

Imbalanced Classification. Multiclass Classification. Nonparametric Estimation

Evgeny Burnaev

Skoltech, Moscow, Russia

Skoltech

Skolkovo Institute of Science and Technology

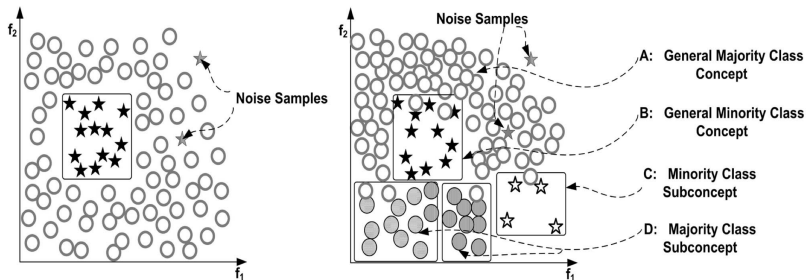
- 1 Imbalanced Classification
- 2 Multi-Class Classification
- 3 Nonparametric Estimation

1 Imbalanced Classification

2 Multi-Class Classification

3 Nonparametric Estimation

- Between-class imbalance (relative imbalance)
- Relative imbalance vs. imbalance due to rare instances or “absolute rarity”
- Data complexity vs. imbalanced data vs. small sample size



- Binary classification: often dataset has “natural” imbalance
- Minor class (of **prime** interest) vs. major class: e.g. classification of “cancerous” vs. “healthy” mammography image
- Standard classifiers (SVM, kNN, log. reg., etc.): classes are equally important \Rightarrow results are biased towards the major class
- Poor prediction of minor class while the average quality can be good:
 - target events occurs in 1% of all cases
 - classifier always gives a ‘no-event’ answer
 - it is wrong just 1% of all cases

- Binary classification: often dataset has “natural” imbalance
- Minor class (of **prime** interest) vs. major class: e.g. classification of “cancerous” vs. “healthy” mammography image
- Standard classifiers (SVM, kNN, log. reg., etc.): classes are equally important \Rightarrow results are biased towards the major class
- Poor prediction of minor class while the average quality can be good:
 - target events occurs in 1% of all cases
 - classifier always gives a ‘no-event’ answer
 - it is wrong just 1% of all cases

- Binary classification: often dataset has “natural” imbalance
- Minor class (of **prime** interest) vs. major class: e.g. classification of “cancerous” vs. “healthy” mammography image
- Standard classifiers (SVM, kNN, log. reg., etc.): classes are equally important \Rightarrow results are biased towards the major class
- Poor prediction of minor class while the average quality can be good:
 - target events occurs in 1% of all cases
 - classifier always gives a ‘no-event’ answer
 - it is wrong just 1% of all cases

- Binary classification: often dataset has “natural” imbalance
- Minor class (of **prime** interest) vs. major class: e.g. classification of “cancerous” vs. “healthy” mammography image
- Standard classifiers (SVM, kNN, log. reg., etc.): classes are equally important \Rightarrow results are biased towards the major class
- Poor prediction of minor class while the average quality can be good:
 - target events occurs in 1% of all cases
 - classifier always gives a ‘no-event’ answer
 - it is wrong just 1% of all cases

- Approaches to increase importance of the minor class:
 - Adapt a probability threshold for classifiers,
 - Modify a loss function, e.g., by assigning more weight to the minor class error,
 - Resample a dataset in order to soften or remove class imbalance
- We focus on resampling: convenient, allows to use standard classifiers
- The main aim:
 - review and compare main resampling methods,
 - compare strategies of resampling amount (i.e., how many observations to add or drop) selection,
 - explore their influence on quality of classification

- Approaches to increase importance of the minor class:
 - Adapt a probability threshold for classifiers,
 - Modify a loss function, e.g., by assigning more weight to the minor class error,
 - Resample a dataset in order to soften or remove class imbalance
- We focus on resampling: convenient, allows to use standard classifiers
- The main aim:
 - review and compare main resampling methods,
 - compare strategies of resampling amount (i.e., how many observations to add or drop) selection,
 - explore their influence on quality of classification

- Approaches to increase importance of the minor class:
 - Adapt a probability threshold for classifiers,
 - Modify a loss function, e.g., by assigning more weight to the minor class error,
 - Resample a dataset in order to soften or remove class imbalance
- We focus on resampling: convenient, allows to use standard classifiers
- The main aim:
 - review and compare main resampling methods,
 - compare strategies of resampling amount (i.e., how many observations to add or drop) selection,
 - explore their influence on quality of classification

- Dataset $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$
- $C_{+1}(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = +1\}$ is a major class,
- $C_{-1}(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = -1\}$ is a minor class, i.e. $|C_{+1}(S_m)| > |C_{-1}(S_m)|$
- Imbalance ratio $IR(S_m) = \frac{|C_{-1}(S_m)|}{|C_{+1}(S_m)|}$, $IR(S_m) \leq 1$

- Dataset $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$
- $C_{+1}(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = +1\}$ is a major class,
- $C_{-1}(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = -1\}$ is a minor class, i.e.
 $|C_{+1}(S_m)| > |C_{-1}(S_m)|$
- Imbalance ratio $IR(S_m) = \frac{|C_{-1}(S_m)|}{|C_{+1}(S_m)|}$, $IR(S_m) \leq 1$

- Dataset $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$
- $C_{+1}(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = +1\}$ is a major class,
- $C_{-1}(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = -1\}$ is a minor class, i.e. $|C_{+1}(S_m)| > |C_{-1}(S_m)|$
- Imbalance ratio $IR(S_m) = \frac{|C_{-1}(S_m)|}{|C_{+1}(S_m)|}$, $IR(S_m) \leq 1$

- Dataset $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$
- $C_{+1}(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = +1\}$ is a major class,
- $C_{-1}(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = -1\}$ is a minor class, i.e. $|C_{+1}(S_m)| > |C_{-1}(S_m)|$
- Imbalance ratio $IR(S_m) = \frac{|C_{-1}(S_m)|}{|C_{+1}(S_m)|}$, $IR(S_m) \leq 1$

- Learn a classifier using imbalanced training sample S_m
- The dataset S_m is resampled using a method r :
 - some observations in S_m are dropped, or
 - some new synthetic observations are added to S_m
- The result of resampling is a dataset $r(S_m)$ with $IR(r(S_m)) > IR(S_m)$
- Standard classification model f is learned on $r(S_m)$ to construct a classifier $f_{r(S_m)} : \mathbb{R}^d \rightarrow \{-1, +1\}$

