Supervised learning: predicting an output variable from high-dimensional observations

The problem solved in supervised learning

:ref:`Supervised learning <supervised-learning>` consists in learning the link between two datasets: the observed data `X` and an external variable `y` that we are trying to predict, usually called "target" or "labels". Most often, `y` is a 1D array of length `n_samples`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, **line 10**); ***backlink***

Unknown interpreted text role "ref".

All supervised estimators in scikit-learn implement a `fit(X, y)` method to fit the model and a `predict(X)` method that, given unlabeled observations `X`, returns the predicted labels `y`.

Vocabulary: classification and regression

If the prediction task is to classify the observations in a set of finite labels, in other words to "name" the objects observed, the task is said to be a **classification** task. On the other hand, if the goal is to predict a continuous target variable, it is said to be a **regression** task.

When doing classification in scikit-learn, `y` is a vector of integers or strings.

Note: See the :ref:`Introduction to machine learning with scikit-learn Tutorial <introduction>` for a quick run-through on the basic machine learning vocabulary used within scikit-learn.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, **line 32**); ***backlink***

Unknown interpreted text role "ref".

Nearest neighbor and the curse of dimensionality

Classifying irises:

The iris dataset is a classification task consisting in identifying 3 different types of irises (Setosa, Versicolour, and Virginica) from their petal and sepal length and width:

```
>>> import numpy as np
>>> from sklearn import datasets
>>> iris_X, iris_y = datasets.load_iris(return_X_y=True)
>>> np.unique(iris_y)
array([0, 1, 2])
```

k-Nearest neighbors classifier

The simplest possible classifier is the nearest neighbor: given a new observation `X_test`, find in the training set (i.e. the data used to train the estimator) the observation with the closest feature vector. (Please see the :ref:`Nearest Neighbors section<neighbors>` of the online Scikit-learn documentation for more information about this type of classifier.)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, **line 59**); ***backlink***

Unknown interpreted text role "ref".

Training set and testing set

While experimenting with any learning algorithm, it is important not to test the prediction of an estimator on the data used to fit the estimator as this would not be evaluating the performance of the estimator on **new data**. This is why datasets are often split into *train* and *test* data.

KNN (k nearest neighbors) classification example:

```
>>> # Split iris data in train and test data
>>> # A random permutation, to split the data randomly
>>> np.random.seed(0)
>>> indices = np.random.permutation(len(iris_X))
>>> iris_X_train = iris_X[indices[:-10]]
>>> iris_y_train = iris_y[indices[:-10]]
>>> iris_X_test = iris_X[indices[-10:]]
>>> iris_y_test = iris_y[indices[-10:]]
>>> # Create and fit a nearest-neighbor classifier
>>> from sklearn.neighbors import KNeighborsClassifier
>>> knn = KNeighborsClassifier()
>>> knn.fit(iris_X_train, iris_y_train)
KNeighborsClassifier()
>>> knn.predict(iris_X_test)
array([1, 2, 1, 0, 0, 0, 2, 1, 2, 0])
>>> iris_y_test
array([1, 1, 1, 0, 0, 0, 2, 1, 2, 0])
```

The curse of dimensionality

For an estimator to be effective, you need the distance between neighboring points to be less than some value d, which depends on the problem. In one dimension, this requires on average $n \sim 1/d$ points. In the context of the above k-NN example, if the data is described by just one feature with values ranging from 0 to 1 and with n training observations, then new data will be no further away than $1/n$. Therefore, the nearest neighbor decision rule will be efficient as soon as $1/n$ is small compared to the scale of between-class feature variations.

If the number of features is p, you now require $n \sim 1/d^p$ points. Let's say that we require 10 points in one dimension: now 10^p points are required in p dimensions to pave the [0, 1] space. As p becomes large, the number of training points required for a good estimator grows exponentially.

For example, if each point is just a single number (8 bytes), then an effective k-NN estimator in a paltry $p \sim 20$ dimensions would require more training data than the current estimated size of the entire internet (Â±1000 Exabytes or so).

This is called the curse of dimensionality and is a core problem that machine learning addresses.

Linear model: from regression to sparsity

Diabetes dataset

The diabetes dataset consists of 10 physiological variables (age, sex, weight, blood pressure) measure on 442 patients, and an indication of disease progression after one year:

```
>>> diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
>>> diabetes_X_train = diabetes_X[:-20]
>>> diabetes_X_test  = diabetes_X[-20:]
>>> diabetes_y_train = diabetes_y[:-20]
>>> diabetes_y_test  = diabetes_y[-20:]
```

The task at hand is to predict disease progression from physiological variables.

Linear regression

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 151)

Unknown directive type "currentmodule".

```
.. currentmodule:: sklearn.linear_model
```

:class:`LinearRegression`, in its simplest form, fits a linear model to the data set by adjusting a set of parameters in order to make the sum of the squared residuals of the model as small as possible.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 153); *backlink*

Unknown interpreted text role "class".

