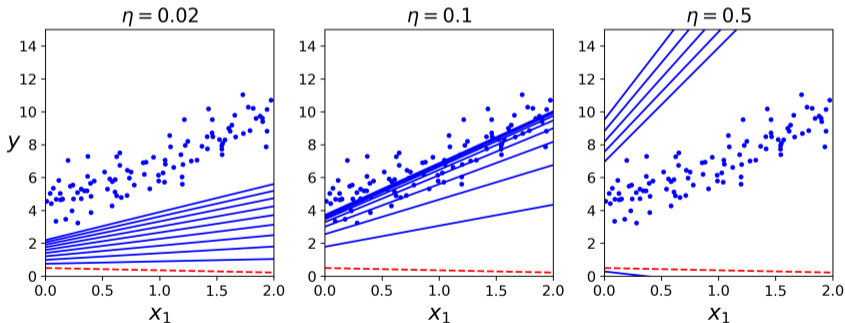


```
from sklearn.linear_model import LinearRegression  
  
lin_reg = LinearRegression()  
lin_reg.fit(X, y)
```

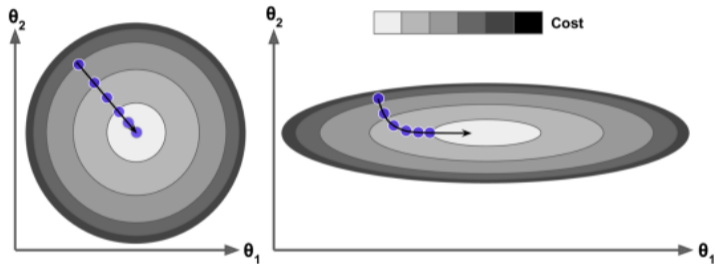
Gradiente descendente

```
from sklearn.linear_model import SGDRegressor
```

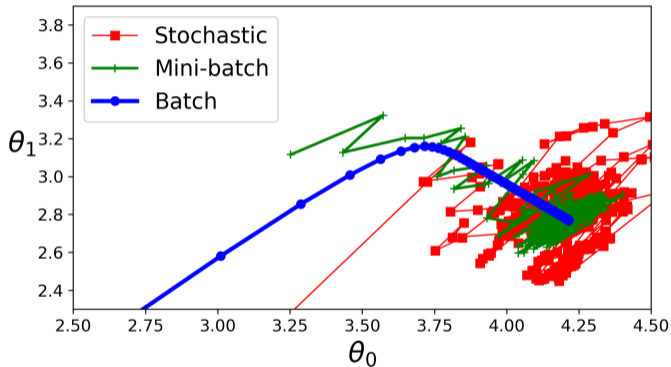
```
sgd_reg = SGDRegressor(max_iter=1000, tol=1e-3, penalty=None, eta0=0.1, random_state=42)  
sgd_reg.fit(X, y.ravel())
```



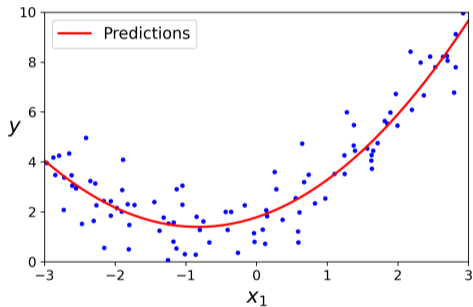
Gradiente descendente



Gradiente descendente



Regresión polinómica

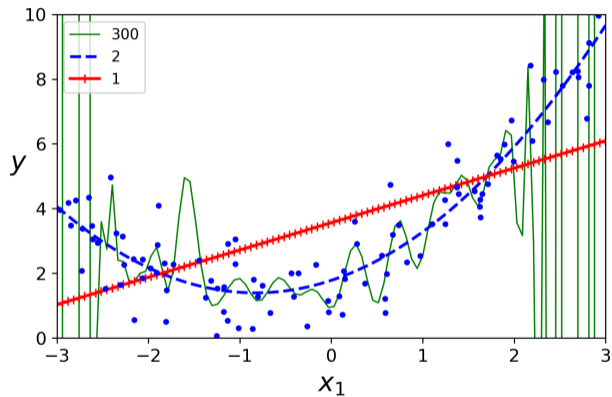


```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=2,
                                   include_bias=False)

X_poly = poly_features.fit_transform(X)

lin_reg = LinearRegression()
lin_reg.fit(X_poly, y)
```


Regresión polinómica

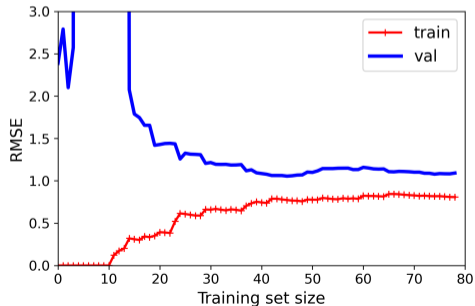
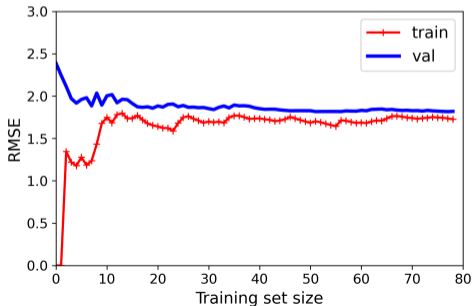


