Topics in Quantitative Finance (FIN 528): Machine Learning for Finance

Jaehyuk Choi

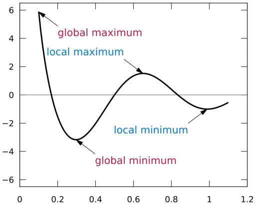
March 9, 2017

Quantitative finance courses in PHBS

- Y1-M3: Stochastic Finance by Jaehyuk Choi **[required for Qfin MA]**
- Y1-M4: Derivative Pricing by Lei (Jack) Sun
- Y2-M1: Applied Stochastic Processes by Jaehyuk CHOI
  Application, Programming, Course project
- Y2-M3: Topics in Quantitative Finance by Jaehyuk CHOI
  Machine Learning for Finance (Mon-Thurs 1:30 PM)
- Y2-M3: Numerical Methods and Analysis by Jake ZHAO (Mon-Thurs 3:30 PM)
- Y2-M3: Bayesian Statistics by Qian CHEN (Mon-Thurs 10:30 AM)

AlphaGo vs Humans (LEE Sedol, KE Jie)

We have to think again about joseki / dìngshì (定石/ 定式) .
In Go (围棋), a joseki is the studied sequences of moves for which the result is considered balanced for both black and white sides.



LEE Sedol afterwards · · ·

- Became more popular and richer
- Perhaps titled as the last human who beat the machine in Go

ML/AI: Rise of the machines?

Probably not!

What and why now?

What is ML?

- Prediction based on data (data into knowledge)
- Extended linear/logistic regression
- Pattern recognition

Why now?

- Abundant Data (Big Data)
- Faster computer (Graphics Processing Unit: GPU)
- Advances in research: Geoffrey Hinton (Google), Yann LeCun (Facebook).

Recent applications of ML

- Automated driving system (Google, Apple, etc)
- Suggestion engine (Amazon, Taobao)
- Cancer diagnosis (IBM Watson)
- Digitizing images (Facebook, Google)
- Shopping without checkout (Amazon Go: big data)

ML in finance?

Prediction

- Asset management / investment
- Trading algorithm (alpha)
- Earnings prediction: e.g., Prediction Valley

Cost cut / labor reduction

- Automated accounting / tax
- Automated analyst report
- Chat-bot (trading and sales)
- Data analytics: e.g., Kensho

Softwares to use

Python

- Anaconda (Python distribution + Environment management)
- Python Language Tutorial
- Sci-Kit Learn
- TensorFlow (wrapped by Keras)

Github.com

- Distributed version control system
- Clone or fork a repository to create a local copy
- `https://github.com/PHBS/2016.M3.TQF-ML` (our course)
- `https://github.com/rasbt/python-machine-learning-book` (**PML**)

Other resources for ML

- Coursera ML course (**CML**) by Andrew Ng (Baidu)
- Stanford CS229 Machine Learning: course notes, student projects, etc
- The Elements of Statistical Learning (**ESL**)
- An Introduction to Statistical Learning (with R) (**ISLR**)
- Pattern Recognition and Machine Learning by Bishop (Microsoft)

Homework

- Install Anaconda (Ver 4.3 Python 3.6)
- Create GitHub account, join PHBS (organization) and 2016.M3.TQF-ML (team).
- Fork the two repositories (the **PML** book and our course)
- Watch the first lecture of **CML**

Notations and conventions: vector and matrix

General rules (guess from the context)

- Scalar (non-bold): x, y, X, Y
- Vector (lowercase bold): $\boldsymbol{x} = (x_i), \boldsymbol{y} = (y_i)$
- Matrix (uppercase bold): $\boldsymbol{X} = (X_{ij})$, $\boldsymbol{Y} = (Y_{ij})$
- The (i, j) component of $\boldsymbol{X}$: X_{ij}
- The i-th row vector of $\boldsymbol{X}$: $\boldsymbol{X}_{i*} = (X_{i1}, X_{i2}, \cdots, X_{ip})^T$
- The j-th column vector of $\boldsymbol{X}$: $\boldsymbol{X}_{*j} = (X_{1j}, X_{2j}, \cdots, X_{Nj})$

Examples

- Dot product: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T \boldsymbol{y}$
- Vector norm: $|\boldsymbol{x}| = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}$
- Matrix multiplication: $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{Y} \to Z_{ij} = \boldsymbol{X}_{i*}\boldsymbol{Y}_{j*}$

