

# Introducción a la programación para Ciencia de Datos

Prof. Miguel García Silvente

Prof. Miguel García Silvente

1

## Tema 3 Aprendizaje automático

Prof. Miguel García Silvente

2

# Aprendizaje automático

Paquetes y herramientas:

- Scikit-learn
- Pandas
- Orange
- MLpy
- MDP
- PyBrain

Prof. Miguel García Silvente

3

## Paquete Scikit-learn

- Python Machine Learning
- Se incluye con `import sklearn`
- Aprendizaje supervisado: ejemplos etiquetados.
  - Clasificación (binaria o multiclas).
  - Regresión.
- Aprendizaje no supervisado.
  - Clustering

Prof. Miguel García Silvente

4

# Ejemplo: Iris Dataset

Repositorio UC Irvine (UCI) Machine Learning

Clasifica los tipos de flores usando los siguientes rasgos:

- Longitud del sépalo.
- Ancho del sépalo.
- Longitud del pétalo.
- Ancho del pétalo.



## Conjunto de datos de ejemplo (I)

Usando el tipo de dato pandas DataFrame, y con seaborn:

```
import seaborn as sns  
iris = sns.load_dataset('iris')  
iris.head()  
  
  sepal_length  sepal_width  petal_length  petal_width species  
0         5.1         3.5          1.4         0.2    setosa  
1         4.9         3.0          1.4         0.2    setosa  
2         4.7         3.2          1.3         0.2    setosa  
3         4.6         3.1          1.5         0.2    setosa  
4         5.0         3.6          1.4         0.2    setosa
```

# Conjunto de datos de ejemplo (II)

- size: número de ejemplos.

- iris.columns

```
Index(['sepal_length', 'sepal_width',
       'petal_length', 'petal_width',
       'species'], dtype='object')
```

```
import seaborn as sns
```

```
sns.set()
```

```
sns.pairplot(iris, hue='species', size=1.5);
```

Prof. Miguel García Silvente

7

## Vector de rasgos y objetivo

```
X_iris = iris.drop('species', axis=1)
```

```
X_iris.shape
```

(150, 4)

```
y_iris = iris['species']
```

```
y_iris.shape
```

(150,)

Prof. Miguel García Silvente

8

# Conjunto de datos de ejemplo (III)

- Dentro del paquete sklearn también hay datasets:
- ```
from sklearn import datasets  
iris = datasets.load_iris()  
digits = datasets.load_digits()
```
- Un dataset internamente es un diccionario que tiene datos y metadatos
  - Datasets externos

<http://scikit-learn.org/stable/datasets/index.html#external-datasets>

Prof. Miguel García Silvente

9

## Data del dataset

Un array de n\_muestras x n\_rasgos

```
print(digits.data)  
[[ 0.  0.  5. ...,  0.  0.  0.]  
 [ 0.  0.  0. ..., 10.  0.  0.]  
 [ 0.  0.  0. ..., 16.  9.  0.]  
 ...,  
 [ 0.  0.  1. ...,  6.  0.  0.]  
 [ 0.  0.  2. ..., 12.  0.  0.]  
 [ 0.  0. 10. ..., 12.  1.  0.]]
```

<http://scikit-learn.org/stable/tutorial/basic/tutorial.html>

Prof. Miguel García Silvente

10

# Target del dataset

```
print(digits.target)
array([0, 1, 2, ..., 8, 9, 8])

>>> digits.images[0]
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

Prof. Miguel García Silvente

11

# Descargar csv

```
import pandas as pd
```

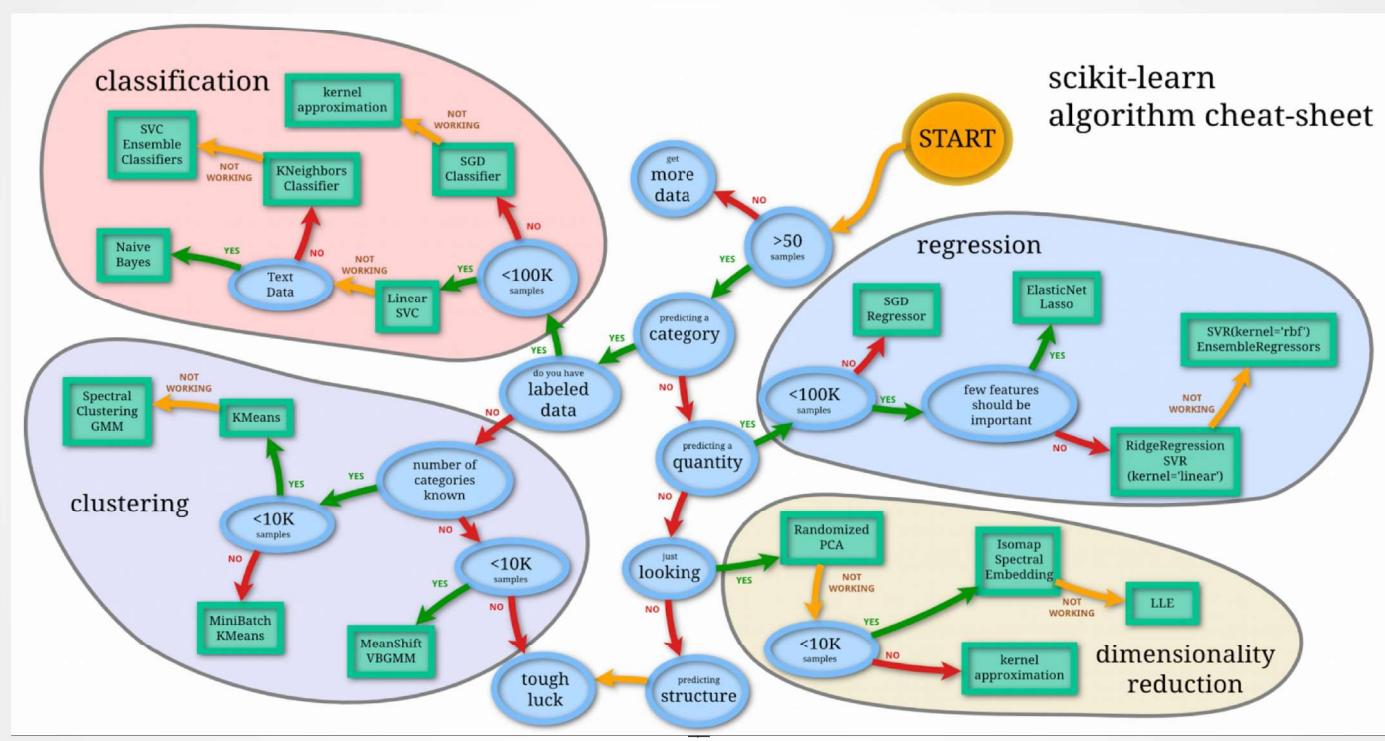
```
# Leer datos con `read_csv()`
digits =
pd.read_csv("http://archive.ics.uci.edu/ml/machine-
learning-databases/optdigits/optdigits.tra", header=None)

