Data preprocessing, model complexity.

Victor Kitov

v.v.kitov@yandex.ru

Yandex
School of Data Analysis.

Table of Contents

Hyperparameters selection

- Using CV we can select hyperparameters of the model[1]
- Each model has hyperparameter, corresponding to model complexity.
- Model complexity - ability to reproduce training set.
- Examples:
  - regression: # of features d, e.g. $x, x^2, ...x^d$
  - K-NN: number of neighbors K

[1] can we use CV loss in this case as estimation for future losses?

Underfitted and overfitted models[2]

Too simple (underfitted) model

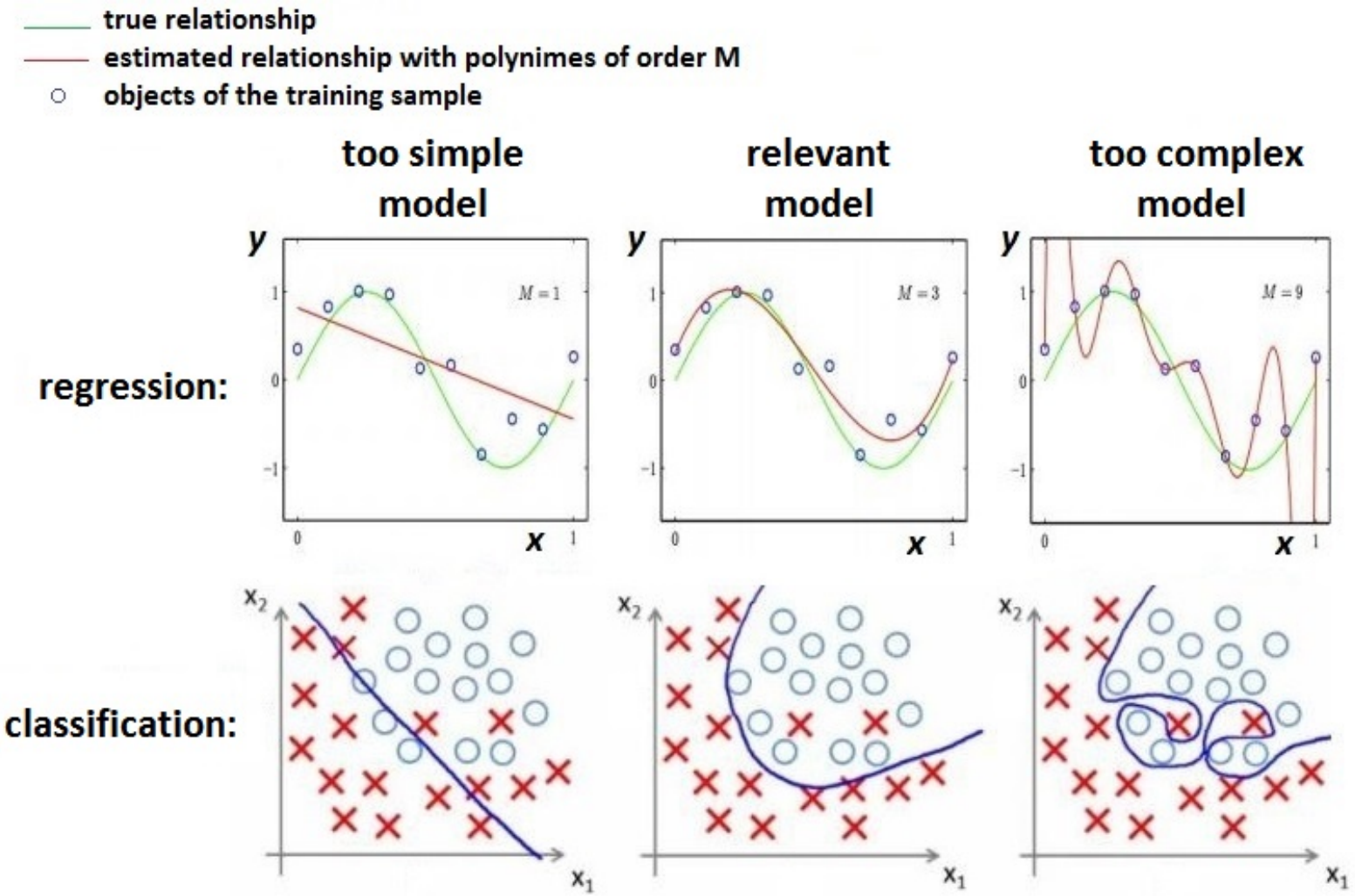
Model that oversimplifies true relationship $\mathcal{X} \to \mathcal{Y}$.

