Recent Advances of Machine Learning

Jong-June Jeon

University of Seoul

September 7, 2017

- Learning based on data
- Regularization method
- Classification and deep learning

Learning based on data

Regression: Statistical method for learning the relation between two more variables

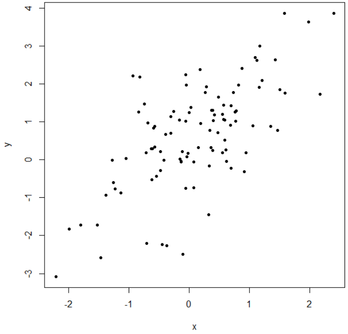


Figure: Scatter plots of paired data

Regression: Statistical method for learning the relation between two more variables

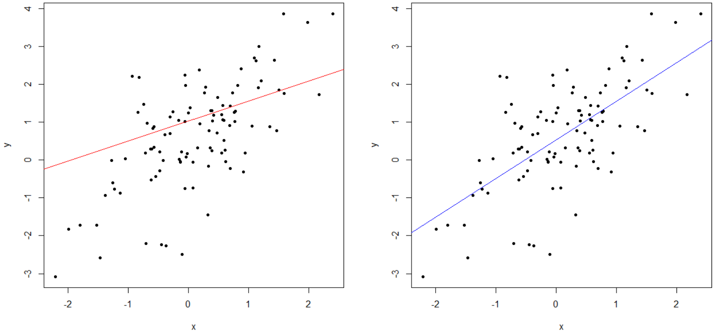


Figure: Which one is better?

Conventional interest in regression models

- Detected signal is statistically significant?
- If the true model is linear, then how can I measure the uncertainty of the estimated model
- What is the lower bound of the asymptotic variance of the estimated models.
- What is the most efficient estimation method?
- How can I select the true model under large samples?

Inference !

Prediction

- If we are only interested in prediction, there is another story.
    - Inference is out of interest (eg p-value, R square, selection consistency)
    - Under the circumstances of accumulating data, the learned (estimated) model depends on the observed data.
    - Predictive performances should be assessed by future data.

Goal is to improve prediction accuracy !

Change of paradigm

- Past: data is scare resource. If data is observed once, the model is estimated based on the observed data.
- Now: data is not scare resource any more. Whenever data is observed, the model is learned(estimated) based on the data.

Linear model

- Explanatory variable : $\mathbf{x} \in \mathbb{R}^p$
- Response variable: $y \in \mathbb{R}$
- Linear predictor: $\hat{y} = \mathbf{x}^T \hat{\boldsymbol{\beta}}$

$\hat{\boldsymbol{\beta}} = (\hat{\beta}_1, \cdots, \hat{\beta}_p)^T \in \mathbb{R}^p$ is called of regression coefficient.

Linear model: estimation

- Let training set be $\mathcal{T}_r = \{(y_i, \mathbf{x}_i) : 1 \le i \le n\}$
- Least square method is given by the minimizer of $RSS(\boldsymbol{\beta})$ where

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2$$

- This estimator is called of least square estimator(LSE)

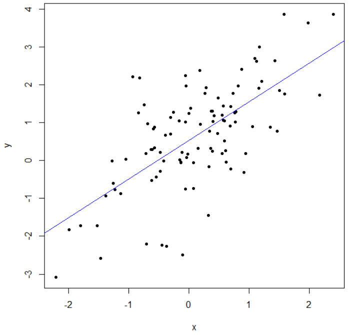


Figure: Estimated predictor: $\hat{Y} = 0.5274 + 1.0212X$

Linear model: estimation

Generative model assumption

- $y_i = \mathbf{x}_i^T \boldsymbol{\beta}^* + \epsilon_i$
- $\epsilon_i \sim_{iid} (0, \sigma^2)$

- If the true model is linear, it is known that LSE, $\hat{\boldsymbol{\beta}}$, converges to the true parameter in probability.
- $\sqrt{n}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*)$ weekly converges Gaussian distribution.

Linear model: estimation

- Note that the uncertainty of $\hat{\boldsymbol{\beta}}$ depends on the training set $\mathcal{T}_r$
- Let $\mathbf{x}_0$ be evaluation points then linear predictor is $\mathbf{x}_0^T\hat{\boldsymbol{\beta}}$ and the risk defined though l_2 loss function is given by $\mathrm{E}_F(Y - \mathbf{x}_0\hat{\boldsymbol{\beta}})^2$ where $Y \sim F$ (conditional distribution given $X = \mathbf{x}_0$.
- Generally risk of the predicted model is given by

$$R(\hat{\boldsymbol{\beta}}) = \mathrm{E}_F(Y - X\hat{\boldsymbol{\beta}})^2$$

where $(Y, X) \sim F$

Linear model:Estimation

- model bias: discrepancy between true model and expectation of estimated model.
- Suppose that $\mathrm{E}(Y|\mathbf{X} = \mathbf{x}) \neq \mathbf{x}_i^T \boldsymbol{\beta}_j^*$
- For example,
  - $Y_i = 3 + X_i^2 + \epsilon_i$ where $\epsilon_i \sim N(0, 1)$
  - Note that $E(Y|\mathbf{X}_i = x) = 3 + x^2$
  - Try to fit a model in linear model space.

Linear model:Estimation

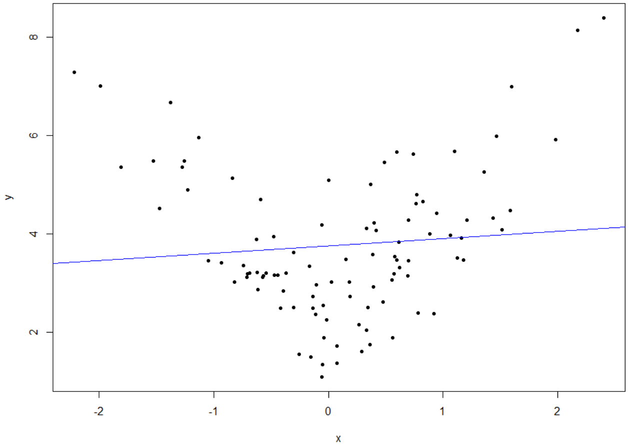


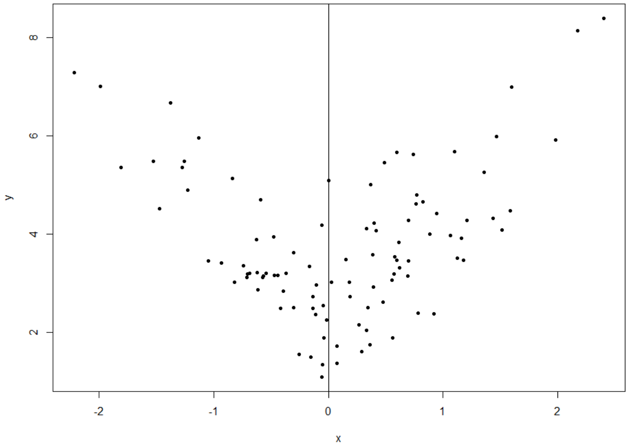
Figure: Blue line denotes the estimate model in the linear model space