print(digits)
```

Prof. Miguel García Silvente

12

# Algoritmos de scikit-learn



Prof. Miguel García Silvente

13

## Ajuste y predicción

- Un estimador para clasificación es un objeto con los métodos:
  - `fit(X, y)`
  - `predict(T)`
- Primero debemos crear el clasificador.  
Ejemplo: `sklearn.svm.SVC`  

```
from sklearn import svm  
clasificador = svm.SVC(gamma=0.001, C=100.)
```

Prof. Miguel García Silvente

14

# Algoritmo

1. Cargar datos para el entrenamiento.
2. Número de rasgos alto → Reducción dimensional.
3. Elección del algoritmo de ajuste y ajuste.
4. Validación del ajuste con datos de ajuste y datos de test. Validación cruzada.
5. Uso del algoritmo ajustado para “predecir”. Se pueden estimar los parámetros.
6. Guardar el ajuste para poder usarlo posteriormente.

Prof. Miguel García Silvente

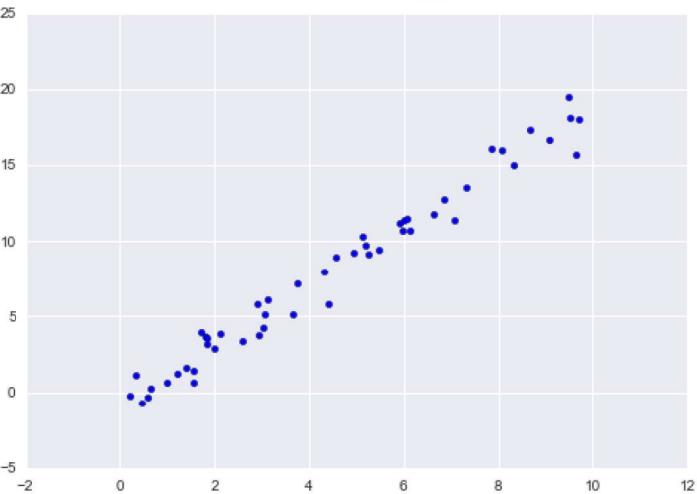
15

## Ejemplo de aprendizaje (I)

Creamos los datos de forma aleatoria

```
import matplotlib.pyplot as plt  
import numpy as np
```

```
rng = np.random.RandomState(42)  
x = 10 * rng.rand(50)  
y = 2 * x - 1 + rng.randn(50)  
plt.scatter(x, y);
```



Prof. Migue

# Ejemplo de aprendizaje (II)

Usamos regresión lineal:

```
from sklearn.linear_model import LinearRegression  
model = LinearRegression(fit_intercept=True)  
X = x[:, np.newaxis]  
X.shape  
(50, 1)  
model.fit(X, y)  
model.coef_  
array([ 1.9776566])  
model.intercept_  
-0.90331072553111635
```

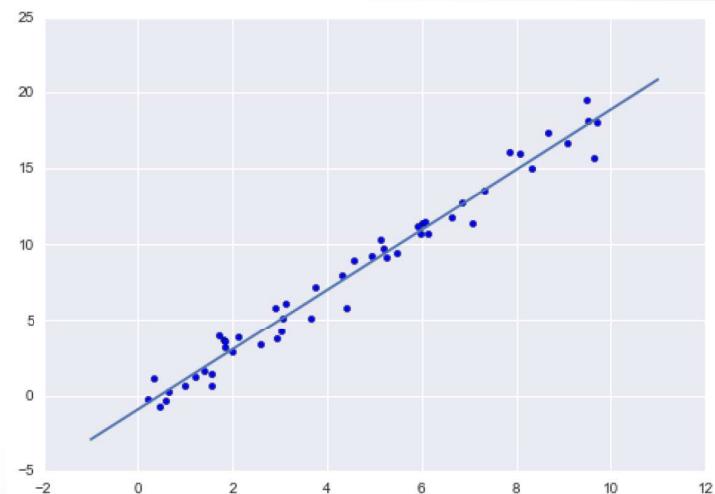
Prof. Miguel García Silvente

17

# Ejemplo de aprendizaje (III)

Predicción de datos

