情報工学実験4：データマイニング班

(week 6) 機械学習概観の振り返り

1. （復習）用語
2. （復習）機械学習における問題設定
3. （復習）scikit-learnの使い方
4. （復習）モデルと仮説
5. （復習）問題・アルゴリズム・モデル
6. （復習）過学習
7. （復習）ペナルティ項（正則化）
8. （復習）交差検定

実験ページ: http://ie.u-ryukyu.ac.jp/~tnal/2015/info4/dm/

Terminology

review

- supervised, unsupervised learning
- classification, regression, clustering
- sample
- features, attributes
  - numerical value
  - categorical value
  - true or false
- supervisory signal, teacher, class, label, output data, target variable

- input, output
- training data / training set
- test data / test set
  - open test
  - close test
- model
- parameters
- learn, fit
- predict, estimate
- evaluation

Machine Leaning: the problem setting

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

- In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), is it said to have several attributes or features.

An introduction to machine learning with scikit-learn (1/3)

hg clone ssh://info3dm@shark//home/info3dm/HG/tnal
less sklearn_intro.py

- Loading and an example dataset
  - workon py34
  - (py34) python
  - >>> from sklearn import datasets
  - >>> iris = datasets.load_iris() # datasets.load[tab]
  - >>> print(iris.DESCR)
  - >>> print(iris.data)
  - >>> print(iris.target)
  - >>> print(iris.target_names)

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

An introduction to machine learning with scikit-learn (2/3)

- Learning and predicting
  - >>> from sklearn import svm
  - >>> clf = svm.SVC(gamma=0.001, C=100.)
  - >>> clf.fit(iris.data[:-1], iris.target[:-1])
  - >>> clf.predict(iris.data[-1]]
  - >>> print(iris.target[-1])
  - >>> clf.score(iris.data, iris.target)

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

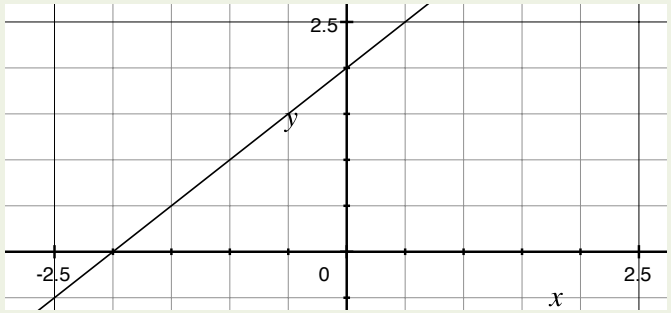
An introduction to machine learning with scikit-learn (3/3)

- Model persistence
  - # save
  - >>> import pickle
  - >>> file = open("PredictiveModel.dump", "wb")
  - >>> pickle.dump(clf, file)
  - >>> file.close()
  - # load
  - >>> file = open("PredictiveModel.dump", "rb")
  - >>> clf2 = pickle.load(file)
  - >>> file.close()
  - >>> clf2.predict(iris.data[-1]]
  - >>> print(iris.target[-1])

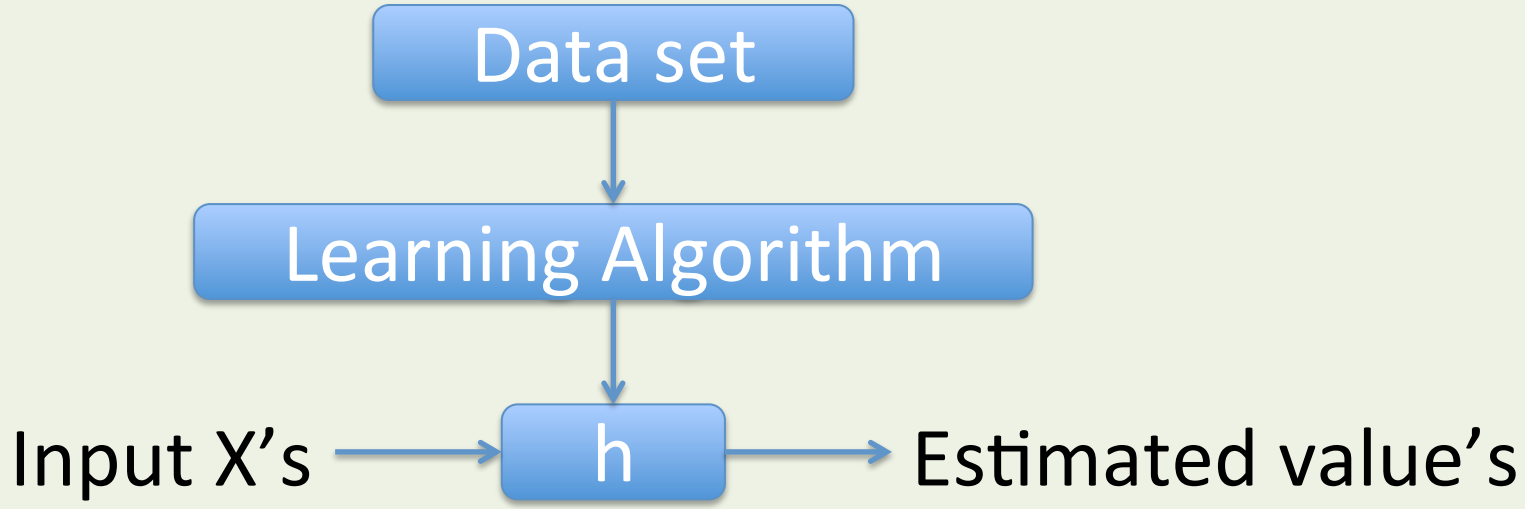
http://scikit-learn.org/stable/tutorial/basic/tutorial.html