Too complex (overfitted) model

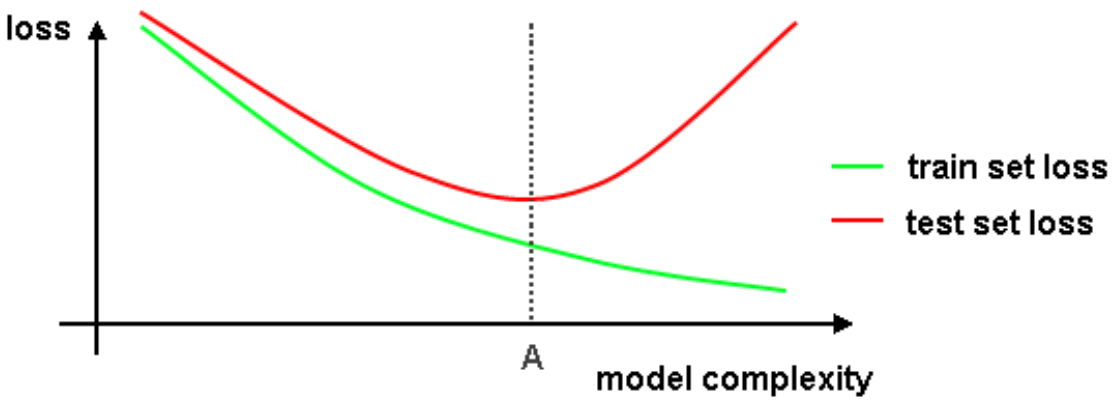
Model that is too tuned on particular peculiarities (noise) of the training set instead of the true relationship $\mathcal{X} \to \mathcal{Y}$.

[2] In fact most models overfit, meaning that empirical risk<expected risk. Underfitted models just have lower difference than overfitted ones.

Examples of overfitted / underfitted models



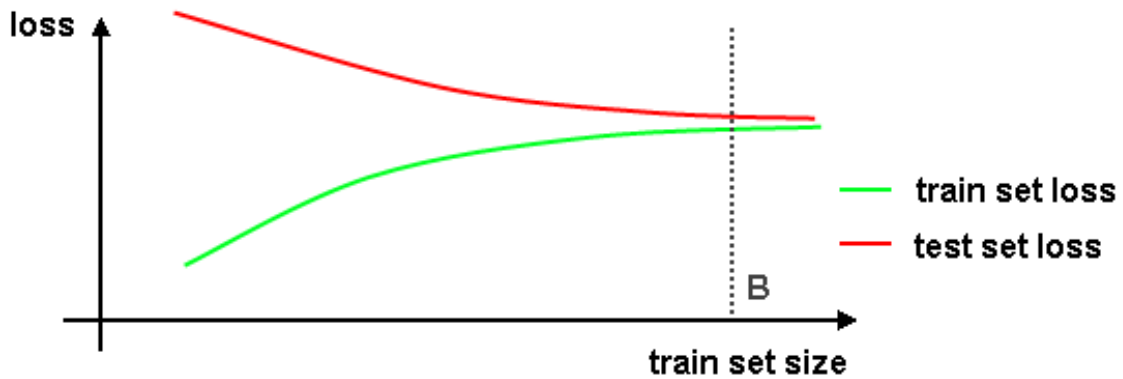
Loss vs. model complexity



Comments:

- expected loss on test set is always higher than on train set.
- left to A: model too simple, underfitting, high bias
- right to A: model too complex, overfitting, high variance

Loss vs. train set size



Comments:

- expected loss on test set is always higher than on train set.
- right to B there is no need to further increase training set size
  - useful to limit training set size when model fitting is time consuming