```
xfit = np.linspace(-1, 11)  
Xfit = xfit[:, np.newaxis]  
yfit = model.predict(Xfit)  
plt.scatter(x, y)  
plt.plot(xfit, yfit);
```



Prof. Miguel García Silvente

18

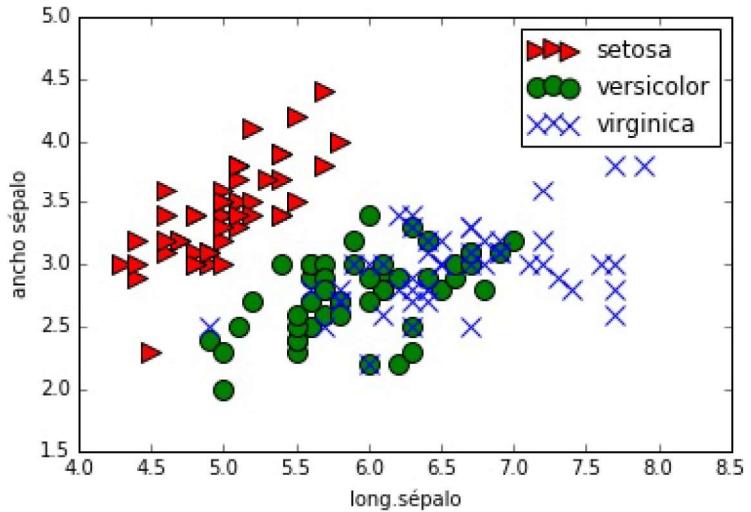
# Visualización de datos (iris)

```
from matplotlib import pyplot as plt
from sklearn.datasets import load_iris
import numpy as np

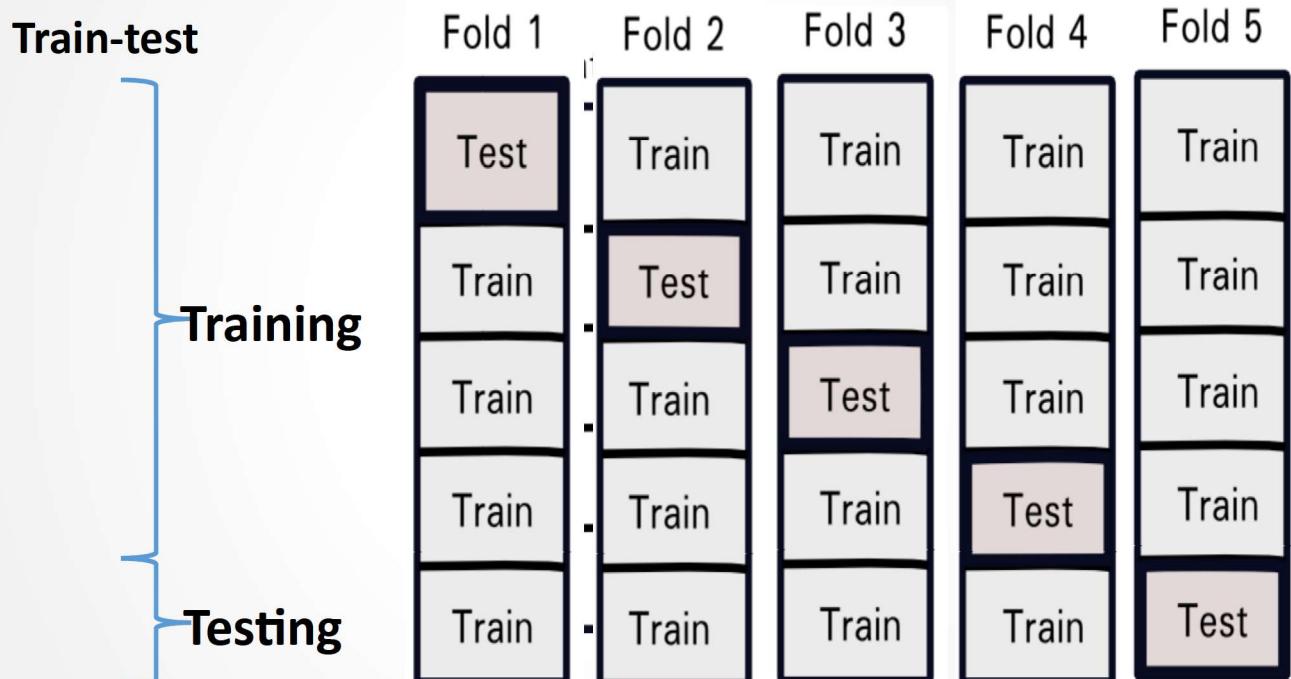
data = load_iris()
features = data['data']
feature_names = data['feature_names']
target = data['target']

cad = []
cad2 = []

for t,marker, c in zip(range(3), ">ox", "rgb"
    aux = plt.scatter(features[target == t], C
        features[target == t, 1],
        marker = marker,
        c=c,
        s=100)
    cad.append(aux)
    cad2.append(data.target_names[t])
plt.legend(cad, cad2, ncol=1, loc='upper right', )
plt.show()
```



## Validación cruzada (I)



# Validación cruzada (II)

- La exactitud del ajuste no debe “depender de la suerte”  
`kf = KFold(n=len(binary_target), n_folds=5, shuffle=True)`
- En kf tendríamos las n\_folds divisiones  
`for tr, tst in kf:`  
 `tr_features = features[tr, :]`  
 `tr_target = binary_target[tr]`  
 `tst_features = features[tst, :]`  
 `tst_target = binary_target[tst]`
- Nunca se deben usar datos de entrenamiento para test
- También se puede usar  
`Xtrain, Xtest, ytrain, ytest = train_test_split(X_iris,`  
`y_iris,random_state=1)`

## Ejemplo de aprendizaje: iris (I)