Models

- Represent by any formulas with (sometimes one) parameters for the relationship between input X's and output Y's.
  - In machine learning, the formulas called as "hypothesis".
  - E.g., h = a*x + b
    - a, b: parameters
  - Parameterized model.
  - Predictive model. (e.g., a=1, b=2)



Problem <-> Algorithm + Model



Linear Regression Model

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 = \sum \theta_i x_i = \sum \theta_i \Phi_i(x)$$

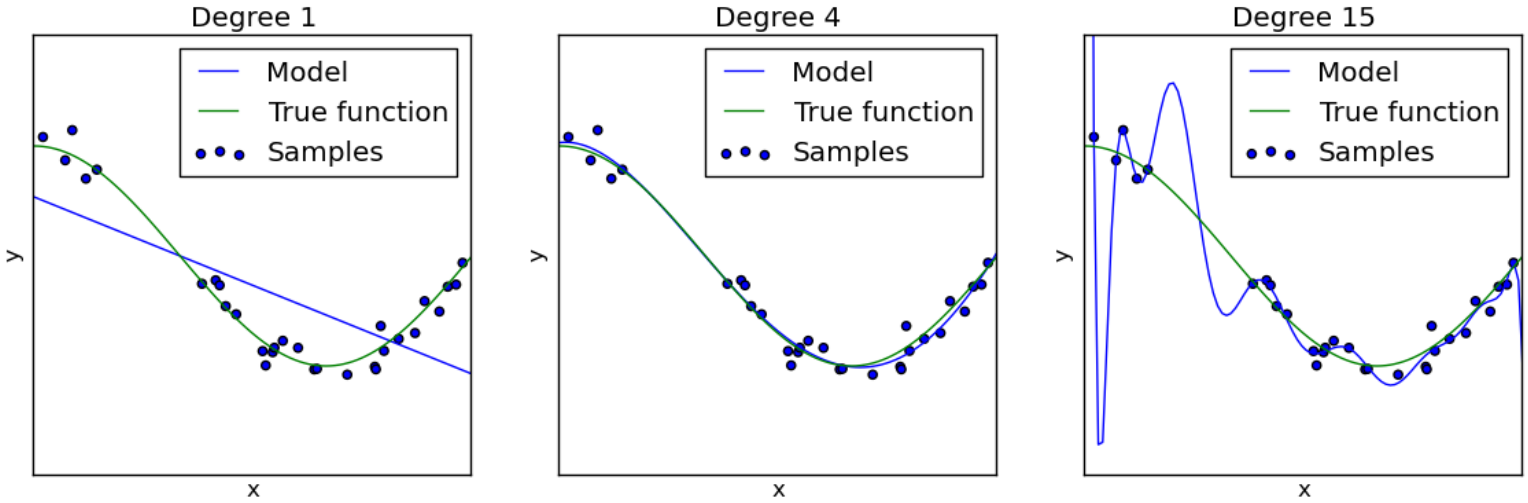
$$h_\theta(x) = \theta_0 + \theta_1 x$$

- How do we prepare a model?
- How do we evaluate the goodness?
- How do we choose the appropriate parameters?

http://scikit-learn.org/stable/auto_examples/plot_underfitting_overfitting.html

Underfitting vs. Overfitting

This example demonstrates the problems of underfitting and overfitting and how we can use linear regression with polynomial features to approximate nonlinear functions. The plot shows the function that we want to approximate, which is a part of the cosine function. In addition, the samples from the real function and the approximations of different models are displayed. The models have polynomial features of different degrees. We can see that a linear function (polynomial with degree 1) is not sufficient to fit the training samples. This is called **underfitting**. A polynomial of degree 4 approximates the true function almost perfectly. However, for higher degrees the model will **overfit** the training data, i.e. it learns the noise of the training data.



a penalty for the parameters (generalization)

- introduce a penalty term to cost function J(θ).
  - sum of squared parameters (L2-norm)

before: $$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Linear Regression

after: $$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2}\|\theta\|^2$$

$$\|\theta\|^2 = \theta_0^2 + \theta_1^2 + \ldots + \theta_M^2$$

Ridge Regression

K-fold cross-validation (K=5)

(step 2) The model.fit() is learned using K-1 folds (4 folds), and the fold left out is used for test.

total dataset

(step 1) divides all the samples in K groups (folds) of equal sizes, if possible.