Linear models: $y = X\beta + \epsilon$

- X: data
- y: target variable
- β: Coefficients
- ϵ: Observation noise

```
>>> from sklearn import linear_model
>>> regr = linear_model.LinearRegression()
>>> regr.fit(diabetes_X_train, diabetes_y_train)
LinearRegression()
>>> print(regr.coef_) # doctest: +SKIP
[   0.30349955 -237.63931533  510.53060544  327.73698041 -814.13170937
  492.81458798  102.84845219  184.60648906  743.51961675   76.09517222]


>>> # The mean square error
>>> np.mean((regr.predict(diabetes_X_test) - diabetes_y_test)**2)
2004.5...

>>> # Explained variance score: 1 is perfect prediction
>>> # and 0 means that there is no linear relationship
>>> # between X and y.
>>> regr.score(diabetes_X_test, diabetes_y_test)
0.585...
```

Shrinkage

If there are few data points per dimension, noise in the observations induces high variance:

```
>>> X = np.c_[ .5, 1].T
>>> y = [.5, 1]
>>> test = np.c_[ 0, 2].T
>>> regr = linear_model.LinearRegression()

>>> import matplotlib.pyplot as plt
>>> plt.figure()
<...>
>>> np.random.seed(0)
>>> for _ in range(6):
...     this_X = .1 * np.random.normal(size=(2, 1)) + X
...     regr.fit(this_X, y)
...     plt.plot(test, regr.predict(test))
...     plt.scatter(this_X, y, s=3)
LinearRegression...
```

A solution in high-dimensional statistical learning is to *shrink* the regression coefficients to zero: any two randomly chosen set of observations are likely to be uncorrelated. This is called :class:`Ridge` regression:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, **line 222**); ***backlink***

Unknown interpreted text role "class".

```
>>> regr = linear_model.Ridge(alpha=.1)

>>> plt.figure()
<...>
>>> np.random.seed(0)
>>> for _ in range(6):
...     this_X = .1 * np.random.normal(size=(2, 1)) + X
...     regr.fit(this_X, y)
...     plt.plot(test, regr.predict(test))
...     plt.scatter(this_X, y, s=3)
Ridge...
```

This is an example of **bias/variance tradeoff**: the larger the ridge `alpha` parameter, the higher the bias and the lower the variance.

We can choose `alpha` to minimize left out error, this time using the diabetes dataset rather than our synthetic data:

```
>>> alphas = np.logspace(-4, -1, 6)
>>> print([regr.set_params(alpha=alpha)
...             .fit(diabetes_X_train, diabetes_y_train)
...             .score(diabetes_X_test, diabetes_y_test)
...         for alpha in alphas])
[0.585..., 0.585..., 0.5854..., 0.5855..., 0.583..., 0.570...]
```

Note

Capturing in the fitted parameters noise that prevents the model to generalize to new data is called overfitting. The bias introduced by the ridge regression is called a regularization.

Sparsity

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 288)

Unknown directive type "rst-class".

```
.. rst-class:: centered

    **Fitting only features 1 and 2**
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 292)

Unknown directive type "centered".

```
.. centered:: |diabetes_ols_1| |diabetes_ols_3| |diabetes_ols_2|
```

Note

A representation of the full diabetes dataset would involve 11 dimensions (10 feature dimensions and one of the target variable). It is hard to develop an intuition on such representation, but it may be useful to keep in mind that it would be a fairly *empty* space.

We can see that, although feature 2 has a strong coefficient on the full model, it conveys little information on `y` when considered with feature 1.

To improve the conditioning of the problem (i.e. mitigating the :ref:`curse_of_dimensionality`), it would be interesting to select only the informative features and set non-informative ones, like feature 2 to 0. Ridge regression will decrease their contribution, but not set them to zero. Another penalization approach, called :ref:`lasso` (least absolute shrinkage and selection operator), can set some coefficients to zero. Such methods are called **sparse method** and sparsity can be seen as an application of Occam's razor: *prefer simpler models*.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 306); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 306); *backlink*

Unknown interpreted text role "ref".

```
>>> regr = linear_model.Lasso()
>>> scores = [regr.set_params(alpha=alpha)
...               .fit(diabetes_X_train, diabetes_y_train)
...               .score(diabetes_X_test, diabetes_y_test)
...           for alpha in alphas]
>>> best_alpha = alphas[scores.index(max(scores))]
>>> regr.alpha = best_alpha
>>> regr.fit(diabetes_X_train, diabetes_y_train)
Lasso(alpha=0.025118864315095794)
>>> print(regr.coef_)
[   0.         -212.4...   517.2...  313.7... -160.8...
   -0.         -187.1...   69.3...  508.6...   71.8... ]
```

Different algorithms for the same problem

Different algorithms can be used to solve the same mathematical problem. For instance the `Lasso` object in scikit-learn solves the lasso regression problem using a coordinate descent method, that is efficient on large datasets. However, scikit-learn also provides the :class:`LassoLars` object using the *LARS* algorithm, which is very efficient for problems in which the weight vector estimated is very sparse (i.e. problems with very few observations).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main) (doc) (tutorial) (statistical_inference) supervised_learning.rst`, **line 332**); ***backlink***

Unknown interpreted text role "class".

Classification

For classification, as in the labeling iris task, linear regression is not the right approach as it will give too much weight to data far from the decision frontier. A linear approach is to fit a sigmoid function or **logistic** function:

$$y = \text{sigmoid}(X\beta - \text{offset}) + \epsilon = \frac{1}{1 + \exp(-X\beta + \text{offset})} + \epsilon$$