```
from sklearn.cross_validation import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X_iris,
y_iris,random_state=1)
from sklearn.naive_bayes import GaussianNB # 1. escoger
modelo
model = GaussianNB()                      # 2. instanciar modelo
model.fit(Xtrain, ytrain)                  # 3. ajustar modelo a
datos
y_model = model.predict(Xtest)             # 4. predecir nuevos datos
```

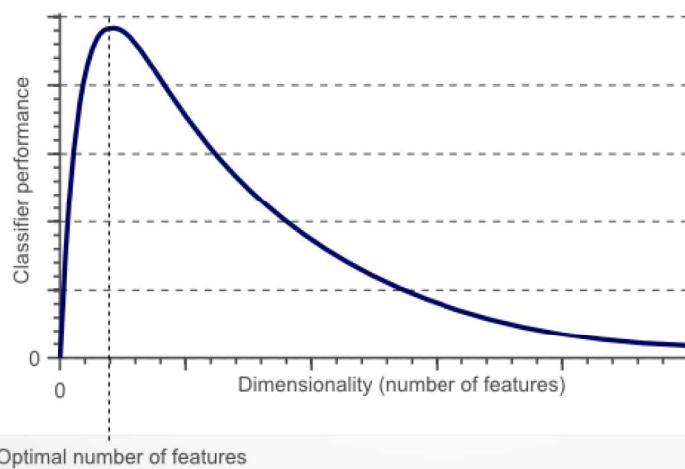
## Ejemplo de aprendizaje: iris (II)

Por último, `accuracy_score` permite ver la fracción de predicciones con un valor verdadero:

```
from sklearn.metrics import accuracy_score  
accuracy_score(ytest, y_model)  
0.97368421052631582
```

## La maldición de la dimensionalidad

- Si el número de variables no es suficiente: no es posible conseguir una calidad deseable.
- Si el número es demasiado alto: puede ocurrir lo mismo



# Número de rasgos

- Dependerá del problema y del tipo de clasificador.
- Si tienden a sobreajustar se deben usar relativamente pocos rasgos. Ej: redes neuronales, kNN, árboles de decisión.
- Por el contrario, si el método generaliza bien, se deben usar muchos rasgos. Ej: naive Bayes, clasificadores lineales.
- ¿Cómo seleccionar el número óptimo?  
[https://en.wikipedia.org/wiki/Feature\\_selection](https://en.wikipedia.org/wiki/Feature_selection)

# Reducción de dimensionalidad

- Selección de rasgos.
- Extracción de rasgos.
  - PCA (Análisis de componentes principales)
  - Kernel PCA
  - Kernel PCA basada en grafos
  - LDA (Análisis discriminante lineal)
  - GDA (Análisis discriminante generalizado)

## PCA (I)

- Se puede definir un número de componentes principales con las que quedarnos.

```
from sklearn.decomposition import PCA  
pca = PCA(n_components=n_components,  
           svd_solver='randomized', whiten=True).  
pca = pca.fit(X_train)
```

- Se aplica a los datos antes de usar el clasificador

```
X_train_pca = pca.transform(X_train)  
X_test_pca = pca.transform(X_test)
```

## PCA (II)

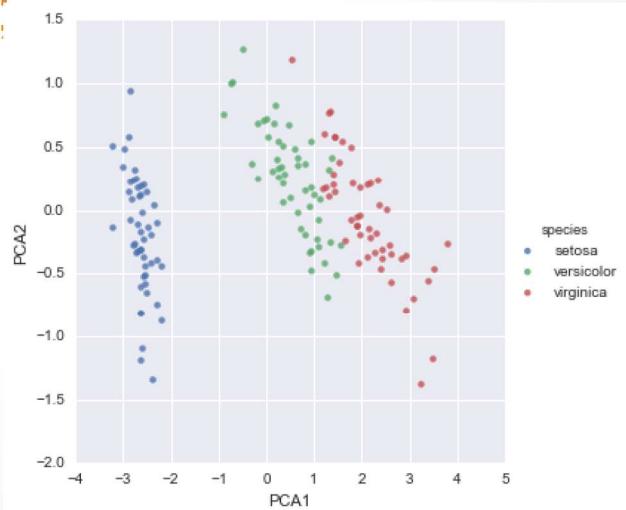
```
from sklearn.decomposition import PCA #  
1.Escoger el modelo  
model_pca = PCA(n_components=2) #  
2.Instanciar el modelo con hiperparámetros  
model_pca.fit(X_iris) # 3. Ajustar los datos, y no  
se incluye  
X_2D = model_pca.transform(X_iris) #  
4.Transforma los datos a 2D
```

# PCA (III)

```
iris['PCA1'] = X_2D[:, 0]
```

```
iris['PCA2'] = X_2D[:, 1]
```

```
sns.lmplot("PCA1", "PCA2", hue='species',  
           data=iris, fit_reg=False);
```



# Regresión

- Normalizar datos

```
features -= np.mean(features, axis=0)
```

```
features /= np.std(features, axis=0)
```

- División en entrenamiento (tr) y test (tst)

```
tr_features = features[tr, :]
```

```
tr_target = binary_target[tr]
```

```
tst_features = features[tst, :]
```

```
tst_target = binary_target[tst]
```

#Etiquetado binario

```
is_versicolor = target == 1
```

```
binary_target = np.zeros(len(target))
```

```
binary_target[is_versicolor] = 1
```

- Ajuste

```
model = LogisticRegression()
```

```
model.fit(tr_features, tr_target)
```

- Exactitud del ajuste

```
tr_accuracy = np.mean(model.predict(tr_features) == tr_target)
```

```
tst_accuracy = np.mean(model.predict(tst_features) == tst_target)
```

# SVM

- Para usar clasificación con SVM

```
from sklearn.svm import SVC
```

...