Learning based on algorithm

- K-nearest neighbourhood algorithm
- Let (y_i, x_i) for $i = 1, \cdots, n$ be training samples.
- $N_k(x)$: the index of k samples close to x
- Predictor is given by

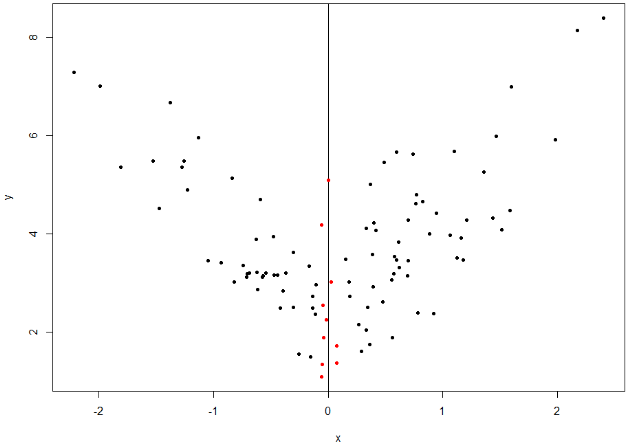
$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

10-**NN algorithm**



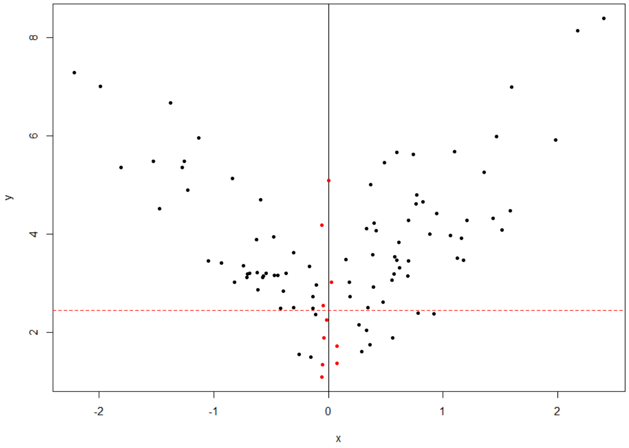
$\hat{Y}(0) =$

10-**NN algorithm**



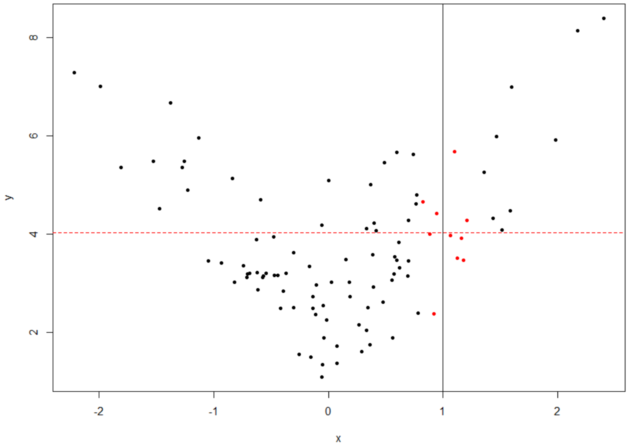
$\hat{Y}(0) =$

10-**NN algorithm**



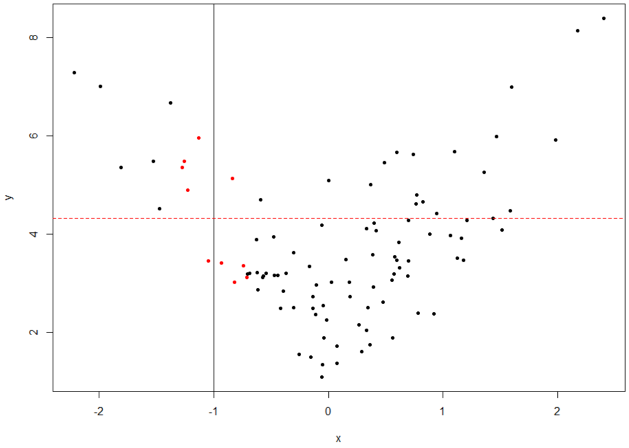
$$\hat{Y}(0) = 2.446989$$

10-**NN algorithm**



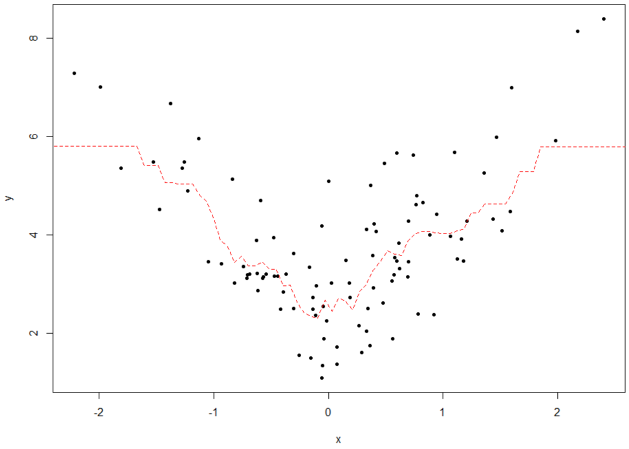
$$\hat{Y}(1) = 4.024909$$

10-NN algorithm



$$\hat{Y}(-1) = 4.314436$$

10-**NN algorithm**



Linear model in classification problem

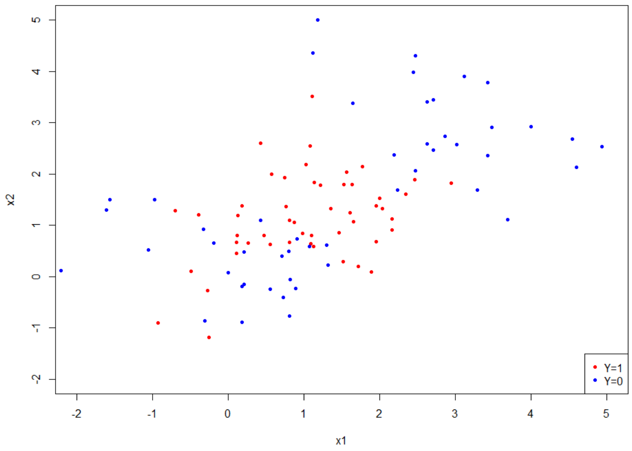
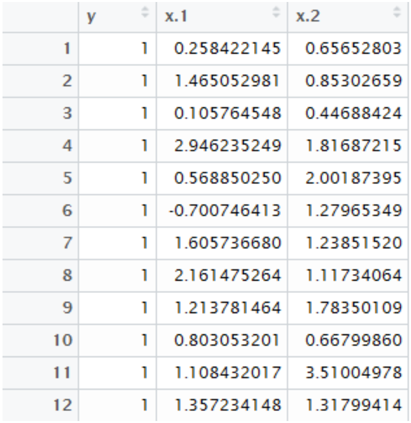