```
>>> log = linear_model.LogisticRegression(C=1e5)
>>> log.fit(iris_X_train, iris_y_train)
LogisticRegression(C=100000.0)
```

This is known as :class:`LogisticRegression`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main) (doc) (tutorial) (statistical_inference) supervised_learning.rst`, **line 368**); ***backlink***

Unknown interpreted text role "class".

Multiclass classification

If you have several classes to predict, an option often used is to fit one-versus-all classifiers and then use a voting heuristic for the final decision.

Shrinkage and sparsity with logistic regression

The `C` parameter controls the amount of regularization in the :class:`LogisticRegression` object: a large value for `C` results in less regularization. `penalty="l2"` gives :ref:`shrinkage` (i.e. non-sparse coefficients), while `penalty="l1"` gives :ref:`sparsity`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main) (doc) (tutorial) (statistical_inference) supervised_learning.rst`, **line 383**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main) (doc) (tutorial) (statistical_inference) supervised_learning.rst`, **line 383**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main) (doc) (tutorial) (statistical_inference) supervised_learning.rst`, **line 383**); ***backlink***

Unknown interpreted text role "ref".

Exercise

Try classifying the digits dataset with nearest neighbors and a linear model. Leave out the last 10% and test prediction performance on these observations.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 396)

Unknown directive type "literalinclude".

```
.. literalinclude:: ../../auto_examples/exercises/plot_digits_classification_exercise.py
    :lines: 15-19
```

A solution can be downloaded :download:`here <../../auto_examples/exercises/plot_digits_classification_exercise.py>`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 399); *backlink*

Unknown interpreted text role "download".

Support vector machines (SVMs)

Linear SVMs

:ref:`svm` belong to the discriminant model family: they try to find a combination of samples to build a plane maximizing the margin between the two classes. Regularization is set by the `C` parameter: a small value for `C` means the margin is calculated using many or all of the observations around the separating line (more regularization); a large value for `C` means the margin is calculated on observations close to the separating line (less regularization).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 409); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 417)

Unknown directive type "currentmodule".

```
.. currentmodule :: sklearn.svm
```

Unregularized SVM

Regularized SVM (default)

Example:

- :ref:`sphx_glr_auto_examples_svm_plot_iris_svc.py`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 431); *backlink*

Unknown interpreted text role "ref".

SVMs can be used in regression --:class:`SVR` (Support Vector Regression)--, or in classification --:class:`SVC` (Support Vector Classification).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 434); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 434); *backlink*

Unknown interpreted text role "class".

```
>>> from sklearn import svm
>>> svc = svm.SVC(kernel='linear')
>>> svc.fit(iris_X_train, iris_y_train)
SVC(kernel='linear')
```

Warning

Normalizing data

For many estimators, including the SVMs, having datasets with unit standard deviation for each feature is important to get good prediction.

Using kernels

Classes are not always linearly separable in feature space. The solution is to build a decision function that is not linear but may be polynomial instead. This is done using the *kernel trick* that can be seen as creating a decision energy by positioning *kernels* on observations:

Linear kernel

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

Polynomial kernel

```
>>> svc = svm.SVC(kernel='poly',
...               degree=3)
>>> # degree: polynomial degree
```

RBF kernel (Radial Basis Function)

```
>>> svc = svm.SVC(kernel='rbf')
>>> # gamma: inverse of size of
>>> # radial kernel
```

Interactive example

See the :ref:`SVM GUI <sphx_glr_auto_examples_applications_svm_gui.py>` to download `svm_gui.py`; add data points of both classes with right and left button, fit the model and change parameters and data.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, line 499); *backlink*

Unknown interpreted text role "ref".

Exercise

Try classifying classes 1 and 2 from the iris dataset with SVMs, with the 2 first features. Leave out 10% of each class and test prediction performance on these observations.

Warning: the classes are ordered, do not leave out the last 10%, you would be testing on only one class.

Hint: You can use the `decision_function` method on a grid to get intuitions.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, **line 516)**

Unknown directive type "literalinclude".

```
.. literalinclude:: ../../auto_examples/exercises/plot_iris_exercise.py
    :lines: 18-23
```

A solution can be downloaded :download:`here <../../auto_examples/exercises/plot_iris_exercise.py>`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\tutorial\statistical_inference\(scikit-learn-main)(doc)(tutorial)(statistical_inference)supervised_learning.rst`, **line 525);** ***backlink***

Unknown interpreted text role "download".