```
model = SVC()
```

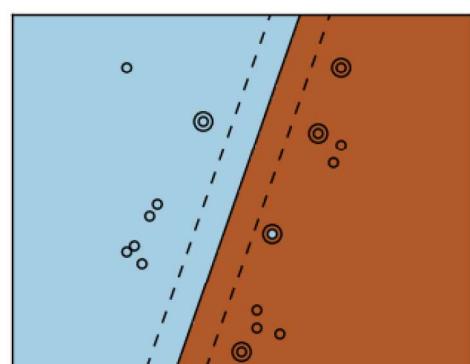
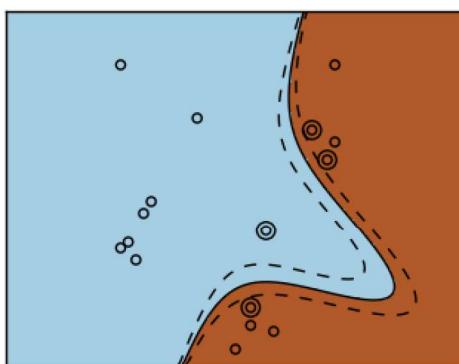
```
model.fit(tr_features, tr_target)
```

- Tiene varios parámetros (qué valores usamos?):

- **C**: parámetro de penalización para puntos “no separables”.
- **Kernel**: rbf, poly, linear.
- **Degree**: para el kernel poly.
- **Gamma**: para el kernel rbf.
- ...

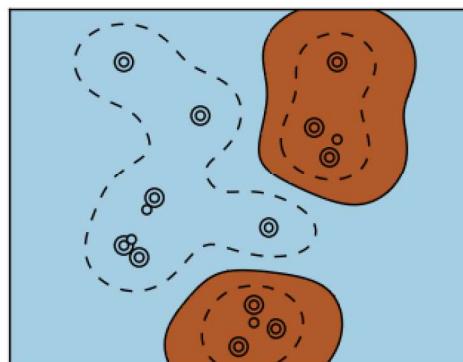
## Kernel de SVM (I)

```
svc = svm.SVC(kernel='linear')
```



```
svc = svm.SVC(kernel='poly', degree=3)  
# degree: grado del polinomio
```

# Kernel de SVM (II)



RBF kernel (Radial Basis Function)

svc = svm.SVC(kernel='rbf')

# gamma: inversa del tamaño del radio del kernel

Prof. Miguel García Silvente

33

## Ejemplo: reconocer dígitos (I)

```
from sklearn.datasets import load_digits  
digits = load_digits()  
digits.images.shape  
(1797, 8, 8)
```

Prof. Miguel García Silvente

34

## Ejemplo: reconocer dígitos (II)

```
import matplotlib.pyplot as plt

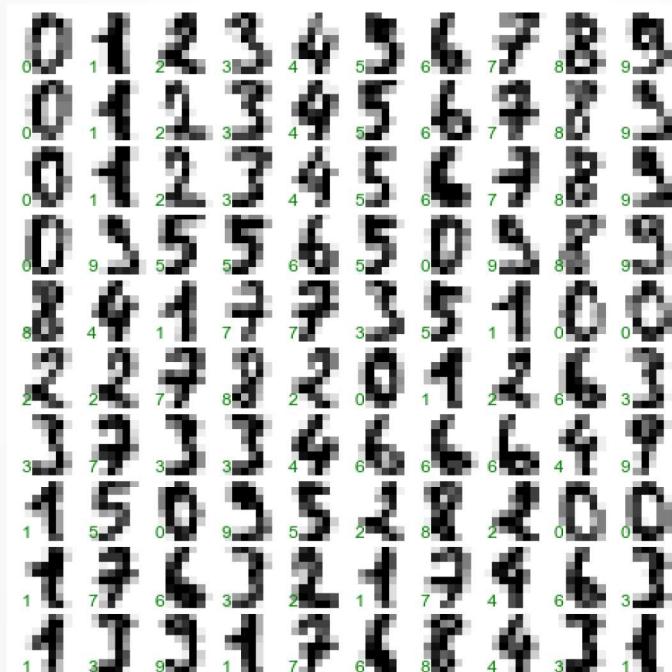
fig, axes = plt.subplots(10, 10, figsize=(8, 8),
                       subplot_kw={'xticks':[], 'yticks':[]},
                       gridspec_kw=dict(hspace=0.1, wspace=0.1))

for i, ax in enumerate(axes.flat):
    ax.imshow(digits.images[i], cmap='binary', interpolation='nearest')
    ax.text(0.05, 0.05, str(digits.target[i]),
            transform=ax.transAxes, color='green')
```

Prof. Miguel García Silvente

35

## Ejemplo: reconocer dígitos (III)



Prof. Miguel García Silvente

36

## Ejemplo: reconocer dígitos (IV)

```
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y,  
random_state=0)
```