- Learn a classifier using imbalanced training sample S_m
- The dataset S_m is resampled using a method r :
 - some observations in S_m are dropped, or
 - some new synthetic observations are added to S_m
- The result of resampling is a dataset $r(S_m)$ with $IR(r(S_m)) > IR(S_m)$
- Standard classification model f is learned on $r(S_m)$ to construct a classifier $f_{r(S_m)} : \mathbb{R}^d \rightarrow \{-1, +1\}$

- Learn a classifier using imbalanced training sample S_m
- The dataset S_m is resampled using a method r :
 - some observations in S_m are dropped, or
 - some new synthetic observations are added to S_m
- The result of resampling is a dataset $r(S_m)$ with $IR(r(S_m)) > IR(S_m)$
- Standard classification model f is learned on $r(S_m)$ to construct a classifier $f_{r(S_m)} : \mathbb{R}^d \rightarrow \{-1, +1\}$

- Learn a classifier using imbalanced training sample S_m
- The dataset S_m is resampled using a method r :
 - some observations in S_m are dropped, or
 - some new synthetic observations are added to S_m
- The result of resampling is a dataset $r(S_m)$ with $IR(r(S_m)) > IR(S_m)$
- Standard classification model f is learned on $r(S_m)$ to construct a classifier $f_{r(S_m)} : \mathbb{R}^d \rightarrow \{-1, +1\}$

- Performance is determined by a predefined classifier quality metrics $Q(f_{S_{train}}, S_{test})$ (e.g. AUC under Precision-Recall curve):
 - input classifier $f_{S_{train}}$,
 - testing dataset S_{test} ,
 - the higher value is the better
- M -fold cross-validation is used to estimate Q^{CV} on S_m

- Performance is determined by a predefined classifier quality metrics $Q(f_{S_{train}}, S_{test})$ (e.g. AUC under Precision-Recall curve):
 - input classifier $f_{S_{train}}$,
 - testing dataset S_{test} ,
 - the higher value is the better
- M -fold cross-validation is used to estimate Q^{CV} on S_m

Resampling method r :

① Takes input:

- dataset S_m ;
- resampling multiplier $m > 1$ for resulting imbalance ratio $IR(r(S_m)) = m \cdot IR(S_m)$;
- additional parameters, specific for the method

② Add synthesized objects to the minor class (oversampling), or drop objects from the major class (undersampling), or both

③ Outputs resampled dataset $r(S_m)$ with imbalance ratio $IR(r(S_m)) = m \cdot IR(S_m)$

Resampling method r :

- ➊ Takes input:
 - dataset S_m ;
 - resampling multiplier $m > 1$ for resulting imbalance ratio $IR(r(S_m)) = m \cdot IR(S_m)$;
 - additional parameters, specific for the method
- ➋ Add synthesized objects to the minor class (oversampling), or drop objects from the major class (undersampling), or both
- ➌ Outputs resampled dataset $r(S_m)$ with imbalance ratio $IR(r(S_m)) = m \cdot IR(S_m)$

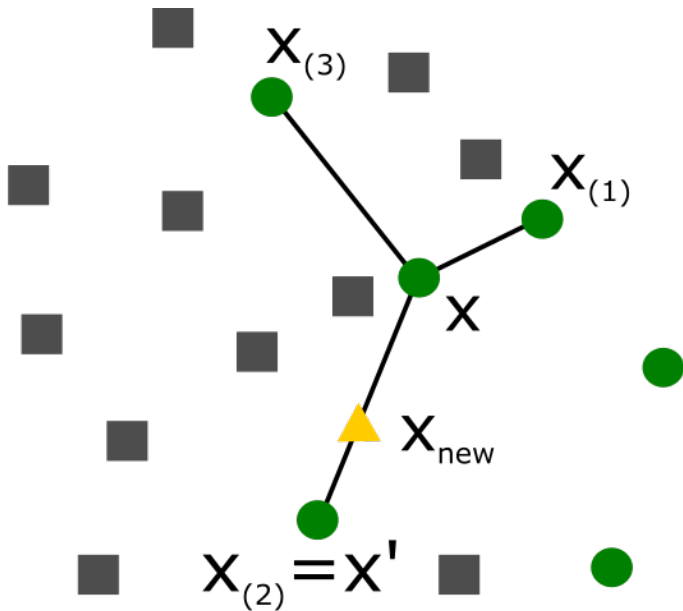
Resampling method r :

- ① Takes input:
 - dataset S_m ;
 - resampling multiplier $m > 1$ for resulting imbalance ratio $IR(r(S_m)) = m \cdot IR(S_m)$;
 - additional parameters, specific for the method
- ② Add synthesized objects to the minor class (oversampling), or drop objects from the major class (undersampling), or both
- ③ Outputs resampled dataset $r(S_m)$ with imbalance ratio $IR(r(S_m)) = m \cdot IR(S_m)$

- ROS, also known as bootstrap oversampling
- No additional input parameters
- It adds to the minor class new $(m - 1)|C_{-1}(S_m)|$ objects
- Each of objects is drawn from uniform distribution on $C_{-1}(S_m)$

- No additional input parameters
- It chooses random subset of $C_{+1}(S_m)$ with $|C_{+1}(S_m)| \frac{m-1}{m}$ elements and drops it from the dataset
- All subsets of $C_{+1}(S_m)$ have equal probabilities to be chosen

Synthetic Minority Oversampling Technique (SMOTE)



- Input parameter: k (number of neighbors)
- Oversampling, it adds to the minor class new synthesized objects
- Initialize: $S_{new} := \emptyset$. Repeat $(m - 1)|C_{-1}(S_m)|$ times:
 - 1 Select randomly $\mathbf{x}_i \in C_{-1}(S_m)$
 - 2 Find k minor class NN of \mathbf{x}_i , randomly select \mathbf{x}'_i from them
 - 3 Select randomly \mathbf{x}_{new} on the segment connecting \mathbf{x}_i and \mathbf{x}'_i
 - 4 $S_{new} := S_{new} \cup \{(\mathbf{x}_{new}, -1)\}$
- Add objects to the dataset: $r(S_m) = S_m \cup S_{new}$

- Artificial pool of data with ~ 1000 datasets
- Artificial datasets were drawn from a Gaussian mixture distribution
- Each of two classes is represented as a Gaussian mixture with not more than 3 components
- Number of features varies from 6 to 40, size of dataset from 200 to 1000, IR from 0.05 to 0.35

- Real pool of data with ~ 100 datasets
- Different areas: biology, medicine, engineering, sociology
- All features are numeric or binary, their number varies from 3 to 1000
- Size of dataset varies from 200 to 1000, IR from 0.02 to 0.75

- For each dataset we varied classifier model, resampling method and resampling multiplier
- We used Bootstrap, RUS and SMOTE with $k = 5$
- We varied resampling multiplier m from 1.25 to 10.0
- We used Decision Trees, k -Nearest Neighbors, and Logistic Regression with ℓ_1 regularization
- Optimal parameters of a classifier were selected by cross-validation