Figure: $\mathbf{x} = (x_1, x_2)' \in \mathbb{R}^2$

Linear model in classification problem



	y	x.1	x.2
1	1	0.258422145	0.65652803
2	1	1.465052981	0.85302659
3	1	0.105764548	0.44688424
4	1	2.946235249	1.81687215
5	1	0.568850250	2.00187395
6	1	-0.700746413	1.27965349
7	1	1.605736680	1.23851520
8	1	2.161475264	1.11734064
9	1	1.213781464	1.78350109
10	1	0.803053201	0.66799860
11	1	1.108432017	3.51004978
12	1	1.357234148	1.31799414

Linear model in classification problem

- linear predictor: $\hat{y} = 0.62223 - 0.03565x_1 - 0.05486x_2$
- decision boundary:
  $\{(x_1, x_2) \in \mathbb{R}^2 : 0.5 = 0.62223 - 0.03565x_1 - 0.05486x_2\}$

Linear model in classification problem

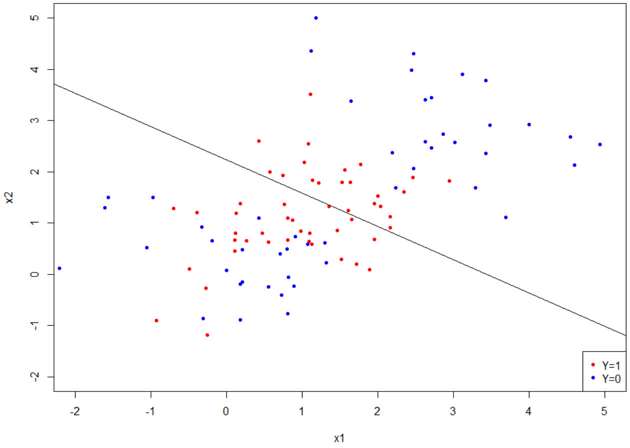
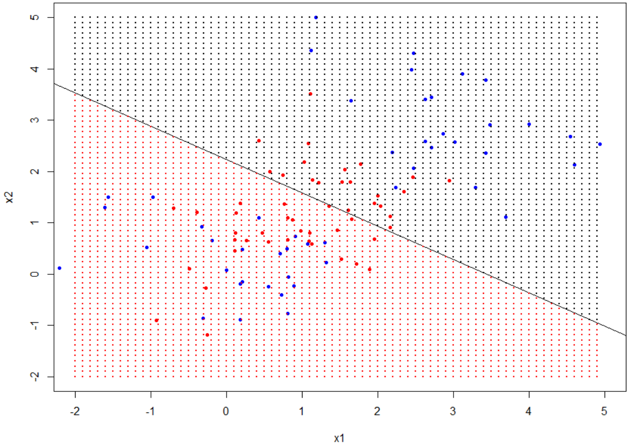


Figure: decision boundary is given by hyperplane

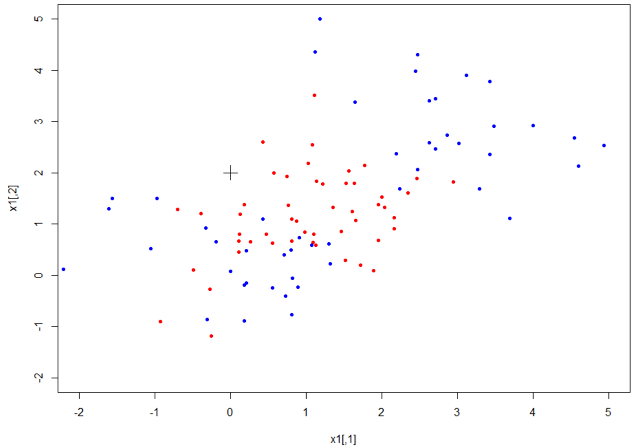
Linear model in classification problem



K-NN in in classification problem

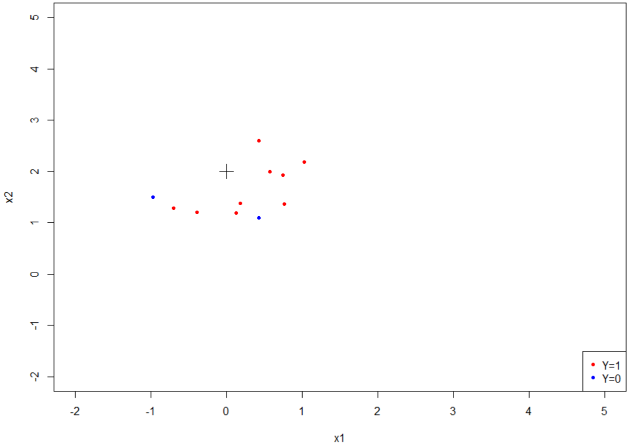
$$\hat{Y}(x) = \begin{cases} 1 & \text{if } \frac{1}{k}\sum_{x_i \in N_k(x)} y_i \geq 1/2 \\ 0 & o.w \end{cases}$$

K-NN in in classification problem



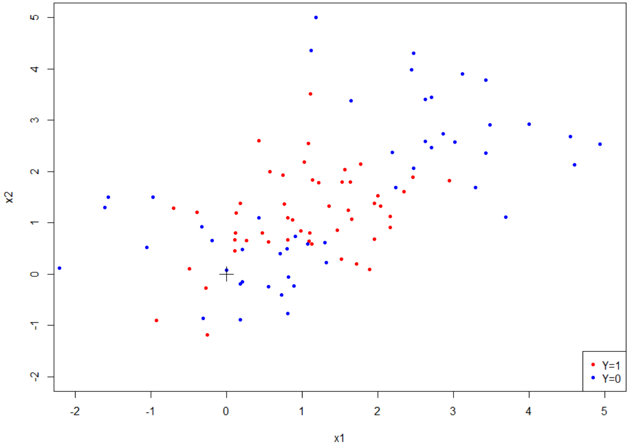
$\frac{1}{k}\sum_{x_i \in N_k(x)} y_i =$ $\hat{Y}((0,2)) =$