Table of Contents

What we need to do

- Data preprocessing:
  - deal with missing data
  - clean incorrect data
  - data subsampling
  - data scaling
  - data type transformtion

Missing data

What we can do with missing features:

- **remove all objects, having at least one missing feature**
  - easiest way, but lose information
- **fill missing features using most likely value**
  - mean, median for numeric features
  - averaged neighbours for continious time-series
  - mode for categorical feature
- **predict missing features using known features**
  - regression task for numeric features
  - classification task for categorical feature
- **use models, which ignore missing features**
  - such as decision trees

Comments on imputation

- imputing missing features with estimates induces imputation bias
  - to get rid of this bias: for feature d add binary feature, indicating whether this feature was known or was imputed.
- imputation implies that feature absence and feature value are independent
  - may not be the case
    - in surveys people prefer not to tell their salary when it is big.
  - if they are dependent additional expert info should be used for feature reconstruction

Incorrect data

We can detect incorrect data using:

- inconsistency detection
  - e.g. surname of the same person is spelled differently in different records
- domain knowledge
  - e.g. human height cannot be 4 meters
- statistical methods: remove outliers

Outlier removal, having extreme values

- 1D outlier removal:

[3] which of these measures are robust to outliers and why?
[4] which of these measures are robust to outliers and why?

Outlier removal, having extreme values

- 1D outlier removal:
  - outliers are outside $[center - \alpha scatter, center + \alpha scatter]$, $\alpha > 0$
  - center[3]: mean, median
  - scatter[4]: standard deviation, 95% quantile - 5%quantile, median $\{|x - \text{median}\{x\}\}$
- Outliers can be not errors, but interesting regimes:
  - medical data: rare disease
    - network data: hacker attack
    - card transaction data: fraud
  - manual inspection of outliers needed

[3] which of these measures are robust to outliers and why?
[4] which of these measures are robust to outliers and why?

2 Data preprocessing
- Missing data
- Data reduction
- Normalization of features
- Feature type transformations

Objects reduction

If N is too large, then

- additional memory/disk/CPU data transfer requirements
- slow-down of optimization in ML methods

Objects reduction:

- **Purely random subsampling**
  - usually without replacement
- **Random with stratification**
  - stratification by output
    - preserve classes distribution
  - stratification by feature value
    - preserve object types distribution

Objects reduction

Objects reduction:

- **Random non-uniform sampling** (less common)
  - sample harder objects more (mistakes)
    - regression: based on $|f_\theta(x) - y|$
    - classification: based on margin (few etalons, then lowest margin objects)
  - sample rare classes/objects more (underrepresented data)
  - sample new objects more (in dynamic context)