- Accuracy measure = Area under precision-recall curve Q_{PRC}
- We performed 10-fold cross-validation and calculated Q_{PRC}^{CV} — average of Q_{PRC}

- Two strategies of resampling multiplier selection:
 - equalizing strategy, EqS: select multiplier providing balanced classes ($IR = 1$) in resulting dataset
 - CV-search, CVS: select optimal multiplier (i.e., providing maximum of Q^{CV}) by cross-validation
- The equalizing strategy seems to be reasonable as it removes class imbalance which we initially tried to tackle. It is quick and widely used
- CV-search may provide better quality but it is more time-consuming

- $\{r_1, \dots, r_n\}$ — the set of considered methods (e.g. resampling methods)
- $\{S_1, \dots, S_T\}$ — the set of tasks (datasets),
- q_{ti} — the quality of the method i on the dataset t ,
- $p_i(\beta)$ is a fraction of tasks, on which the method i is worse than the best one not more than β times:

$$p_i(\beta) = \frac{1}{T} \left| \left\{ t : q_{ti} \geq \frac{1}{\beta} \max_i q_{ti} \right\} \right|, \quad \beta \geq 1$$

- $p_i(1)$ is a fraction of datasets where the method i is the best
- A graph of $p_i(\beta)$ on β is called Dolan-More curve for the method i
- The higher the curve, the better the method!

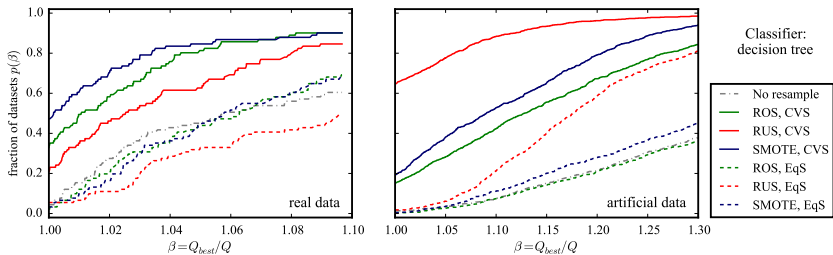


Figure – Dolan-More curves for metric Q_{PRC}^{CV}

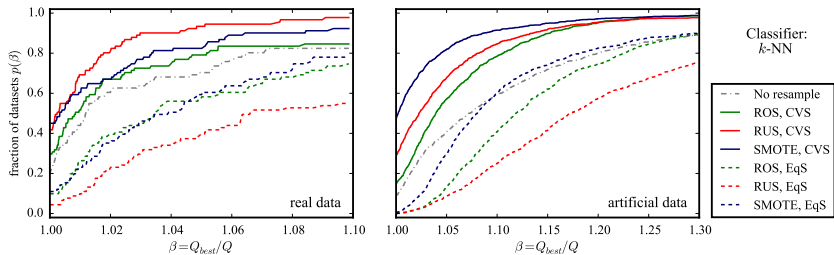


Figure – Dolan-More curves for metric Q_{PRC}^{CV}

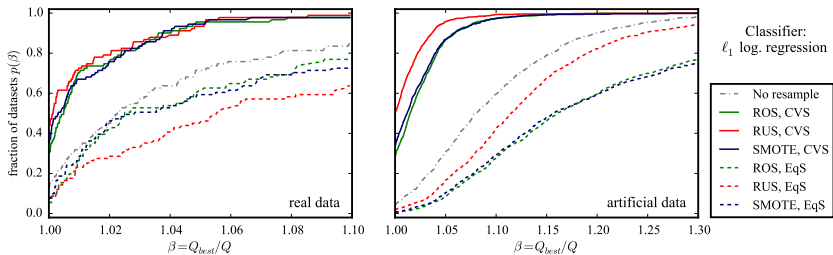


Figure – Dolan-More curves for metric Q_{PRC}^{CV}

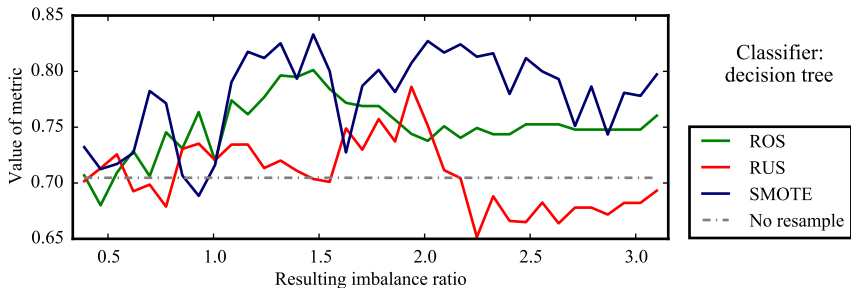


Figure – Value of Q_{PRC}^{CV} vs. resulting value of IR for dataset "Delft pump 1x3"

- Resampling with CV-search of multiplier provides better results, especially for Decision trees and Logistic regression
- The equalizing strategy (EqS) shows much lower quality, especially in case of k -Nearest neighbors and Logistic regression
- There is no method that would always outperform the others
- Classification without resampling is the best choice in some cases. E.g., for Logistic regression it is about 15% of real datasets and 5% of artificial
- Resampling improves classification of imbalanced datasets in most cases if a method and a multiplier are selected properly

1 Imbalanced Classification

2 Multi-Class Classification

3 Nonparametric Estimation

- **Training sample:** i.i.d. generated by D

$$S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \in X^m \times Y^m, y = f(\mathbf{x})$$

- mono-label case: $\text{Card}(Y) = K$
- multi-label case: $Y = \{-1, +1\}^K$
- **Problem:** find classifier $h : X \rightarrow Y$ with small generalization error
 - mono-label case: $R_D(h) = \mathbb{E}_{\mathbf{x} \sim D} [1_{h(\mathbf{x}) \neq f(\mathbf{x})}]$
 - multi-label case: $R_D(h) = \mathbb{E}_{\mathbf{x} \sim D} \left[\frac{1}{K} \sum_{j=1}^K 1_{[h(\mathbf{x})]_j \neq [f(\mathbf{x})]_j} \right]$
 - **Remark:** Precision/Recall with Micro/Macro averaging!