K-NN in in classification problem



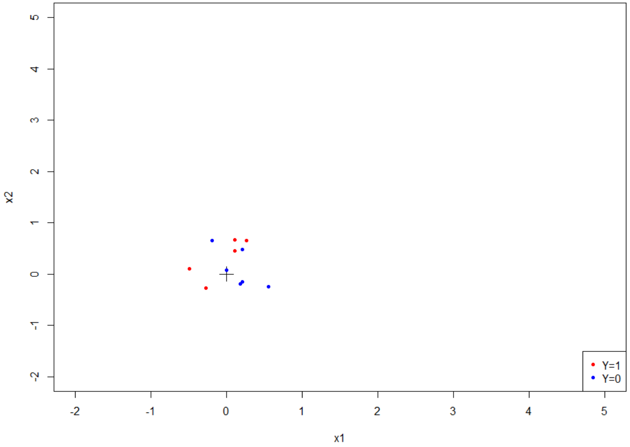
$$\frac{1}{k}\sum_{x_i \in N_k(x)} y_i = 9/11 \geq 1/2 \Rightarrow \hat{Y}((0,2)) = 1$$

K-NN in in classification problem



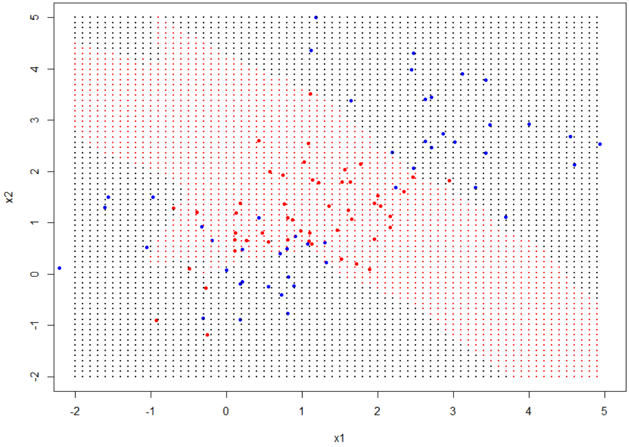
$\frac{1}{k}\sum_{x_i \in N_k(x)} y_i =$ $\hat{Y}((0,0)) =$

K-NN in in classification problem



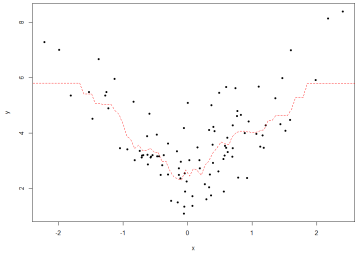
$$\frac{1}{k}\sum_{x_i \in N_k(x)} y_i = 5/11 \le 1/2 \Rightarrow \hat{Y}((0,0)) = 0$$

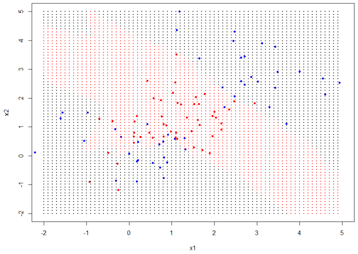
K-NN in in classification problem



Non-linear decision boundary is obtained by K-NN algorithm.

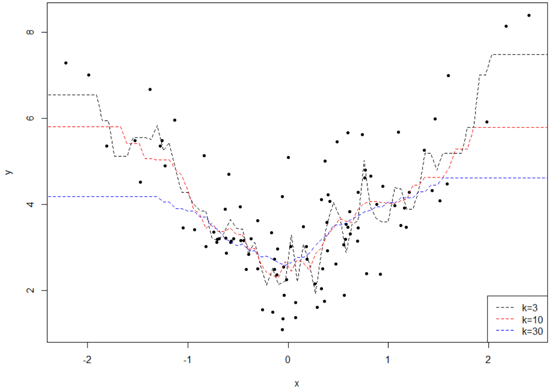
K-NN in in classification problem



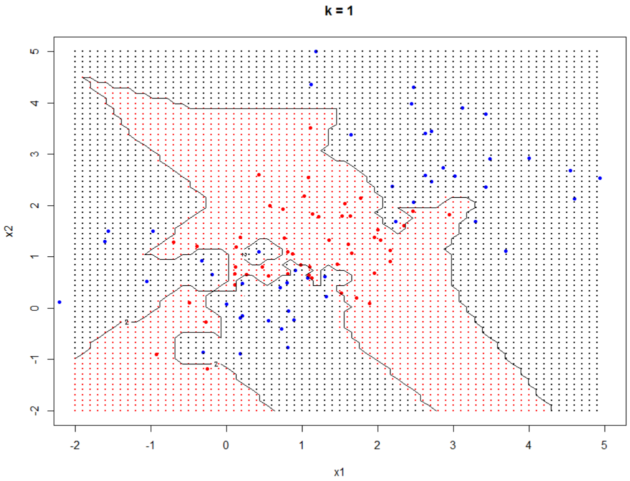


Model complexity

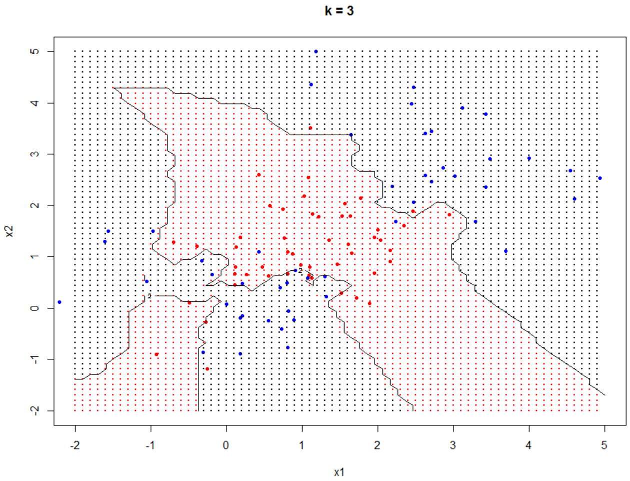
- The selection of K is crucial to obtain the best predictive performance.