Notation and conventions: variables and observations

General rules

- Generic (or representative) variables (uppercase non-bold): X (input), Y (output), G (classification output)
- The predictions: $\hat{Y}$, $\hat{G}$
- X (input) may be p-dimensional (features): X_j ($j \le p$), row vector
- Y (output) may be K-dimensional (responses): Y_k ($k \le K$), row vector.
- The N observations of X or Y is stacked as rows: $\boldsymbol{X}$ ($N \times p$), $\boldsymbol{Y}$ ($N \times K$)
- The i-th observation set: $\boldsymbol{X}_{i*}$ $(1 \times p)$
- All observation of j-th feature X_j: $\boldsymbol{X}_{*j}$ $(N \times 1)$
- $\boldsymbol{X} = (\boldsymbol{X}_{*1} \cdots \boldsymbol{X}_{*p})$ (column-wise concatenation)
- The weight vector: $\boldsymbol{\beta}$ or $\boldsymbol{w}$ used interchangeably.

Simple Linear Regression (Ordinary Least Square)

For scalar predictor (X) and response (Y),

$$Y \approx \beta_0 + \beta_1 X \quad \longrightarrow \quad \hat{\boldsymbol{y}} = \beta_0 + \beta_1 \boldsymbol{x}.$$

For N observations $(x_1, y_1), \cdots, (x_N, y_N)$, the set of $(\hat{\beta}_0, \hat{\beta}_1)$ to minimize the residual sum of squares (RSS):

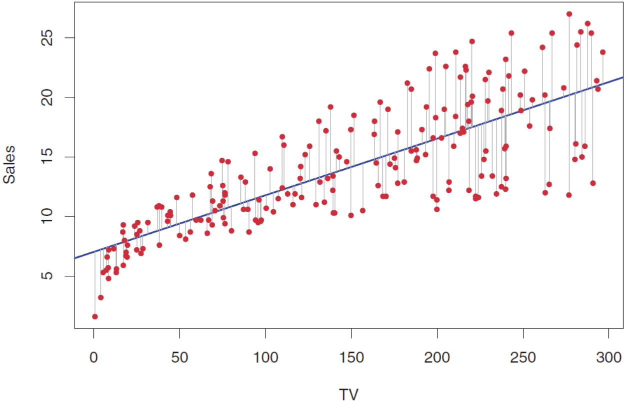
$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)^2 = (\boldsymbol{y} - \beta_0 - \beta_1 \boldsymbol{x})^T (\boldsymbol{y} - \beta_0 - \beta_1 \boldsymbol{x})$$

is given as

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{\mathrm{Cov}(X, Y)}{\mathrm{Var}(X)},$$
$$\hat{\beta}_0 = \bar{y} - \beta_1 \bar{x}$$

for $\bar{x} = \sum x_i / N$ and $\bar{y} = \sum y_i / N$.

Figure 3.1 (p62) from **ISLR**



Multi-dimensional Linear Regression

For $(p+1)$-vector predictor (X) and scalar response (Y),

$$Y \approx X\beta \quad \longrightarrow \quad \hat{\boldsymbol{y}} = \boldsymbol{X}\beta,$$

where $X_0 = 1$ ($\boldsymbol{X}_{*0} = \mathbf{1}$) and β is a $(p+1)$-column vector.

$$\mathrm{RSS}(\beta) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta)$$

$$\frac{\partial}{\partial\beta}\mathrm{RSS}(\beta) = -\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta) \quad \Rightarrow \quad \hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

For $(p+1)$-vector predictor (X) and K-vector response (Y), the result is similarly given as

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}\boldsymbol{B} \quad \text{where} \quad \hat{\boldsymbol{B}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y},$$

which is the independent regressions on Y_j ($\boldsymbol{Y}_{*j}$) combined together,

$$\hat{\boldsymbol{B}}_{*j} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}_{*j}$$

Newton's method

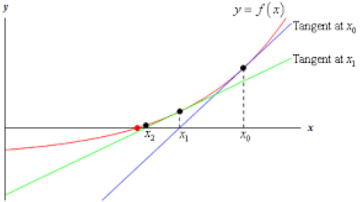
The root x satisfying $f(x) = 0$ can be found by the following iteration:

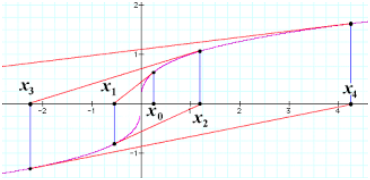
$$x_{n+1} = x_n - \eta \frac{f(x_n)}{f'(x_n)}$$

In multi-dimensional problems, the gradient is used instead of the differentiation:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \eta \frac{f(\boldsymbol{x}_n)\boldsymbol{\nabla} f(\boldsymbol{x}_n)}{|\boldsymbol{\nabla} f(\boldsymbol{x}_n)|^2}$$

Typically we use $0 < \eta < 1$ to avoid *overshooting*.