- **Training sample:** i.i.d. generated by D

$$S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \in X^m \times Y^m, y = f(\mathbf{x})$$

- mono-label case: $\text{Card}(Y) = K$
- multi-label case: $Y = \{-1, +1\}^K$
- **Problem:** find classifier $h : X \rightarrow Y$ with small generalization error
 - mono-label case: $R_D(h) = \mathbb{E}_{\mathbf{x} \sim D} [1_{h(\mathbf{x}) \neq f(\mathbf{x})}]$
 - multi-label case: $R_D(h) = \mathbb{E}_{\mathbf{x} \sim D} \left[\frac{1}{K} \sum_{j=1}^K 1_{[h(\mathbf{x})]_j \neq [f(\mathbf{x})]_j} \right]$
 - **Remark:** Precision/Recall with Micro/Macro averaging!

- **Training sample:** i.i.d. generated by D

$$S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \in X^m \times Y^m, y = f(\mathbf{x})$$

- mono-label case: $\text{Card}(Y) = K$
- multi-label case: $Y = \{-1, +1\}^K$
- **Problem:** find classifier $h : X \rightarrow Y$ with small generalization error
 - mono-label case: $R_D(h) = \mathbb{E}_{\mathbf{x} \sim D} [1_{h(\mathbf{x}) \neq f(\mathbf{x})}]$
 - multi-label case: $R_D(h) = \mathbb{E}_{\mathbf{x} \sim D} \left[\frac{1}{K} \sum_{j=1}^K 1_{[h(\mathbf{x})]_j \neq [f(\mathbf{x})]_j} \right]$
 - **Remark:** Precision/Recall with Micro/Macro averaging!

- Usually $K \leq 100$
- If $K \gg 1$ then some other methods are used, e.g. ranking
- Big values of K increases computational burden
- In general, classes are not balanced

- **Technique**

- for each class $k \in Y$ learn a binary classifier

$$h_k(\mathbf{x}) = \text{sign}(f_k(\mathbf{x}))$$

- combine binary classifiers via voting, e.g. majority voting

$$h : \mathbf{x} \rightarrow \arg \max_{k \in Y} f_k(\mathbf{x})$$

- **Comments**

- calibration: classifiers scores are not comparable
- simple and frequently used in practice, computational advantages in some cases

- **Technique**

- for each class $k \in Y$ learn a binary classifier

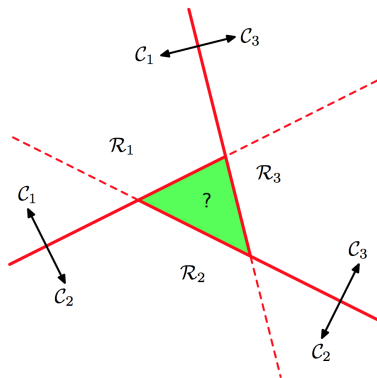
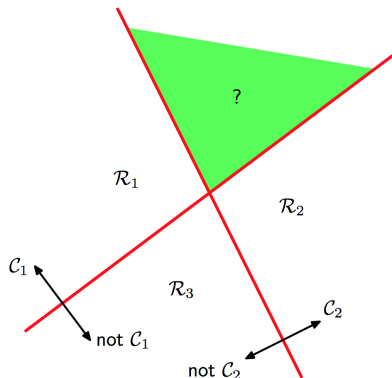
$$h_k(\mathbf{x}) = \text{sign}(f_k(\mathbf{x}))$$

- combine binary classifiers via voting, e.g. majority voting

$$h : \mathbf{x} \rightarrow \arg \max_{k \in Y} f_k(\mathbf{x})$$

- **Comments**

- calibration: classifiers scores are not comparable
- simple and frequently used in practice, computational advantages in some cases



Consider the use of $K - 1$ classifiers each of which solves a two-class problem of separating points in a particular class from points not in that class. This approach leads to regions of input space that are ambiguously classified

- **Technique**

- for each pair $(k, k') \in Y$, $k \neq k'$ learn a binary classifier $h_{k,k'} : X \rightarrow \{0, 1\}$
- combine binary classifiers via majority vote

$$h(\mathbf{x}) = \arg \max_{k' \in Y} |\{k : h_{k,k'}(\mathbf{x}) = 1\}|$$

- **Comments**

- computational complexity: train $K(K-1)/2$ binary classifiers
- overfitting: size of a training sample can be small for a given pair of classifiers

- **Technique**

- for each pair $(k, k') \in Y, k \neq k'$ learn a binary classifier $h_{k,k'} : X \rightarrow \{0, 1\}$
- combine binary classifiers via majority vote

$$h(\mathbf{x}) = \arg \max_{k' \in Y} |\{k : h_{k,k'}(\mathbf{x}) = 1\}|$$

- **Comments**

- computational complexity: train $K(K-1)/2$ binary classifiers
- overfitting: size of a training sample can be small for a given pair of classifiers

Approach based on Error-Correcting Codes

- 8 classes, codes of length 6

classes	codes					
	1	2	3	4	5	6
	1	0	0	0	1	0
	2	1	0	0	0	0
	3	0	1	1	0	1
	4	1	1	0	0	0
	5	1	1	0	0	1
	6	0	0	1	1	0
	7	0	0	1	0	0
	8	0	1	0	1	0

$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
0	1	1	0	1	1

new example x

- Assign L -long binary code word to each class, i.e. represent each class as

$$\mathbb{C} = [\mathbb{C}_{k,j}] \in \{0, 1\}^{[1,K] \times [1,L]}$$

- Learn a binary classifier $f_j : X \rightarrow \{0, 1\}$ for each column.
Example \mathbf{x} in class k is labeled with $\mathbb{C}_{k,j}$
- Classifier output:

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_L(\mathbf{x})),$$

- Final classifier

$$h : \mathbf{x} \rightarrow \arg \min_{k \in Y} d_{\text{Hamming}}(\mathbb{C}_{k,\cdot}, \mathbf{f}(\mathbf{x}))$$

- One-vs-all approach is the most widely used
- No clear empirical evidence of the superiority of other approaches
- Large structured multi-class problems are often treated as ranking problems
- Above we considered how to reduce multi-class classification to the binary case.
- Also we can incorporate a multi-class structure explicitly into a classification algorithm, see e.g. multi-class logistic regression or multi-class SVM (below)

- Linear Classifier in a multi-class case, i.e. $|Y| > 1$
- Probability of an object to belong to some class y is equal to

$$p(y|\mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w}_y^\top \mathbf{x})}{\sum_{z \in Y} \exp(\mathbf{w}_z^\top \mathbf{x})} = \text{SoftMax}_{y \in Y}(\mathbf{w}_y^\top \mathbf{x})$$

- Regularized logistic regression

$$\bar{L}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m \log p(y_i | \mathbf{x}_i, \mathbf{w}) - \frac{\lambda}{2} \sum_{y \in Y} \|\mathbf{w}_y\|^2 \rightarrow \max_{\mathbf{w}}$$