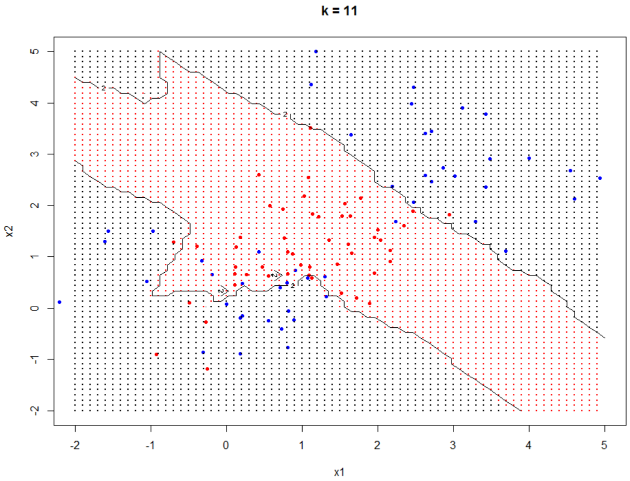
Model complexity



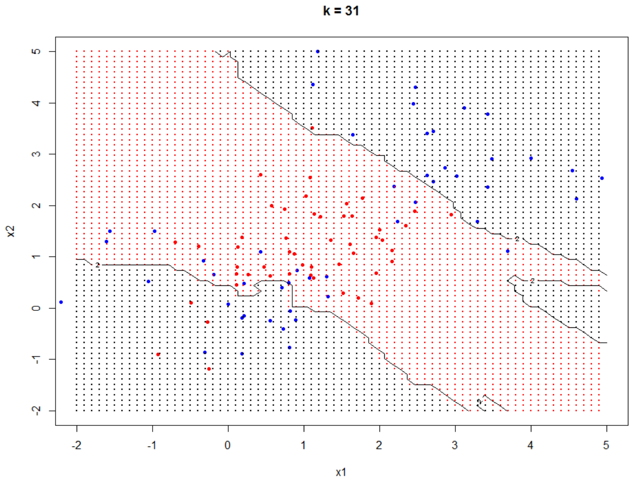
Model complexity



Model complexity



Model complexity



Idea of K-NN algorithm

- K-NN algorithm can produce nonlinear model easily.
- The idea of K-NN algorithm is local approximation.

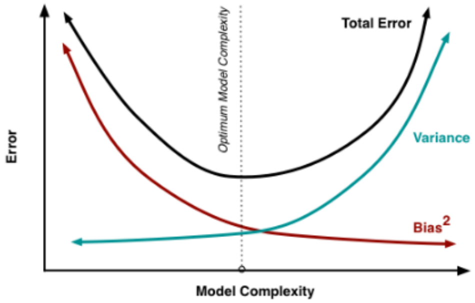
$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

- For small K less data are used for estimating local mean such that model become complex and variance of the local mean increases.
- For large K vice versa.

Bias-Variance Trade off

$$\text{Prediction error} = (\text{Model bias})^2 + \text{Model Variance}$$

- Complex model : Large variance and small bias
- Simple model: small variance and large bias



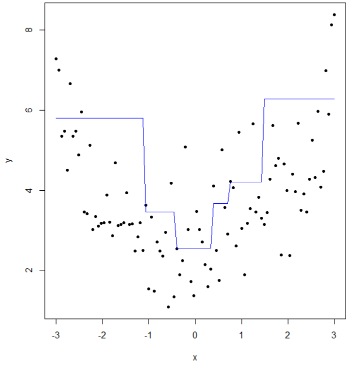
Selection of model with moderate complexity is required to achieve the best predictive model under restricted training sample.

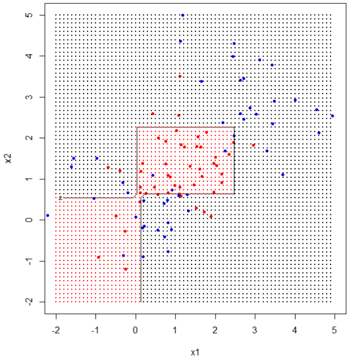
Ensemble method: variance reducing method by combining predicted models(weak learners)

We will see examples of ensemble method applied to tree model.

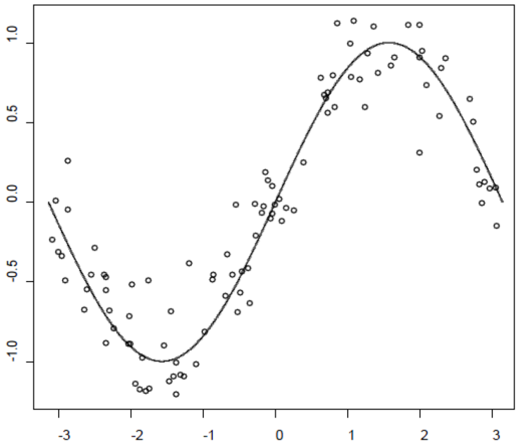
Tree model

- Learning algorithm to split regions

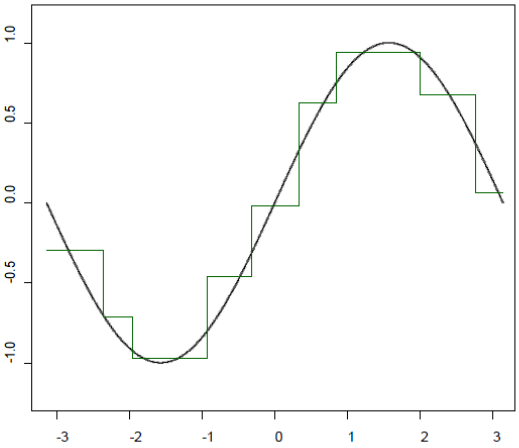




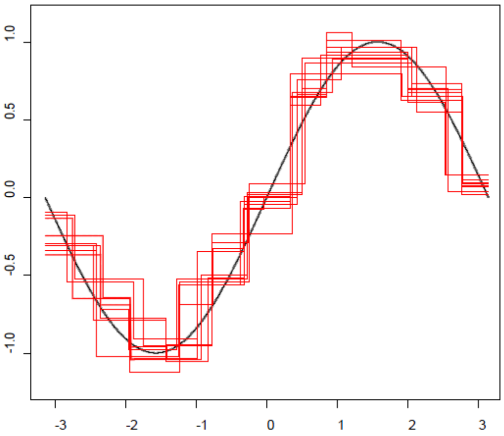
data underlying function



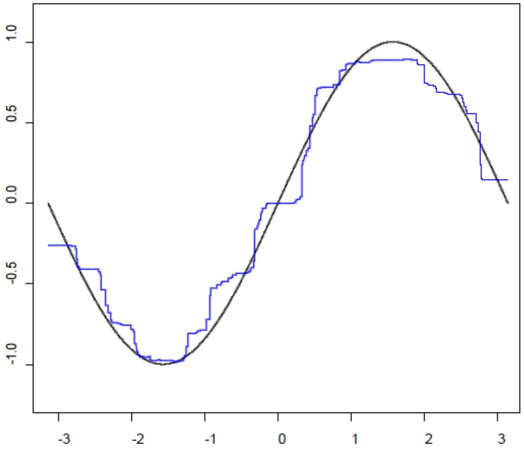
Single regression tree



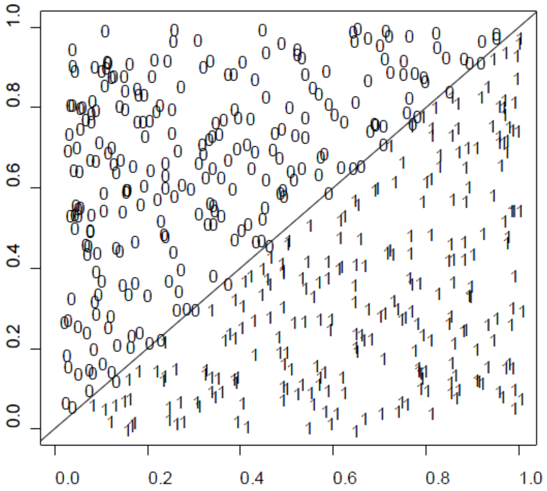
10 regression trees using randomly sampled data