Curvas de aprendizaje

```
from sklearn.pipeline import Pipeline
```

```
polynomial_regression = Pipeline([  
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),  
    ("lin_reg", LinearRegression()),  
])
```

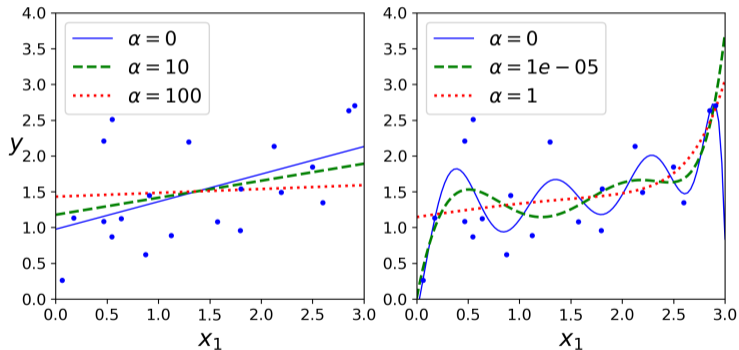
```
plot_learning_curves(polynomial_regression, X, y)
```



Regresión de Ridge

$$J(\theta) = \text{MSE}(\theta) + \alpha \|\theta\|_2^2$$

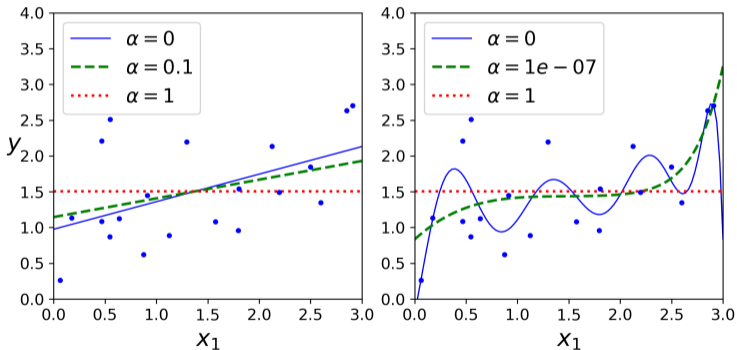
```
sgd_reg = SGDRegressor(penalty="l2")  
sgd_reg.fit(X, y.ravel())
```



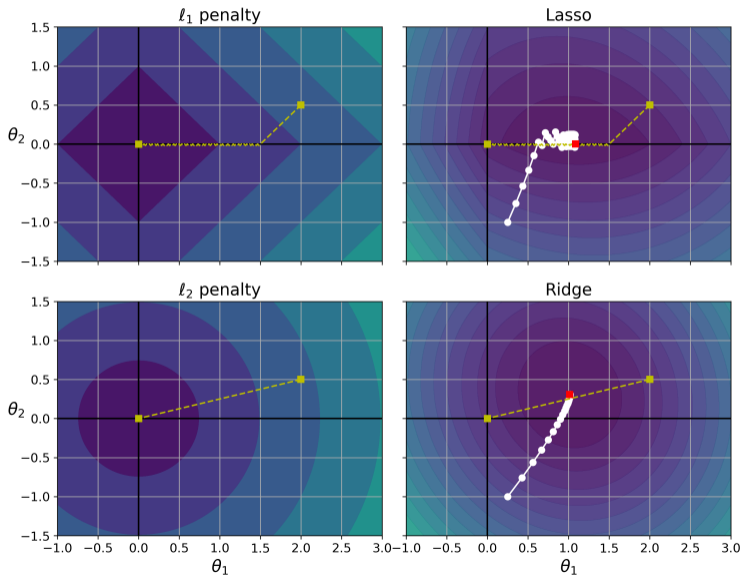
Regresión Lasso

$$J(\theta) = \text{MSE}(\theta) + \alpha \|\theta\|_1$$

```
sgd_reg = SGDRegressor(penalty="l1")  
sgd_reg.fit(X, y.ravel())
```



Lasso vs Ridge

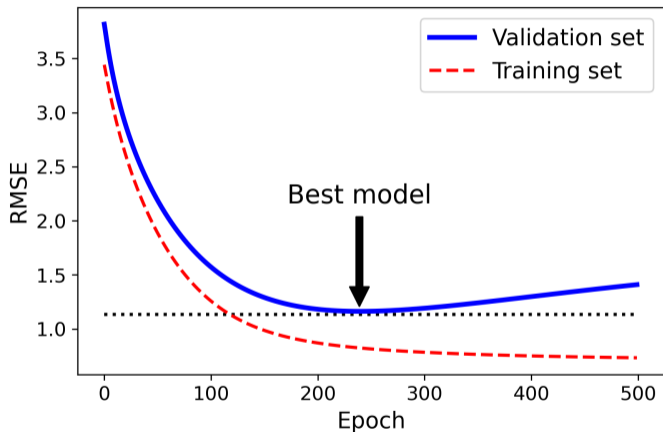


Elastic Net

$$J(\theta) = \text{MSE}(\theta) + r \alpha \|\theta\|_1 + (1 - r) \alpha \|\theta\|_2^2$$

```
from sklearn.linear_model import ElasticNet
elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5, random_state=42)
elastic_net.fit(X, y)
elastic_net.predict([[1.5]])
```

Early stopping



Referencias



A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition*. O'Reilly Media, Inc., 2022.