- Linear Classifier in a multi-class case, i.e. $|Y| > 1$
- Probability of an object to belong to some class y is equal to

$$p(y|\mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w}_y^\top \mathbf{x})}{\sum_{z \in Y} \exp(\mathbf{w}_z^\top \mathbf{x})} = \text{SoftMax}_{y \in Y}(\mathbf{w}_y^\top \mathbf{x})$$

- Regularized logistic regression

$$\bar{L}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m \log p(y_i|\mathbf{x}_i, \mathbf{w}) - \frac{\lambda}{2} \sum_{y \in Y} \|\mathbf{w}_y\|^2 \rightarrow \max_{\mathbf{w}}$$

- **Optimization problem**

$$\min_{\mathbf{w}, \xi} \frac{1}{2} \sum_{k=1}^K \|\mathbf{w}_k\|^2 + C \sum_{i=1}^m \xi_i$$

$$\begin{aligned} s.t. \quad & \mathbf{w}_{y_i} \mathbf{x}_i^\top - \mathbf{w}_k \mathbf{x}_i^\top + \delta_{y_i, k} \geq 1 - \xi_i, \quad \xi_i \geq 0 \\ & (i, k) \in [1, m] \times Y \end{aligned}$$

- **Decision function:**

$$h : \mathbf{x} \rightarrow \arg \max_{k \in Y} (\mathbf{w}_k \cdot \mathbf{x}^\top) = \arg \max_{k \in Y} \left(\sum_{i=1}^m \alpha_{i,k} (\mathbf{x}_i \cdot \mathbf{x}^\top) \right),$$

where $\{\alpha_{i,k}\}_{i=1}^m$, $k \in Y$ are dual variables

- Complex constraints, $m \cdot K$ size

- Naive Bayes (NB) is a conditional probability model
- Given an object to be classified, represented by features $\mathbf{x} = (x_1, \dots, x_d)$, Bayes classifier assigns probabilities

$$p(y|x_1, \dots, x_d)$$

for each of K possible classes $y \in Y = \{1, \dots, K\}$

- The conditional probability can be decomposed as

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})} \sim p(y)p(\mathbf{x}|y), y \in Y$$

- The maximum a posteriori (MAP) decision rule

$$\hat{y} = \arg \max_{y \in \{1, \dots, K\}} p(y)p(\mathbf{x}|y)$$

- How to model $p(\mathbf{x}|y)$? \Rightarrow Independence assumption!

$$p(\mathbf{x}|y) = \prod_{j=1}^d p(x_j|y)$$

- Naive Bayes (NB) is a conditional probability model
- Given an object to be classified, represented by features $\mathbf{x} = (x_1, \dots, x_d)$, Bayes classifier assigns probabilities

$$p(y|x_1, \dots, x_d)$$

for each of K possible classes $y \in Y = \{1, \dots, K\}$

- The conditional probability can be decomposed as

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})} \sim p(y)p(\mathbf{x}|y), y \in Y$$

- The maximum a posteriori (MAP) decision rule

$$\hat{y} = \arg \max_{y \in \{1, \dots, K\}} p(y)p(\mathbf{x}|y)$$

- How to model $p(\mathbf{x}|y)$? \Rightarrow Independence assumption!

$$p(\mathbf{x}|y) = \prod_{j=1}^d p(x_j|y)$$

- Naive Bayes (NB) is a conditional probability model
- Given an object to be classified, represented by features $\mathbf{x} = (x_1, \dots, x_d)$, Bayes classifier assigns probabilities

$$p(y|x_1, \dots, x_d)$$

for each of K possible classes $y \in Y = \{1, \dots, K\}$

- The conditional probability can be decomposed as

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})} \sim p(y)p(\mathbf{x}|y), y \in Y$$

- The maximum a posteriori (MAP) decision rule

$$\hat{y} = \arg \max_{y \in \{1, \dots, K\}} p(y)p(\mathbf{x}|y)$$

- How to model $p(\mathbf{x}|y)$? \Rightarrow Independence assumption!

$$p(\mathbf{x}|y) = \prod_{j=1}^d p(x_j|y)$$

- Naive Bayes (NB) is a conditional probability model
- Given an object to be classified, represented by features $\mathbf{x} = (x_1, \dots, x_d)$, Bayes classifier assigns probabilities

$$p(y|x_1, \dots, x_d)$$

for each of K possible classes $y \in Y = \{1, \dots, K\}$

- The conditional probability can be decomposed as

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})} \sim p(y)p(\mathbf{x}|y), y \in Y$$

- The maximum a posteriori (MAP) decision rule

$$\hat{y} = \arg \max_{y \in \{1, \dots, K\}} p(y)p(\mathbf{x}|y)$$

- How to model $p(\mathbf{x}|y)$? \Rightarrow Independence assumption!

$$p(\mathbf{x}|y) = \prod_{j=1}^d p(x_j|y)$$

- Naive Bayes (NB) is a conditional probability model
- Given an object to be classified, represented by features $\mathbf{x} = (x_1, \dots, x_d)$, Bayes classifier assigns probabilities

$$p(y|x_1, \dots, x_d)$$

for each of K possible classes $y \in Y = \{1, \dots, K\}$

- The conditional probability can be decomposed as

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})} \sim p(y)p(\mathbf{x}|y), y \in Y$$

- The maximum a posteriori (MAP) decision rule

$$\hat{y} = \arg \max_{y \in \{1, \dots, K\}} p(y)p(\mathbf{x}|y)$$

- How to model $p(\mathbf{x}|y)$? \Rightarrow Independence assumption!

$$p(\mathbf{x}|y) = \prod_{j=1}^d p(x_j|y)$$

- The joint model

$$p(y|\mathbf{x}) \sim p(y) \prod_{j=1}^d p(x_j|y), y \in Y$$

- The maximum a posteriori (MAP) decision rule

$$\hat{y} = \arg \max_{y \in \{1, \dots, K\}} p(y) \prod_{j=1}^d p(x_j|y)$$

- Nonparametric estimation for $p(x_j|y)$, $y \in Y$!

- The joint model

$$p(y|\mathbf{x}) \sim p(y) \prod_{j=1}^d p(x_j|y), y \in Y$$

- The maximum a posteriori (MAP) decision rule

$$\hat{y} = \arg \max_{y \in \{1, \dots, K\}} p(y) \prod_{j=1}^d p(x_j|y)$$

- Nonparametric estimation for $p(x_j|y)$, $y \in Y$!

- The joint model

$$p(y|\mathbf{x}) \sim p(y) \prod_{j=1}^d p(x_j|y), y \in Y$$

- The maximum a posteriori (MAP) decision rule

$$\hat{y} = \arg \max_{y \in \{1, \dots, K\}} p(y) \prod_{j=1}^d p(x_j|y)$$

- Nonparametric estimation for $p(x_j|y)$, $y \in Y$!