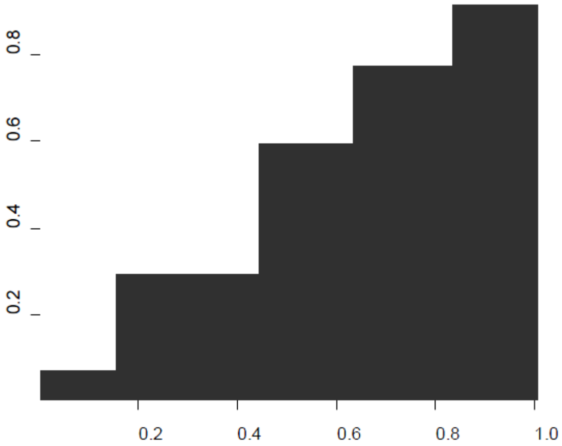
averaging tree



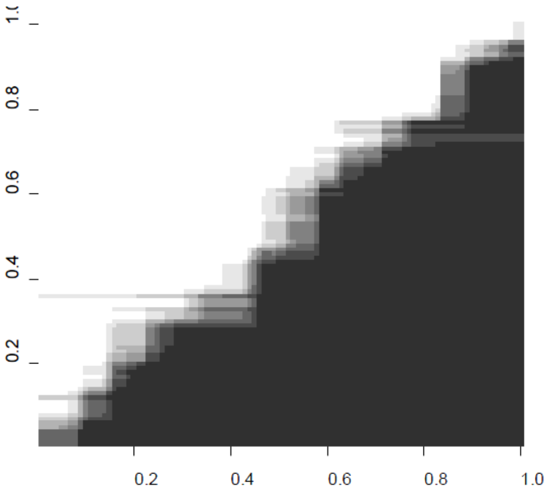
Hard problem for classification tree



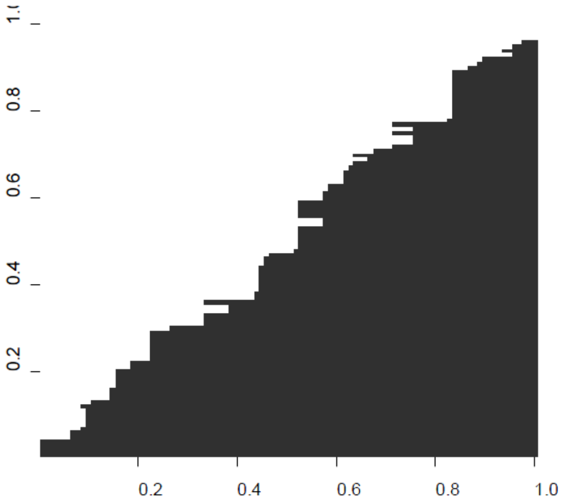
single tree



25 averaged tree



25 voted tree



bagging: Bootstrap Aggregating

- Bootstrapping: re-sample data B times $\mathcal{T}_r^{(b)}$
- Aggregating
  - Learn model $\hat{f}^b$ using $\mathcal{T}_r^{(b)}$ for each $b = 1, \cdots, B$
  - Aggregating
    - Regression

$$\hat{f}_{bag}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(\mathbf{x})$$

    - Classification

$$\hat{f}_{bag}(\mathbf{x}) = \text{Voting}\{\hat{f}^b(\mathbf{x}) : 1 \leq b \leq B\}$$

Thererical background of bagging

- Variance reduction
- Bias does not change

Note that "Prediction error $=$Bias^2 $+$ Variance"

Linear model

Why did so many statistician and mathematician study linear model?

High dimensional linear model

- Suppose that $Y_i = f(Z_i) + \epsilon_i$ where f is smooth function.
- For sufficiently large p $f(z) \simeq \beta_0 + \sum_{j=1}^{p} \beta_j z^j$ and

$$\mathrm{E}(Y_i|Z_i) \simeq \beta_0 + \sum_{j=1}^{p} \beta_j X_{ij}$$

  where $X_{ij} = Z_i^j$.
- Let $\mathbf{X}_i = (X_{i1}, \cdots, X_{ip})' \in \mathbb{R}^p$ and $Y_i \in \mathbb{R}$
- Linear model

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} + \epsilon_i$$

High dimensional linear model linear model
Even for non-smooth function f,

$$\mathrm{E}(Y_i|\mathbf{X}_i) \simeq \beta_0 + \sum_{j=1}^{p} \boldsymbol{\beta}_j' B_j(\mathbf{X}_i)$$

by appropriate selection of basis functions B_js.

How to control model complexity in the high dimensional linear model?

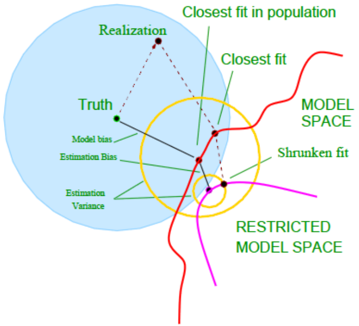
Regularization method

Statistical learning by empirical risk minimization

- $\mathbf{z}_i = (y_i, \mathbf{x}_i')' \sim_{iid} \mathcal{P}$
- $y_i = g(\mathbf{x}_i) + \epsilon_i$ for $i = 1 \cdots, n$ for $g \in \mathcal{G}$
- loss function : $l : \mathcal{Z} \times \mathcal{G} \mapsto \mathbb{R}^+$
  - l_2 loss function: $l(\mathbf{z}_i, g) = (y_i - g(\mathbf{x}_i))^2$
- risk function:
  - Risk function : $R(g) = \mathrm{E}_{\mathcal{P}} l(\mathbf{z}, g)$
  - Empirical risk function: $R_n(g) = \sum_{i=1}^n l(\mathbf{z}_i, g)/n$
- Statistical learning

$$\hat{g} = \mathrm{argmin}_{g \in \mathcal{G}} R_n(g)$$

Visualization of bias-variance trade-off



Regularization method utilizes the bias-variance trade-off by restricting model space