```
from sklearn.naive_bayes import GaussianNB  
model = GaussianNB()  
model.fit(Xtrain, ytrain)  
y_model = model.predict(Xtest)  
from sklearn.metrics import accuracy_score  
accuracy_score(ytest, y_model)  
0.8333333333333337
```

Prof. Miguel García Silvente

37

## Ejemplo: reconocer dígitos (V)

```
from sklearn.metrics import confusion_matrix
```

```
mat = confusion_matrix(ytest, y_model)
```

```
sns.heatmap(mat, square=True, annot=True,  
cbar=False)  
plt.xlabel('predicted value')  
plt.ylabel('true value');
```

Prof. Miguel García Silvente

38

# Ejemplo: reconocer dígitos (VI)

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
|---|----|----|----|----|----|----|----|----|----|----|
| 0 | 37 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 1 | 0  | 39 | 0  | 0  | 0  | 0  | 0  | 0  | 4  | 0  |
| 2 | 0  | 7  | 20 | 2  | 0  | 0  | 0  | 0  | 15 | 0  |
| 3 | 0  | 0  | 0  | 39 | 0  | 0  | 0  | 1  | 5  | 0  |
| 4 | 0  | 1  | 0  | 0  | 31 | 0  | 0  | 6  | 0  | 0  |
| 5 | 0  | 1  | 0  | 1  | 0  | 43 | 0  | 3  | 0  | 0  |
| 6 | 0  | 0  | 1  | 0  | 0  | 0  | 51 | 0  | 0  | 0  |
| 7 | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 47 | 0  | 0  |
| 8 | 0  | 6  | 0  | 1  | 0  | 1  | 0  | 2  | 38 | 0  |
| 9 | 0  | 2  | 0  | 4  | 1  | 0  | 0  | 3  | 7  | 30 |

Prof. Miguel García Silvente

39

## Ajuste de los datos

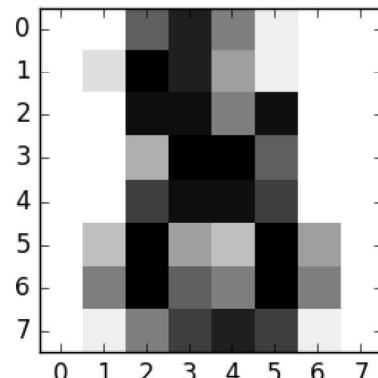
```
from sklearn import svm  
clf = svm.SVC(gamma=0.001, C=100.)  
clf.fit(digits.data[:-1], digits.target[:-1])  
SVC(C=100.0, cache_size=200, class_weight=None,  
coef0=0.0, decision_function_shape=None,  
degree=3, gamma=0.001, kernel='rbf', max_iter=-1,  
probability=False, random_state=None,  
shrinking=True, tol=0.001, verbose=False)
```

Prof. Miguel García Silvente

40

# Ejemplo: clasificar dígitos

```
>>> clf.predict(digits.data[-1:])  
array([8])
```



[http://scikit-learn.org/stable/auto\\_examples/classification/plot\\_digits\\_classification.html#sphx-glr-auto-examples-classification-plot-digits-classification-py](http://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html#sphx-glr-auto-examples-classification-plot-digits-classification-py)

Prof. Miguel García Silvente

41

## Parámetros del modelo (I)

- En el ejemplo, gamma aparece con un valor fijo.
- Los algoritmos tienen parámetros que se pueden estimar:
  - Grid search
  - Random search
- Ejemplo:

```
param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],  
             'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], }  
  
clf = GridSearchCV(SVC(kernel='rbf', class_weight='balanced'),  
                    param_grid)  
  
clf = clf.fit(X_train_pca, y_train)  
  
print("Best estimator found by grid search:")  
print(clf.best_estimator_)
```

Prof. Miguel García Silvente

42

# Parámetros del modelo (II)

```
import numpy as np
from scipy.stats import uniform as sp_rand
from sklearn import datasets
from sklearn.linear_model import Ridge
from sklearn.grid_search import RandomizedSearchCV
# load the diabetes datasets
dataset = datasets.load_diabetes()
# prepare a uniform distribution to sample for the alpha parameter
param_grid = {'alpha': sp_rand()}
# create and fit a ridge regression model, testing random alpha values
model = Ridge()
rsearch = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_iter=100)
rsearch.fit(dataset.data, dataset.target)
print(rsearch)
# summarize the results of the random parameter search
print(rsearch.best_score_)
print(rsearch.best_estimator_.alpha)
```

Prof. Miguel García Silvente

43

# Guardar y recuperar un ajuste

```
import pickle
s = pickle.dumps(clf)
clf2 = pickle.loads(s)
print(clf2.predict(X[0:1]))
array([0])
print(y[0])
0
```

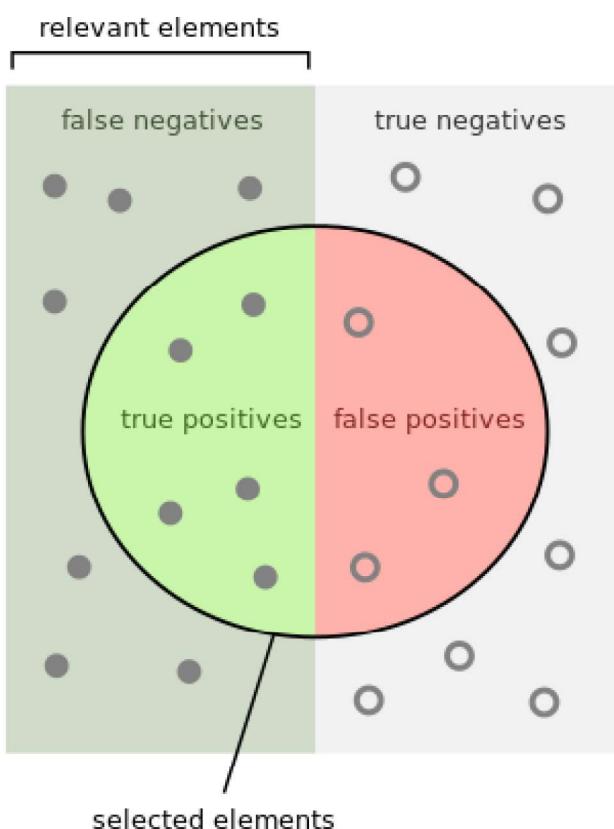
- Es más eficiente usar **joblib** con grandes cantidades de datos