Margin

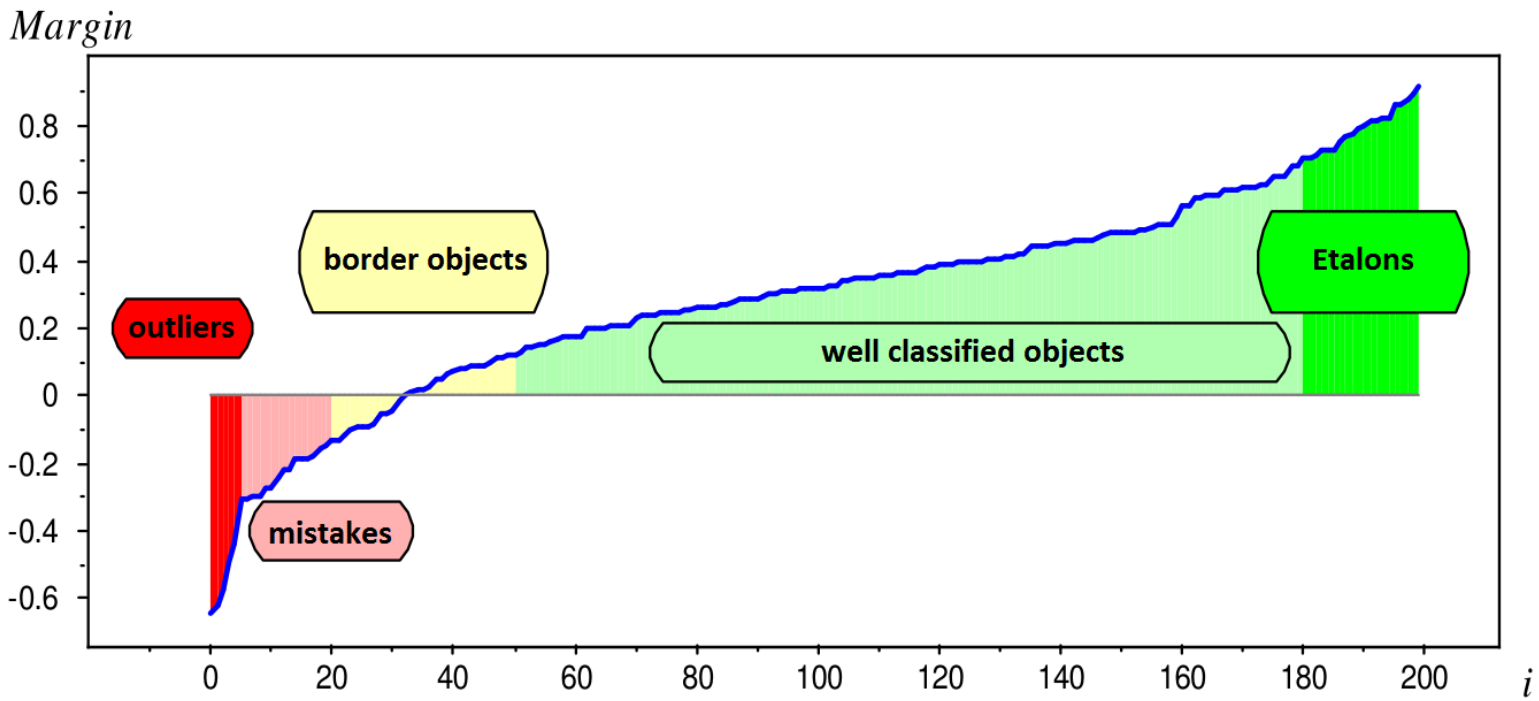
- Consider
  - $(x_1, y_1), (x_2, y_2), ...(x_N, y_N)$ - training set
  - $Y = \{1, 2, ...C\}$ - set of all possible classes.
- Define margin:

$$M(x_i, y_i) = g_{y_i}(x_i) - \max_{c \in \mathbf{Y} \setminus \{\mathbf{y_i}\}} g_c(x_i)$$

- Margin shows the preference of the algorithm to assign x_i to the correct class y_i compared to other classes.
  - being continious margin is more informative than $\mathbb{I}[\widehat{y}(x_i) = y_i]$
- Margin is negative <=> object x_i was incorrectly classified.

Categorization of objects based on margin

Estimate margin for each object in the training set and order objects by margin:



Good classifier should:

- minimize the number of negative margin region
- classify correctly with high margin