1 Imbalanced Classification

2 Multi-Class Classification

3 Nonparametric Estimation

- $S_m = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \sim F$ is a given sample, $\mathbf{x} \in \mathbb{R}^1$
- $F(\mathbf{x})$ is an absolutely continuous CDF with an unknown density $p(\mathbf{x})$
- We estimate $p(\mathbf{x})$ in the point \mathbf{x} , i.e. construct $\hat{p}_m(\mathbf{x}) = \hat{p}_m(\mathbf{x}|S_m)$
- Typical parametric assumption

$$p \in \{p(\mathbf{x}; \boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}, \Theta \subset \mathbb{R}^p$$

- Now we do not use such assumption \Rightarrow Nonparametric Estimation

- $S_m = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \sim F$ is a given sample, $\mathbf{x} \in \mathbb{R}^1$
- $F(\mathbf{x})$ is an absolutely continuous CDF with an unknown density $p(\mathbf{x})$
- We estimate $p(\mathbf{x})$ in the point \mathbf{x} , i.e. construct $\hat{p}_m(\mathbf{x}) = \hat{p}_m(\mathbf{x}|S_m)$
- Typical parametric assumption

$$p \in \{p(\mathbf{x}; \boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}, \Theta \subset \mathbb{R}^p$$

- Now we do not use such assumption \Rightarrow Nonparametric Estimation

- $S_m = \{\mathbf{x}_1, \dots, \mathbf{x}_m\} \sim F$ is a given sample, $\mathbf{x} \in \mathbb{R}^1$
- $F(\mathbf{x})$ is an absolutely continuous CDF with an unknown density $p(\mathbf{x})$
- We estimate $p(\mathbf{x})$ in the point \mathbf{x} , i.e. construct $\hat{p}_m(\mathbf{x}) = \hat{p}_m(\mathbf{x}|S_m)$
- Typical parametric assumption

$$p \in \{p(\mathbf{x}; \boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}, \Theta \subset \mathbb{R}^p$$

- Now we do not use such assumption \Rightarrow Nonparametric Estimation

- We estimate $\hat{p}_m(\mathbf{x}) \forall \mathbf{x} \in \mathbb{R}^1$

Definition

Mean Integrated Squared Error:

$$MISE(\hat{p}_m, p) = \mathbb{E}_p \left[\int_{\mathbb{R}} (\hat{p}_m(\mathbf{x}) - p(\mathbf{x}))^2 d\mathbf{x} \right]$$

Histogram

- We consider interval $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \in [a, b)$ and divide it into N equal bins Δ_i of size $h = \frac{b-a}{N}$:

$$\Delta_i = [a + ih, a + (i + 1)h), i = 0, 1, \dots, N - 1$$

- Let ν_i be a number of data points, belonging to Δ_i

Definition

$$\hat{p}_m(\mathbf{x}) = \begin{cases} \frac{\nu_0}{mh}, & \mathbf{x} \in \Delta_0, \\ \dots & \\ \frac{\nu_{N-1}}{mh}, & \mathbf{x} \in \Delta_{N-1}; \end{cases} = \frac{1}{mh} \sum_{i=0}^{N-1} \nu_i 1\{\mathbf{x} \in \Delta_i\}$$

- For $\mathbf{x} \in \Delta_j$ and small h :

$$\mathbb{E}_p \hat{p}_m(\mathbf{x}) = \frac{\mathbb{E}_p \nu_j}{mh} = \frac{\int_{\Delta_j} p(\mathbf{z}) d\mathbf{z}}{h} \approx \frac{p(\mathbf{x})h}{h} = p(\mathbf{x})$$

At the same time

$$\text{Var}(\hat{p}_m(\mathbf{x})) = \frac{\int_{\Delta_j} p(\mathbf{z}) d\mathbf{z} \left(1 - \int_{\Delta_j} p(\mathbf{z}) d\mathbf{z}\right)}{mh^2}$$

Histogram

- We consider interval $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \in [a, b)$ and divide it into N equal bins Δ_i of size $h = \frac{b-a}{N}$:

$$\Delta_i = [a + ih, a + (i + 1)h), i = 0, 1, \dots, N - 1$$

- Let ν_i be a number of data points, belonging to Δ_i

Definition

$$\hat{p}_m(\mathbf{x}) = \begin{cases} \frac{\nu_0}{mh}, & \mathbf{x} \in \Delta_0, \\ \dots & \\ \frac{\nu_{N-1}}{mh}, & \mathbf{x} \in \Delta_{N-1}; \end{cases} = \frac{1}{mh} \sum_{i=0}^{N-1} \nu_i 1\{\mathbf{x} \in \Delta_i\}$$

- For $\mathbf{x} \in \Delta_j$ and small h :

$$\mathbb{E}_p \hat{p}_m(\mathbf{x}) = \frac{\mathbb{E}_p \nu_j}{mh} = \frac{\int_{\Delta_j} p(\mathbf{z}) d\mathbf{z}}{h} \approx \frac{p(\mathbf{x})h}{h} = p(\mathbf{x})$$

At the same time

$$\text{Var}(\hat{p}_m(\mathbf{x})) = \frac{\int_{\Delta_j} p(\mathbf{z}) d\mathbf{z} \left(1 - \int_{\Delta_j} p(\mathbf{z}) d\mathbf{z}\right)}{mh^2}$$

- We consider interval $\{\mathbf{x}_1, \dots, \mathbf{x}_m\} \in [a, b)$ and divide it into N equal bins Δ_i of size $h = \frac{b-a}{N}$:

$$\Delta_i = [a + ih, a + (i + 1)h), i = 0, 1, \dots, N - 1$$

- Let ν_i be a number of data points, belonging to Δ_i

Definition

$$\hat{p}_m(\mathbf{x}) = \begin{cases} \frac{\nu_0}{mh}, & \mathbf{x} \in \Delta_0, \\ \dots & \\ \frac{\nu_{N-1}}{mh}, & \mathbf{x} \in \Delta_{N-1}; \end{cases} = \frac{1}{mh} \sum_{i=0}^{N-1} \nu_i 1\{\mathbf{x} \in \Delta_i\}$$

- For $\mathbf{x} \in \Delta_j$ and small h :

$$\mathbb{E}_p \hat{p}_m(\mathbf{x}) = \frac{\mathbb{E}_p \nu_j}{mh} = \frac{\int_{\Delta_j} p(\mathbf{z}) d\mathbf{z}}{h} \approx \frac{p(\mathbf{x})h}{h} = p(\mathbf{x})$$

At the same time

$$\text{Var}(\hat{p}_m(\mathbf{x})) = \frac{\int_{\Delta_j} p(\mathbf{z}) d\mathbf{z} \left(1 - \int_{\Delta_j} p(\mathbf{z}) d\mathbf{z}\right)}{mh^2}$$