Regularization

$$\begin{aligned}\hat{g} &= \text{argmin}_{g\in\mathcal{G}} R_n(g) \\ &\quad \text{subject to } J(g) \le C,\end{aligned}$$

where $J : \mathcal{G} \mapsto \mathbb{R}^+$ is a penalty (regularization) function

We can write the above optimization problem as follows:

$$\hat{g} = \text{argmin}_g R_n(g) + \lambda J(g)$$

for some $\lambda \ge 0$

LSE

- $(y_i, \mathbf{x}_i)$ for $i = 1, \cdots, n$: pairs of response and explanatory variables ($y_i \in \mathbb{R}$ and $\mathbf{x}_i \in \mathbb{R}^p$)
- $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_p)'$
- $y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$ ($\epsilon_i \sim_{iid} (0, \sigma^2)$)
- LSE:

$$\hat{\beta} = \underset{\boldsymbol{\beta}}{\text{argmin}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i'\boldsymbol{\beta})^2$$

LSE

- LSE is the Best linear Unbiased Estimator (BLUE).
- When $n > p$ LSE is not unique.
- When $n \simeq p$, the variance of LSE is large.

Ridge estimator

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}}{\text{argmin}} \sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{\text{penalty function}}$$

for $\lambda > 0$.

- Ridge estimator always exists.
- There exist a $\lambda > 0$ such that the ridge estimator corresponding to the λ has better predictive performance than LSE.
- When $p > n$ the consistency of $\hat{\boldsymbol{\beta}}_\lambda$ (convergence to the true parameter) is not guaranteed.

LASSO estimator

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}}{\text{argmin}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i' \boldsymbol{\beta})^2 + \underbrace{\lambda \sum_{j=1}^{p} |\beta_j|}_{\text{penalty function}}$$

for $\lambda > 0$.

- In the high dimensional problem ($p \simeq \exp(n)$) LASSO works well.
- Under regularity condition the lasso estimator achieves minimax optimal error bound.
- But strong model conditions are required for selection consistency.

Regularized estimator with nonconvex penalty

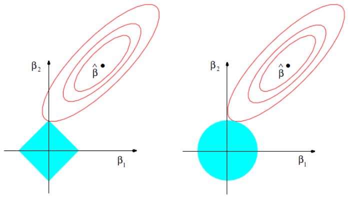
$$\hat{\boldsymbol{\beta}}_\lambda = \operatorname*{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i' \boldsymbol{\beta})^2 + \underbrace{\sum_{j=1}^{p} G_\lambda(|\beta_j|)}_{\text{penalty function}}$$

for $\lambda > 0$.

- When signals is large ($> O(1/\sqrt{n})$), the method can choose the signal variable well (oracle property).
- But strong model conditions are required for minimax optimality.

Regularization method provides a useful view of machine learning application

Sparsity



- Form non-differential points in model constraints we know that there are positive probability that estimated model has exactly zero coefficients.
- By introducing various type of constraints having non-differential points, structural learning is possible.

Structural sparse modeling 1

- $y_i = \mu_i + \epsilon_i$ for $i = 1, \cdots, n$.
- $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_n)$:

$$(\hat{\mu}_1, \cdots, \hat{\mu}_n) = \underset{\mu}{\text{argmin}} \sum_{i=1}^{n} (y_i - \mu_i)^2 + \underbrace{\lambda \sum_{j=1}^{n-1} |\mu_{j+1} - \mu_j|}_{\text{penalty function}}$$

Note that

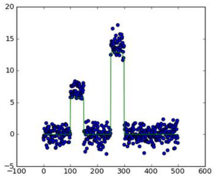
- $\lambda = \infty$: $\hat{\mu}_j = \bar{y}$ for all j.
- $\lambda = 0$: $\hat{\mu}_j = y_i$ for all j.
- λ controls the number of change points.

Signal approximator

Model:

$$y_i = \beta_i + \epsilon_i$$

where $\epsilon_i \sim_{iid} N(0, \sigma^2)$ for $i = 1, \cdots, n$.



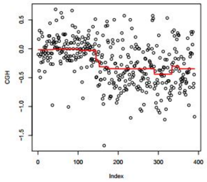


Figure: Sigmal approximator

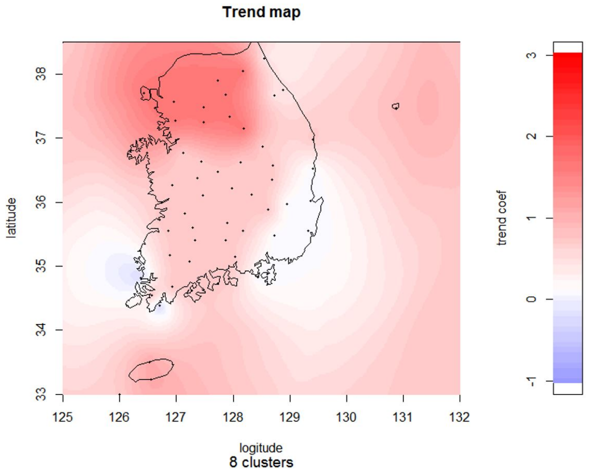
Structural sparse modeling 2

- $y_i = \mathbf{x}_i'\boldsymbol{\beta} + \epsilon_i$ for $i = 1, \cdots, n$.
- We obtain the following estimator of

$$\hat{\boldsymbol{\beta}} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \lambda \sum_{j \neq k} |\beta_j - \beta_k|$$

- clustering of estimate coefficients.

Map of estimated trends in extreme precipitation



Classification and deep learning

Binary Classification

- Reponse varible: $y \in \mathcal{Y} = \{-1, 1\}$
- Explanatory variable: $\mathbf{x} \in \mathcal{X}$
- Classification function: $C : \mathcal{X} \mapsto \mathcal{Y}$

ex) Let y be a variable denotes disease or normal, and let $\mathbf{x}$ be a vector of result of diagnosys. A doctor is a classification function to map $\mathbf{x}$ to $\mathcal{Y}$.

Let P be a distribution of $(y, \mathbf{x})$.

- missclassification error of C on population:

$$P(C(\mathbf{x}) \neq y) = P(C(\mathbf{x}) = -1, y = 1) + P(C(\mathbf{x}) = 1, y = -1)$$

- Bayes error:

$$\min_{C} P(C(\mathbf{x}) \neq y)$$

- Bayes classifier:

$$C^* = argmin_C P(C(\mathbf{x}) \neq y)$$

Note that the bayes classifier is given by

$$C^*(\mathbf{x}) = \begin{cases} 1 & \text{if } P(y = 1|\mathbf{x}) \geq 0.5 \\ -1 & o.w. \end{cases}$$

We assume that $(y_i, \mathbf{x}_i)$ for $i = 1, \cdots, n$ are iid random samples.

$$\min_C \sum_{i=1}^{n} \#(C(\mathbf{x}_i) \neq y_i)/n$$

- The final goal is to estimate a classifier which minimizes the classification error.
- The function C is too complex such that we adopt an alternative method to construct classifier.