```
from sklearn.externals import joblib
joblib.dump(clf, 'filename.pkl')
clf = joblib.load('filename.pkl')
```

Prof. Miguel García Silvente

44

# Calidad de la clasificación



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

García Silvente

45

## Medidas (I)

F1 score: Es la media harmónica entre precisión y sensibilidad (recall)

$$F1 = \frac{2TP}{2TP + FP + FN}$$

```
import numpy as np
from sklearn.metrics import accuracy_score
y_pred = [0, 2, 1, 3]
y_true = [0, 1, 2, 3]
print(accuracy_score(y_true, y_pred))
print(accuracy_score(y_true, y_pred, normalize=False))
http://scikit-learn.org/stable/modules/model\_evaluation.html#model-evaluation
```

## Medidas (II)

```
from sklearn import svm, datasets
from sklearn.model_selection import cross_val_score

iris = datasets.load_iris()

X, y = iris.data, iris.target
clf = svm.SVC(probability=True, random_state=0)
print(cross_val_score(clf, X, y, scoring='accuracy', cv=5) )
[ 0.96666667  1.          0.96666667  0.96666667  1.          ]
```

Prof. Miguel García Silvente

47

## Medidas (III)

Se pueden calcular varias al mismo tiempo

```
from sklearn.metrics import recall_score
scoring = ['precision_macro', 'recall_macro']
clf = svm.SVC(kernel='linear', C=1, random_state=0)
scores = cross_validate(clf, iris.data, iris.target,
scoring=scoring, cv=5, return_train_score=False)
sorted(scores.keys())
['fit_time', 'score_time', 'test_precision_macro',
'test_recall_macro']
scores['test_recall_macro']
```

Prof. Miguel García Silvente

48

# Clasificación no supervisada (I)

# 1. Escoge el modelo

```
from sklearn.mixture import GMM
```

# 2. Instancia el modelo con hiperparámetros

```
model = GMM(n_components=3, covariance_type='full')
```

# 3. Ajusta los datos. Y no se especifica

```
model.fit(X_iris)
```

# 4. Determina las etiquetas

```
y_gmm = model.predict(X_iris)
```

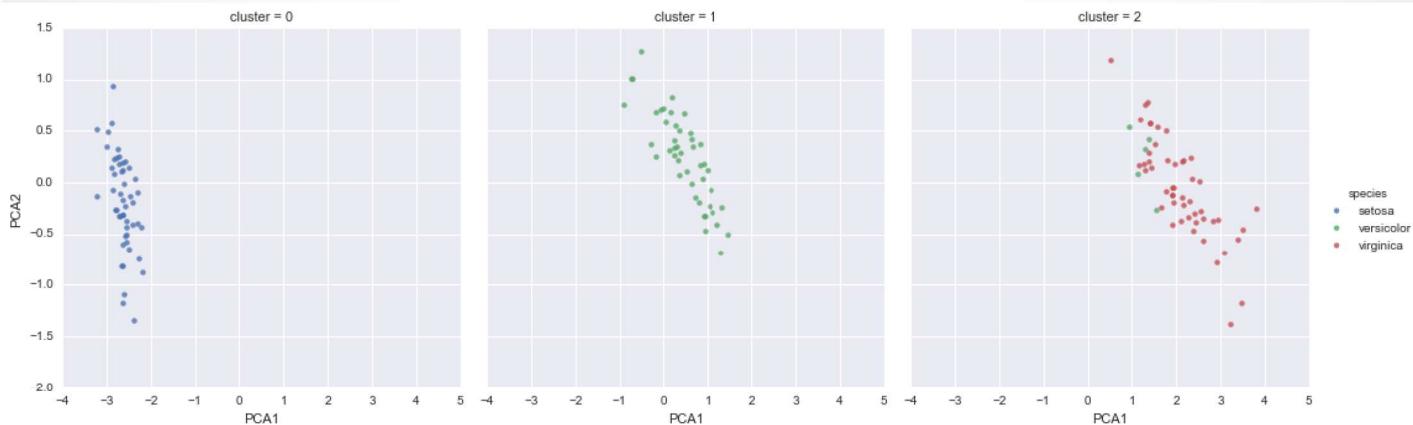
Prof. Miguel García Silvente

49

# Clasificación no supervisada (II)

```
iris['cluster'] = y_gmm
```

```
sns.lmplot("PCA1", "PCA2", data=iris,  
hue='species', col='cluster', fit_reg=False);
```



Prof. Miguel García Silvente

50

# Reducción de dimensiones (I)

Usando isomap

(<https://jakevdp.github.io/PythonDataScienceHandbook/05.10-manifold-learning.html>)