Margin for binary classification

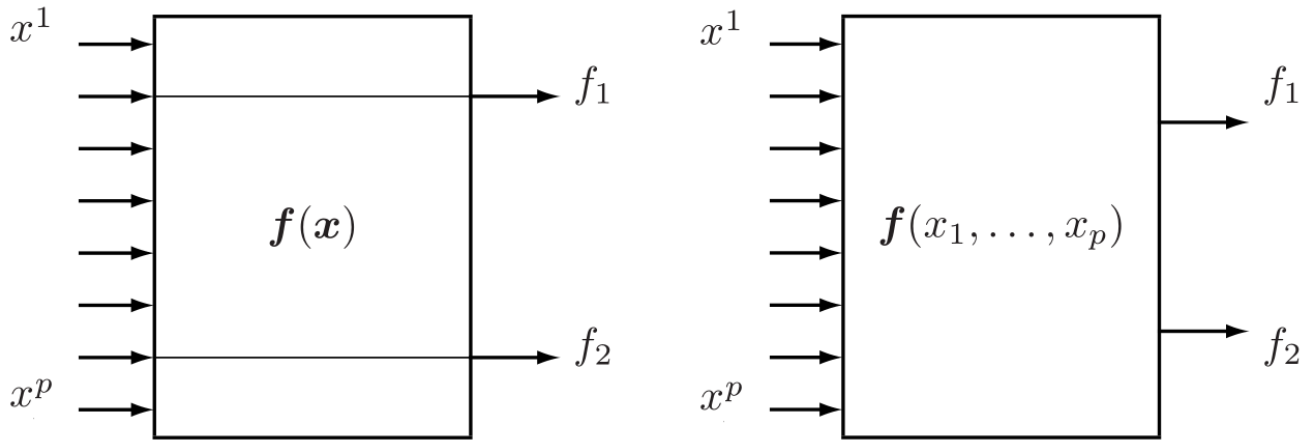
Consider

- $y \in \{+1, -1\}$
- $g(\mathbf{x}) = g_{+1}(\mathbf{x}) - g_{-1}(\mathbf{x})$ - score of positive class versus negative.
- $\widehat{y}(\mathbf{x}) = \text{sign}\, g(\mathbf{x})$

  $M(\mathbf{x}, y) = g_y(\mathbf{x}) - g_{-y}(\mathbf{x}) = y\left(g_{+1}(\mathbf{x}) - g_{-1}(\mathbf{x})\right) = yg(\mathbf{x})$

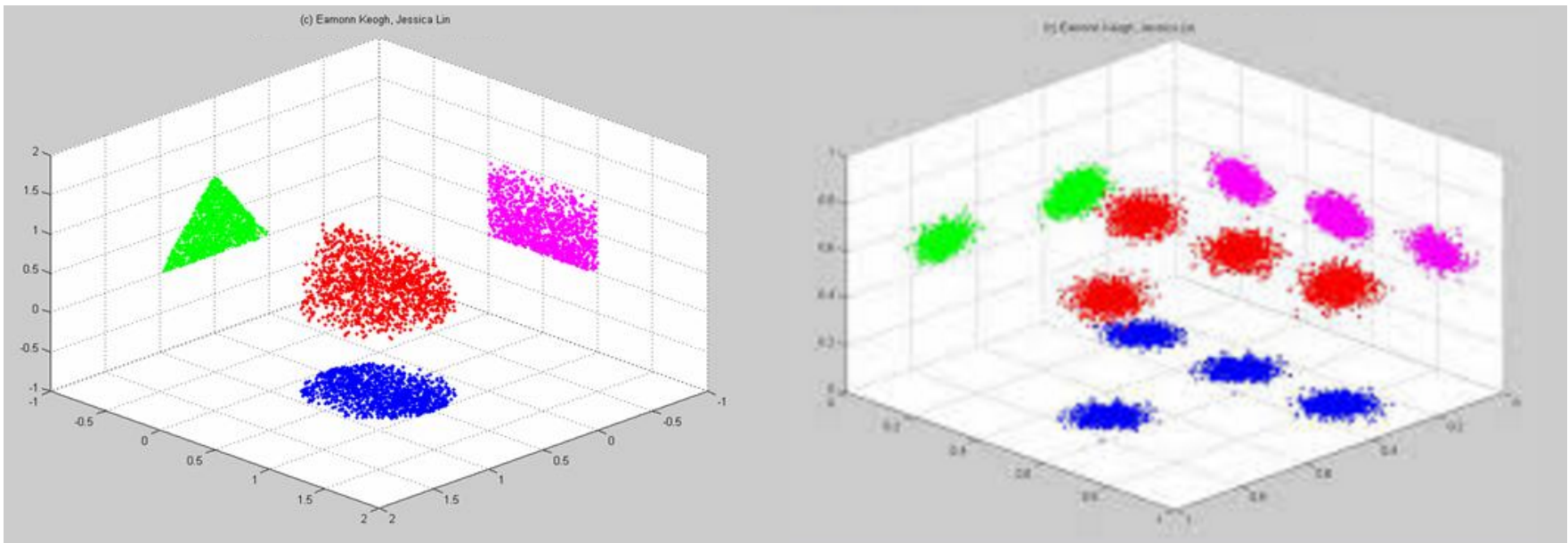
Feature reduction

- Feature selection vs. feature extraction:



- Feature selection:
  - unsupervised (e.g. variance<threshold)
  - filter (e.g. by correlation with output)
  - wrapper (e.g. compare performance with/without feature)
  - embedded inside ML model

Brute-force feature selection may lead to information loss



2 Data preprocessing
- Missing data
- Data reduction
- Normalization of features
- Feature type transformations

Normalization of features

- Feature scaling may affect ML model, e.g. K-NN.
- If different features should have equal impact - make their scatter common.
- If some features should be more important - give them higher scatter.
- Typical normalizations:

Name	Transformation	Properties of resulting feature
Autoscaling	$x'_j = \frac{x_j - \mu_j}{\sigma_j}$	zero mean and unit variance.
Range scaling	$x'_j = \frac{x_j - L_j}{U_j - L_j}$	belongs to $[0, 1]$ interval.

where μ_j, σ_j, L_j, U_j are mean value, standard deviation, minimum and maximum value of the j-th feature.

Discussion

- For non-negative features range scaling does not affect feature sparsity: 0->0.
- Autoscaling is more robust to outliers than range scaling[5]

[5] Propose a scaling that is fully robust to outliers.

Normalization of features

- Non-linear transformations incorporating features with rare large values:
  - $x_i' = \log(x_i)$
  - $x_i' = x^p$, $0 \leq p < 1$
- For $F_i(\alpha) = P(x^i \leq \alpha)$ transformation $\tilde{x}^i \rightarrow F_i(x^i)$ will give feature uniformly distributed on $[0,1]$[6].
- Sometimes scaling is performed not per feature, but per object $x \rightarrow x/\|x\|$
  - e.g.: x- counts of words within document->frequencies of words within document
    documents of different length become comparable!

[6] Prove that

2 Data preprocessing
- Missing data
- Data reduction
- Normalization of features
- Feature type transformations

Possible features types

- **Numeric**
  - salary
  - flat size
- **Categorical**
  - occupation (programmer, manager, engineer, etc.)
  - sex (male, female)
  - city (Moscow, Kaluga, etc.)
- **Binary** (may be considered both numeric and categorical)
  - sex
  - marital status

Numeric->categorical

- **Numeric->categorical** (discretization):
  - Split feature domain into intervals $[b_1, b_2]$, $[b_2, b_3]$, ...$[b_K, b_{K+1}]$
  - $f \to \tilde{f} \in \mathbb{R}^K$

$$\tilde{f} = (\mathbb{I}[f \in [b_1, b_2]], \mathbb{I}[f \in [b_2, b_3]], ...\mathbb{I}[f \in [b_K, b_{K+1}]])^T$$

- Loose some information.
- Such representation may be better for the model (e.g. linear regression).
- Intervals selection:
  - equiwidth (equal length of each interval)
  - equidepth (equal density of points in each interval)
    - sort feature values in increasing order and set new border every $\frac{N}{K}$ objects.

Categorical->numeric

- **One hot encoding** - encode categorical feature $f \in \{c_1, c_2, ...c_K\}$ with $\tilde{f} \in \mathbb{R}^K$

$$\tilde{f} = (\mathbb{I}[f = c_1], \mathbb{I}[f = c_2], ... \mathbb{I}[f = c_K])^T$$

Original data:	
id	Color
1	White
2	Red
3	Black
4	Purple
5	Gold

One-hot encoding format:					
id	White	Red	Black	Purple	Gold
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

Categorical->numeric

Probabilistic encoding - replace discrete feature f with aggregated another feature g.

- Continious g:
  - replace f with $average(g|f)$
- Discrete $g \in \{1, 2, ...C\}$:
  - replace f with C binary features $p(g = 1|f)$, $p(g = 2|f)$, ... $p(g = C|f)$

g may be taken as output y.

- intuitive method but overfits
  - e.g. consider f=client id, unique for each object.
- to prevent overfitting calculate aggregation statistics on **separate training set**.

Summary

- Each model has complexity parameter - tune it!
- Data preprocessing is important and includes the following steps:
  - deal with missing data
  - clean incorrect data
  - data subsampling
  - data scaling
  - data type transformation
    - one-hot and aggregation encodings are most important.