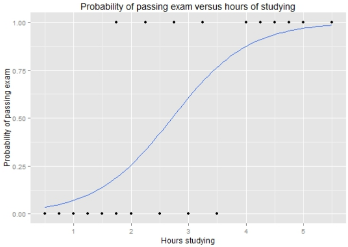
Logistic Regression (Classification)

- Qualitative (categorical) response (binary dependent variable, $Y \in \{0, 1\}$)
- Multiple categories: how to give order?
- Linear regression (quantitative) is not proper
- Logistic (sigmoid) function: $\sigma(\text{logit}) = \text{quantile}$

$$p = \phi(t) = \frac{e^t}{1+e^t} = \frac{1}{1+e^{-t}} \quad \text{for} \quad t = X\beta \; (X_0 = 1)$$

- Logit function (the inverse): log odds

$$\phi^{-1}(p) = \log\left(\frac{p}{1-p}\right) = \log(p) - \log(1-p)$$



Fitting of logistic regression

Likelihood function

- For a given the prediction model, measures the likelihood of a data set.
- The best prediction model/weight is the one that maximizes the likelihood of the dataset.

For a data set $(\mathbf{X}, \mathbf{y})$ where $y_i \in \{0, 1\}$),

$$\begin{aligned} L(\beta) &= \prod_i P(y_i = \hat{y}_i) = \prod_{i:y_i=1} \phi(\mathbf{X}_{i*}\beta) \prod_{i:y_i=0} \big(1 - \phi(\mathbf{X}_{i*}\beta)\big) \\ &= \prod_i \phi(\mathbf{X}_{i*}\beta)^{y_i} \big(1 - \phi(\mathbf{X}_{i*}\beta)\big)^{1-y_i} \\ \log L(\beta) &= \sum_i y_i \log \phi(\mathbf{X}_{i*}\beta) + (1 - y_i) \log \big(1 - \phi(\mathbf{X}_{i*}\beta)\big) \end{aligned}$$

The cost function (to minimize) is $J(\beta) = -\log L(\beta)$

Updating weights

After some algebra,

$$\boldsymbol{w} := \boldsymbol{w} + \Delta\boldsymbol{w}, \quad \Delta\boldsymbol{w} = -\eta\boldsymbol{\nabla} J(\boldsymbol{w}) \tag{1}$$

$$\frac{\partial J(\boldsymbol{w})}{\partial w_j} = -\sum_i (y_i - \phi(\boldsymbol{X}_{i*}w))X_{ij} = -\boldsymbol{X}_{*j}^T(\boldsymbol{y} - \phi(\boldsymbol{X}w)) \tag{2}$$

$$\text{or} \quad \boldsymbol{\nabla} J(\boldsymbol{w}) = -\boldsymbol{X}^T(\boldsymbol{y} - \phi(\boldsymbol{X}w)). \tag{3}$$

We get a similar weight updating rule as that of linear regression and Adaline! It gives a basis for Perceptron's updating rule.

Regularization

We avoid $\boldsymbol{w}$ being too big.

$$J(\beta) = -\log L(\boldsymbol{w}) + \frac{\lambda}{2}|\boldsymbol{w}|^2 \tag{4}$$

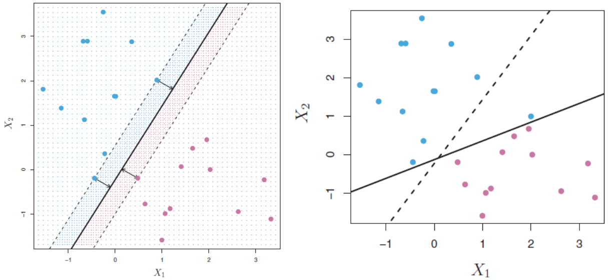
$$J(\beta) = -C\log L(\boldsymbol{w}) + \frac{1}{2}|\boldsymbol{w}|^2 \quad \left(C = \frac{1}{\lambda}, \text{SciKit-Learn}\right) \tag{5}$$

Maximal margin classifier

For $y_i \in \{-1, 1\}$, maximize the margin of the separating hyperplane M,

$$y_i(w_0 + \sum_{j=1}^{p} X_{ij} w_j) = y_i(w_0 + \boldsymbol{X}_{i*}\boldsymbol{w}) \geq M > 0 \text{ for all } i, \text{ with } |\boldsymbol{w}| = 1$$

Maximal margin classifier only works for the separable data set and is sensitive to the change in the *support vectors*.