Scoring function

- $f : \mathcal{X} \to \mathbb{R}$: a function assigns a score to the observation with covariate x. Using this score, we construct a classifier as following;

$$C(\mathbf{x}; f) = \begin{cases} 1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & o.w. \end{cases}$$

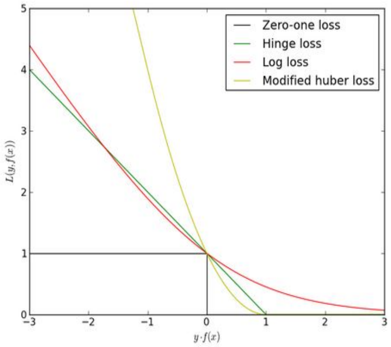
Estimation and surrogate loss function It is reasonable to find f minimizing

$$\sum_{i=1}^{n} I(C(\mathbf{x}_i; f) \neq y_i)/n,$$

which is equivalently written by $\sum_{i=1}^{n} I(y_i f(\mathbf{x}_i) < 0)$. Here $y_i f(\mathbf{x}_i)$ is called margin.
cf) $I(x > 0) = 1$,if $x > 0$, $I(x > 0) = 0$, otherwise.

Convex surrogate loss function
However, this task requires to heavy computation so that we cannot use this method. The complexity comes frome 0-1 loss function. We replace the 0-1 loss function with other convex loss function called of the surrogate loss function.

Surrogate loss function



Surrogate loss function

Finally, we minimize the surrogate risk function

$$L(f;\phi) = \sum_{i=1}^{n} \phi(y_i f(\mathbf{x}_i))/n$$

The estimator in the classfication problem is given by

$$\hat{f} = argmin_f L(f;\phi)$$

If the region of $\hat{f}(\mathbf{x}) = 0$ is similar to that of $P(Y = 1|\mathbf{x}) = 0.5$, then $\hat{f}(\mathbf{x})$ gives an approximated bayes classifier.

$$\hat{C}(\mathbf{x}) = \begin{cases} 1 & \text{if } \hat{f}(\mathbf{x}) \geq 0.5 \\ -1 & o.w \end{cases}$$

If $\hat{f}(\mathbf{x})$ is linear, the region $\hat{f}(\mathbf{x}) = 0$ is linear space. If the bayes classifier is not linear, the model bias in classification always exists.

Example: logistic regression

- linear model: $f(\mathbf{x}) = \mathbf{x}'\beta$
- logistic loss: $\phi(yf(\mathbf{x})) = -yf(\mathbf{x}) + \log(1 + \exp(-yf(\mathbf{x})))$
- estimator: $argmin_\beta \sum_{i=1}^n \phi(y_i \mathbf{x}_i' \beta)$

The estimator is equal to Maximum likelihood estimator in the logistic regression model.

How can we produce a complex model f?

Estimation of non-linear classifier

- Additive models: assume that f is a additive model of trees (stumps) or simple classifier. When $\phi(z) = \exp(-z)$ and ridge penalty (see regularization) is applied, then the classification problem is called adaBoost.
- Feature mapping: assume that the input space are project on a feature space. When $\phi(z) = (1 - z)_+$ and ridge penalty (see regularization) is applied, then the classification problem is called support vector machine.

Neural Network for classification

- Composition
  - Input data is mapped onto linear space
  - Its image is transformed by non-linear activation function (sigmoid function, tanh, RELU...)
  - The transformed data is mapped onto linear space again.
  - $\cdots$
  - A feature of input data is obtained by above compositions.
  - Apply the conventional classification method.

Neural Network for classification

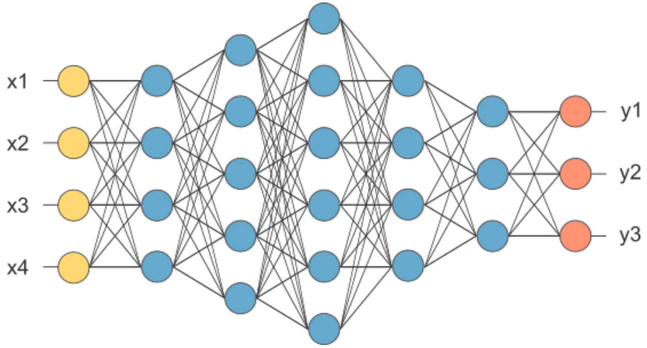


Figure: Visualization of neural network

Neural Network with l_2 loss for classification

Here we consider a single layer neural network without a bias term for notational simplicity. The objective function is given by

$$L(\boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} (1 - y_i (\sum_{j=1}^{k} \beta_j \sigma(\mathbf{x}' \boldsymbol{\alpha}_j))^2 / n,$$

where σ is sigmoid function, $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_k)$ and $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_k)$.

Neural Network for classification

$$argmin_{\beta,\alpha} L(\boldsymbol{\beta}) = argmin_{\beta,\alpha} \sum_{i=1}^{n} (1 - y_i \sum_{j=1}^{k} \beta_j \sigma(\mathbf{x}_i' \boldsymbol{\alpha}_j))^2$$

Note that the score functions is given by $f(x) = \sum_{j=1}^{k} \beta_j \sigma(\mathbf{x}' \boldsymbol{\alpha}_j)$ and the surrogate loss fucntion $\phi(x) = (1-x)^2$.

- It is known that all methods (additive models, feature mapping, composition) can achieves ideal decision boundary as # of data goes to infinity.
- Then the natural question is that "what is more efficient way to construct a complex model": which methods requires less parameters to construct a nearly optimal decision functions.
- Answer is simple. It depends on the true model. But many difficult problems are solved by deep neural network (image, video data analysis). $\Rightarrow$ High order composition works ! (deep learning)

Deep neural network

- Construct a nonlinear model by compositions

$$f(\mathbf{x}) = h_k \circ h_{k-1} \cdots \circ h_1(\mathbf{x})$$

- Regularization (tune the estimated model)
  - Scheduling learning rate
  - Selection of moment parameter
  - Drop-out rate
  - # of layer (depth)
  - $\cdots$
- Computational issue
  - # of parameter is very large $(10^7 \sim)$
  - Develop parallel optimization algorithm.
  - GPGPU computing

- Recent advances of machine learning answers the question, “how to efficiently construct a model space that can fitted well for considered data.”
- Three methods, additive models, high order feature mapping, compositions, are competing.
- Successes in engineering fields tells that complex model space induced by compositions is useful for many areas.
- But regularizing the model is still crucial task to select the best model.

Thank you