```
from sklearn.manifold import Isomap  
iso = Isomap(n_components=2)  
iso.fit(digits.data)  
data_projected = iso.transform(digits.data)  
data_projected.shape  
(1797, 2)
```

Prof. Miguel García Silvente

51

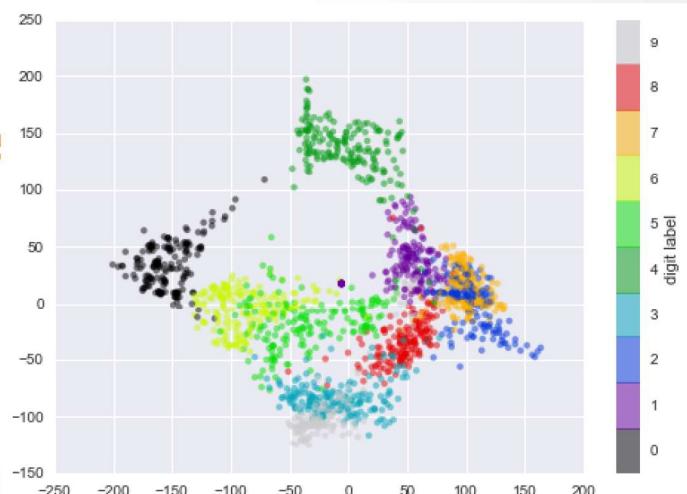
# Reducción de dimensiones (II)

```
plt.scatter(data_projected[:,  
0], data_projected[:, 1],  
c=digits.target, edgecolor='n  
one', alpha=0.5,
```

```
cmap=plt.cm.get_cmap('spe  
ctral', 10))
```

```
plt.colorbar(label='digit  
label', ticks=range(10))
```

```
plt.clim(-0.5, 9.5);
```



Prof. Miguel García Silvente

52

## Resources

- **LIBSVM and LIBLINEAR**

- Chih-Jen Lin, National Taiwan University.
- Simple and easy-to-use support vector machines tool.
- Hsu, C.W., Chang, C.C. and Lin, C.J., 2003. **A practical guide to support vector** classification.  
<https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>

### **SVMlight**

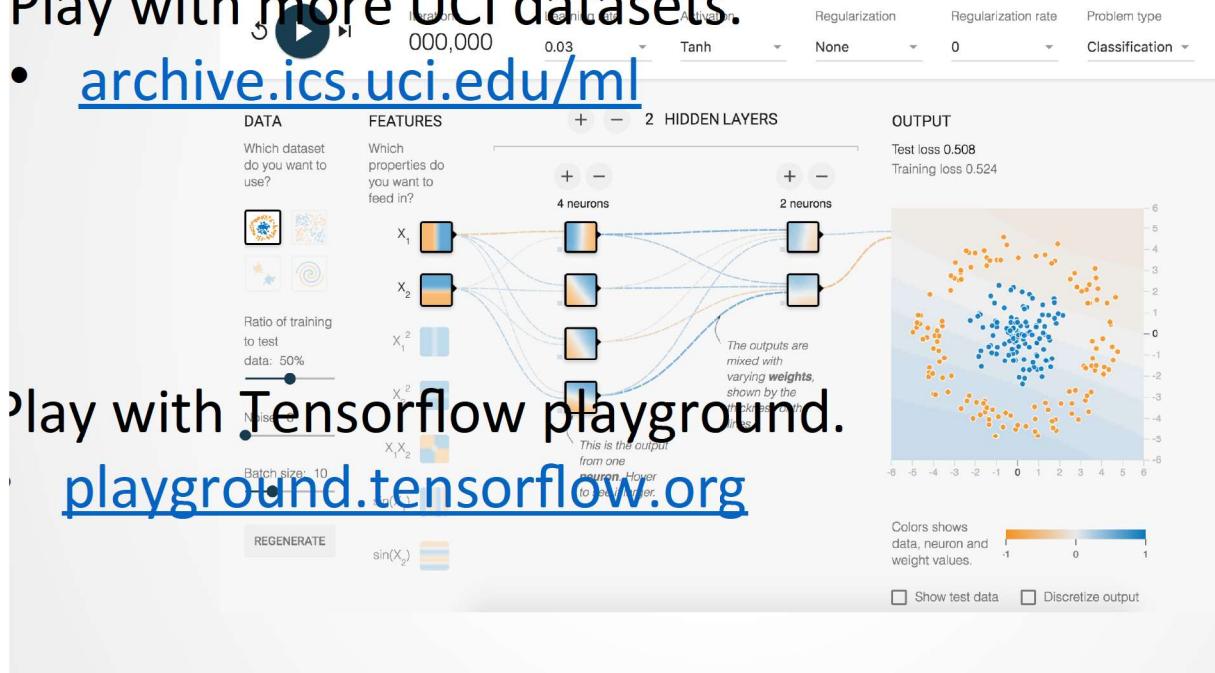
- Thorsten Joachims, Cornell University.
- An implementation of Support Vector Machines (SVMs) in C.

### **Vowpal Wabbit**

- Microsoft Research and (previously) Yahoo! Research
- Fast and scalable tool for learning linear model.
- **Mahout on Hadoop.**
- **MILib on Spark.**
- **Petuum.**

## Play with more UCI datasets.

- [archive.ics.uci.edu/ml](http://archive.ics.uci.edu/ml)



Play with Tensorflow playground.

[playground.tensorflow.org](http://playground.tensorflow.org)

[https://en.wikipedia.org/wiki/Hyperparameter\\_optimization](https://en.wikipedia.org/wiki/Hyperparameter_optimization)

<http://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/>