Support vector classifier

We make maximal margin classifier flexible: maximize the margin of the separating hyperplane M with $|\boldsymbol{w}| = 1$,

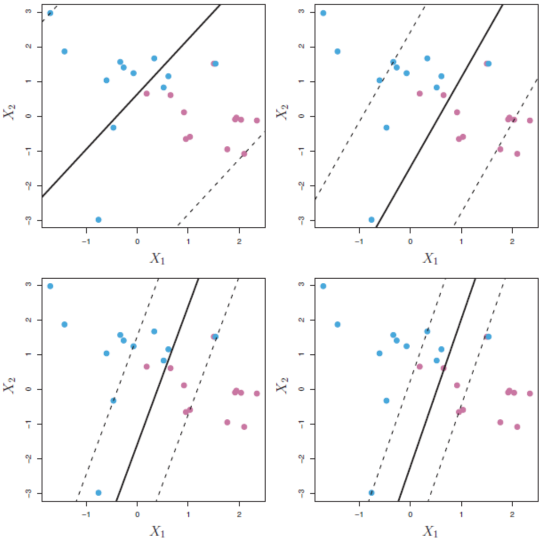
$$y_i(w_0 + \boldsymbol{X}_{i*}\boldsymbol{w}) \geq M(1 - \epsilon_i) \text{ for all } i, \quad \textstyle\sum_{i=1}^n \epsilon_i \leq C,$$

where $\epsilon_i \geq 0$ is *slack variable* indicating the degree of violation ($\varepsilon_i = 0$: no violation, $\varepsilon_i < 1$: margin violation, $\varepsilon_i > 1$: classification violation) and C is a *budget* for the amount of violations by all observations.
Alternatively (in PML), we minimize

$$\frac{1}{2}|\boldsymbol{w}|^2 + C' \sum_{i=1}^{n} \xi_i,$$

where $\{\xi_i \geq 0\}$ satisfies $y_i(w_0 + \boldsymbol{X}_{i*}\boldsymbol{w}) \geq 1 - \xi_i$ for all i. The model converges to maximal margin classifier if C' is very large, so the role of C' is opposite to that of C in the original formulation.

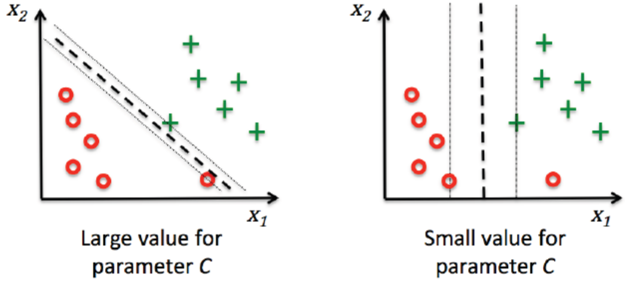
Support vector classifier: role of C



The value of C decreasing from the largest value on top left.

Support vector classifier: role of C'

Large C' (left) vs small C' (right)



Support vector machines (SVM)

How can we deal with non-linear decision boundary?

Enlarging feature space

Including high-order terms, $1, X_j, \cdots, X_j^2, \cdots, X_i X_j, \cdots$, can be helpful, but the computation becomes very heavy.

Kernel

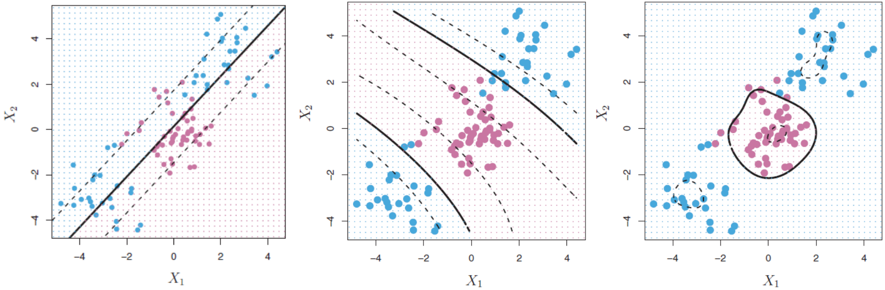
Instead we introduce kernel function, as a generalization of dot product in hyperplane:

- Linear: $K(\boldsymbol{X}_{i*}, \boldsymbol{X}_{i'*}) = \boldsymbol{X}_{i*} \boldsymbol{X}_{i'*}^T$
- Polynomial: $K(\boldsymbol{X}_{i*}, \boldsymbol{X}_{i'*}) = (1 + \boldsymbol{X}_{i*} \boldsymbol{X}_{i'*}^T)^d$
- Radial basis: $K(\boldsymbol{X}_{i*}, \boldsymbol{X}_{i'*}) = \exp(-\gamma |\boldsymbol{X}_{i*} - \boldsymbol{X}_{i'*}|^2)$

Kernel $K(\boldsymbol{X}_{i*}, \boldsymbol{X}_{i'*})$ can be understood as a *distance* between two observations: $\boldsymbol{X}_{i*}$ and $\boldsymbol{X}_{i'*}$ are similar if the kernel value is high (low) whereas they are different if low (high).

SVM: non-linear decision boundary

SVM classification with linear kernel (left) polynomial kernel of degree 3 (middle) and radial basis kernel (right)



K Nearest Neighbor (KNN)

If $N_K(x)$ is the set of the K nearest neighbors around x,

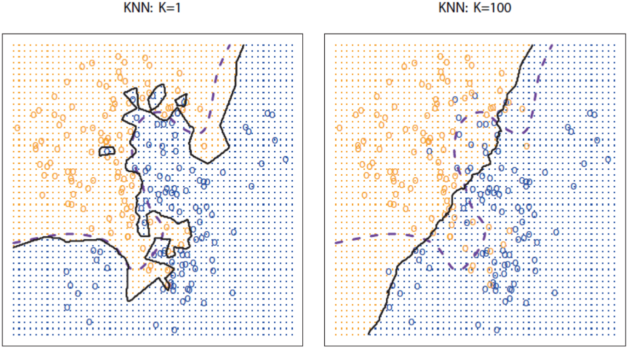
$$\text{Regression:}\hat{y} = f(x) = \frac{1}{K} \sum_{x_i \in N_K(x)} y_i$$

$$\text{Classifier: } \hat{y} = \text{majority of } \{y_i\} \text{ for } x_i \in N_K(x)$$

$$\text{Prob}(y = j|x) = \frac{1}{K} \sum_{x_i \in N_K(x)} I(y_i = j)$$

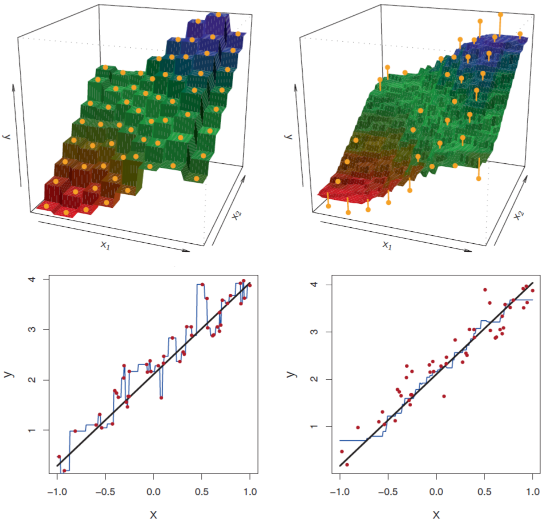
- Parametric vs Non-parametric model
- Learning step is not required, but KNN is intensive in both computation and storage (*memorize* training data set for prediction)

KNN: Classfier example

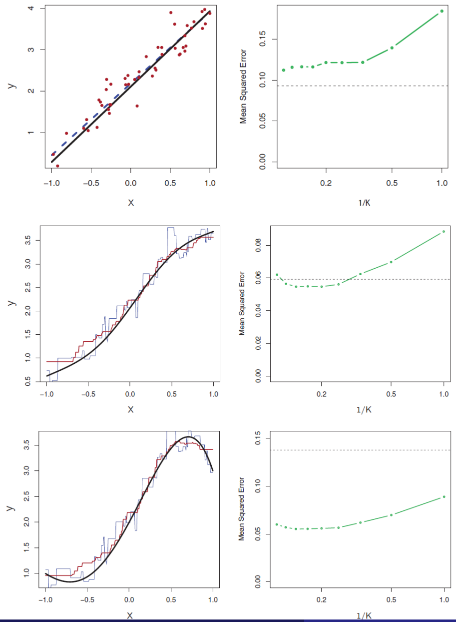


KNN: Regression example

$K = 1$ (left) vs $K = 9$ (right)



KNN: Parametric vs Non-parametric



Non-parametric regression works better as the true function deviates from the basis function (linear function in this example).