- We can prove that mean integrated squared error

$$MISE(\hat{p}_m, p) \approx \left(\int_{\mathbb{R}} [p'(\mathbf{x})]^2 d\mathbf{x} \right) \frac{h^2}{12} + \frac{1}{mh}$$

- The bigger h we use the bigger bias and smaller variance we get, and vice versa: Bias-Variance Tradeoff
- Too big h = oversmoothing, too small h = undersmoothing
- Thus for a histogram with optimal h , we get that

$$MISE = O(m^{-\frac{2}{3}})$$

- We can prove that mean integrated squared error

$$MISE(\hat{p}_m, p) \approx \left(\int_{\mathbb{R}} [p'(\mathbf{x})]^2 d\mathbf{x} \right) \frac{h^2}{12} + \frac{1}{mh}$$

- The bigger h we use the bigger bias and smaller variance we get, and vice versa: Bias-Variance Tradeoff
- Too big h = oversmoothing, too small h = undersmoothing
- Thus for a histogram with optimal h , we get that

$$MISE = O(m^{-\frac{2}{3}})$$

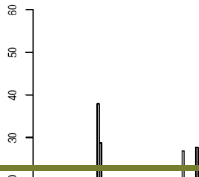
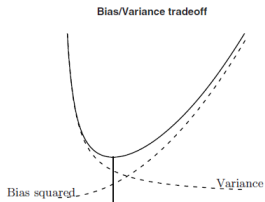
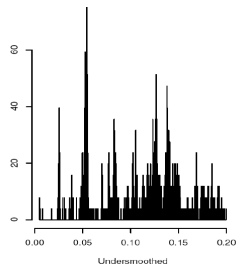
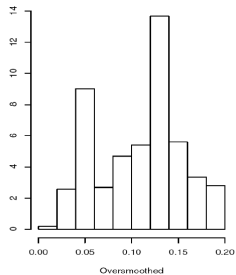
- We can prove that mean integrated squared error

$$MISE(\hat{p}_m, p) \approx \left(\int_{\mathbb{R}} [p'(\mathbf{x})]^2 d\mathbf{x} \right) \frac{h^2}{12} + \frac{1}{mh}$$

- The bigger h we use the bigger bias and smaller variance we get, and vice versa: Bias-Variance Tradeoff
- Too big h = oversmoothing, too small h = undersmoothing
- Thus for a histogram with optimal h , we get that

$$MISE = O(m^{-\frac{2}{3}})$$

Smoothing Selection: Example



KDE allows to get smoother estimate (compared to histogram based ones) with faster convergence rates

Definition

Kernel is a function K , such that

$$K(\mathbf{x}) \geq 0, \int_{\mathbb{R}} K(\mathbf{x}) d\mathbf{x} = 1, \int_{\mathbb{R}} \mathbf{x} K(\mathbf{x}) d\mathbf{x} = 0, \sigma_K^2 \equiv \int_{\mathbb{R}} \mathbf{x}^2 K(\mathbf{x}) d\mathbf{x} < \infty$$

Examples

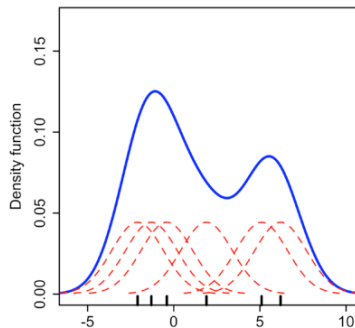
- ◀ $K(x) = \frac{1}{2} \mathbb{I}\{|x| < 1\}$ — rectangular kernel
- ◀ $K(x) = (1 - |x|) \mathbb{I}\{|x| < 1\}$ — triangle kernel
- ◀ $K(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2})$ — Gaussian kernel
- ◀ $K(x) = \frac{3}{4} (1 - x^2) \mathbb{I}\{|x| < 1\}$ — Epanechnikov kernel

Definition

KDE has the form

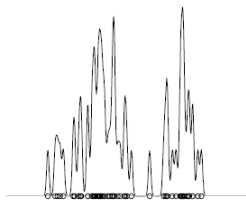
$$\hat{p}_m(\mathbf{x}) = \frac{1}{mh} \sum_{i=1}^m K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right),$$

h is a kernel width

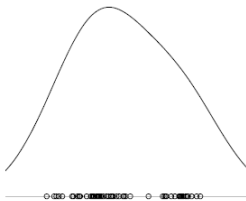


Kernel width selection: Example

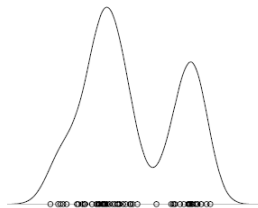
Undersmoothing



Oversmoothing



Correct smoothing



- Multidimensional case, i.e. $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$ is a point in \mathbb{R}^d .
- Thus i -th observation is an d -dimensional vector:

$$\mathbf{x}_i = (x_{i1}, \dots, x_{id})$$

- Let $h = (h_1, \dots, h_d)$ be a vector of kernel widths
- Then

$$\hat{p}_m(\mathbf{x}) = \frac{1}{mh_1 \cdot \dots \cdot h_d} \sum_{i=1}^m \left[\prod_{j=1}^d K\left(\frac{x_j - x_{ij}}{h_j}\right) \right]$$

- Optimal rate of convergence is $O(m^{-\frac{4}{4+d}})$: if d increases convergence rate decreases
- Let us consider a table with values of m necessary to get the mean squared estimation error in $\mathbf{x}_0 = 0$ less than 0.1 depending on d for a multidimensional normal density and optimal kernel width:

d	1	2	3	4	5	6	7	8	9
m	4	19	67	223	768	2790	10700	43700	187000

- Here d is a dimension, m is a sample size

- Optimal rate of convergence is $O(m^{-\frac{4}{4+d}})$: if d increases convergence rate decreases
- Let us consider a table with values of m necessary to get the mean squared estimation error in $\mathbf{x}_0 = 0$ less than 0.1 depending on d for a multidimensional normal density and optimal kernel width:

d	1	2	3	4	5	6	7	8	9
m	4	19	67	223	768	2790	10700	43700	187000

- Here d is a dimension, m is a sample size

- Let us consider m observations: $S_m = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, generated from a joint density $p(\mathbf{x}, y)$
- These observations are generated by the model

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$

where ε_i is an i.i.d white noise, $\mathbb{E}\varepsilon_i = 0$, $\mathbb{V}(\varepsilon_i) = \sigma^2$

- We should estimate a regression function

$$f(\mathbf{x}) = \mathbb{E}(y|\mathbf{x}) = \int_{\mathbb{R}} yp(y|\mathbf{x})dy = \frac{\int_{\mathbb{R}} yp(\mathbf{x}, y)dy}{\int_{\mathbb{R}} p(\mathbf{x}, y)dy} = \frac{\int_{\mathbb{R}} yp(\mathbf{x}, y)dy}{p(\mathbf{x})}$$

- Let us consider m observations: $S_m = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, generated from a joint density $p(\mathbf{x}, y)$
- These observations are generated by the model

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$

where ε_i is an i.i.d white noise, $\mathbb{E}\varepsilon_i = 0$, $\mathbb{V}(\varepsilon_i) = \sigma^2$

- We should estimate a regression function

$$f(\mathbf{x}) = \mathbb{E}(y|\mathbf{x}) = \int_{\mathbb{R}} yp(y|\mathbf{x})dy = \frac{\int_{\mathbb{R}} yp(\mathbf{x}, y)dy}{\int_{\mathbb{R}} p(\mathbf{x}, y)dy} = \frac{\int_{\mathbb{R}} yp(\mathbf{x}, y)dy}{p(\mathbf{x})}$$

- Let us consider m observations: $S_m = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, generated from a joint density $p(\mathbf{x}, y)$
- These observations are generated by the model

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$

where ε_i is an i.i.d white noise, $\mathbb{E}\varepsilon_i = 0$, $\mathbb{V}(\varepsilon_i) = \sigma^2$

- We should estimate a regression function

$$f(\mathbf{x}) = \mathbb{E}(y|\mathbf{x}) = \int_{\mathbb{R}} yp(y|\mathbf{x})dy = \frac{\int_{\mathbb{R}} yp(\mathbf{x}, y)dy}{\int_{\mathbb{R}} p(\mathbf{x}, y)dy} = \frac{\int_{\mathbb{R}} yp(\mathbf{x}, y)dy}{p(\mathbf{x})}$$

Definition

Let K denote a kernel, and

- $\hat{p}_m(\mathbf{x})$ be a kernel density estimate, obtained using $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$,
- $\hat{p}_m(\mathbf{x}, y)$ be a kernel density estimate, obtained using $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$.

If $\hat{p}_m(\mathbf{x}) \neq 0$, then

$$\hat{f}_m^{NW}(\mathbf{x}) = \frac{\int_{\mathbb{R}} y \hat{p}_m(\mathbf{x}, y) dy}{\hat{p}_m(\mathbf{x})}$$

We can notice that Nadaraya-Watson estimate can be used also in case when $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ are some fixed and deterministic values, e.g. $\mathbf{x}_i = \frac{i}{m}$

We use Nadaraya-Watson estimate for $f(\mathbf{x})$:

Definition

Nadaraya-Watson estimate has the form

$$\hat{f}_m^{NW}(\mathbf{x}) = \sum_{i=1}^m w_i(\mathbf{x}) y_i,$$

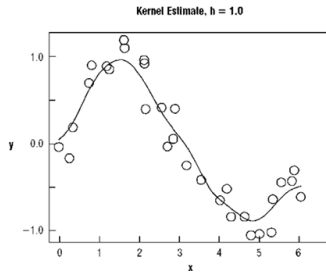
where

$$w_i = \frac{K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)}{\sum_{j=1}^m K\left(\frac{\mathbf{x}-\mathbf{x}_j}{h}\right)},$$

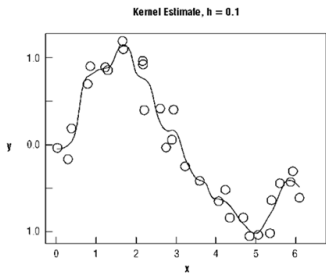
and K is a some kernel function

Thus, the estimate is a weighted sum of y_i , where any point, close to \mathbf{x} , has a big weight

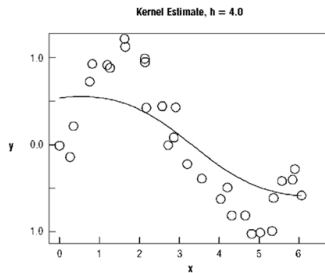
Nonparametric regression: Example



Correct smoothing



Undersmoothing



Oversmoothing

Definition

Bias: $\text{bias}(\mathbf{x}_0) = \mathbb{E}_p \hat{p}_m(\mathbf{x}_0) - p(\mathbf{x}_0)$

We get the following decomposition

Lemma

$$\begin{aligned} MSE(\hat{p}_m, p, \mathbf{x}_0) &= \text{bias}^2(\mathbf{x}_0) + \mathbb{V}_p(\hat{p}_m(\mathbf{x}_0)) = \\ &= [\mathbb{E}_p \hat{p}_m(\mathbf{x}_0) - p(\mathbf{x}_0)]^2 + \mathbb{E}_p [\hat{p}_m(\mathbf{x}_0) - \mathbb{E}_p \hat{p}_m(\mathbf{x}_0)]^2 \end{aligned}$$

Lemma

$$MISE(\hat{p}_m, p) = \int_{\mathbb{R}} \text{bias}^2(\mathbf{x}) d\mathbf{x} + \int_{\mathbb{R}} \mathbb{V}_p(\hat{p}_m(\mathbf{x})) d\mathbf{x}$$

We will use these statements when constructing optimal density estimates

A shape of K influence quality of estimate not so significant compared to a value of h

Theorem

$$MISE(\hat{p}_m, p) \approx \frac{1}{4} \sigma_K^4 h^4 \int_{\mathbb{R}} (p''(\mathbf{x}))^2 d\mathbf{x} + \frac{1}{mh} \int_{\mathbb{R}} (K(\mathbf{x}))^2 d\mathbf{x}$$

For an optimal kernel width we get that $MISE(\hat{p}_m, p) = O\left(m^{-\frac{4}{5}}\right)$

- Again we can not calculate h^* in practice, since it depends on unknown values of $r(\mathbf{x})$, $p(\mathbf{x})$
- Then we should minimize the risk estimate w.r.t. h

$$\hat{\mathcal{J}}(h) = \sum_{i=1}^m (y_i - \hat{r}_{(-i)}^{NW}(\mathbf{x}_i))^2,$$

where $\hat{r}_{(-i)}^{NW}$ is a Nadaraya-Watson estimate, constructed using the sample, from which the observation (\mathbf{x}_i, y_i) is excluded

Theorem

$$\hat{\mathcal{J}}(h) = \sum_{i=1}^m (y_i - \hat{r}^{NW}(\mathbf{x}_i))^2 \frac{1}{\left(1 - \frac{K(0)}{\sum_{j=1}^m K\left(\frac{\mathbf{x}_i - \mathbf{x}_j}{h}\right)}\right)^2}$$