# Data preprocessing, model complexity.

Victor Kitov

v.v.kitov@yandex.ru

# 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**

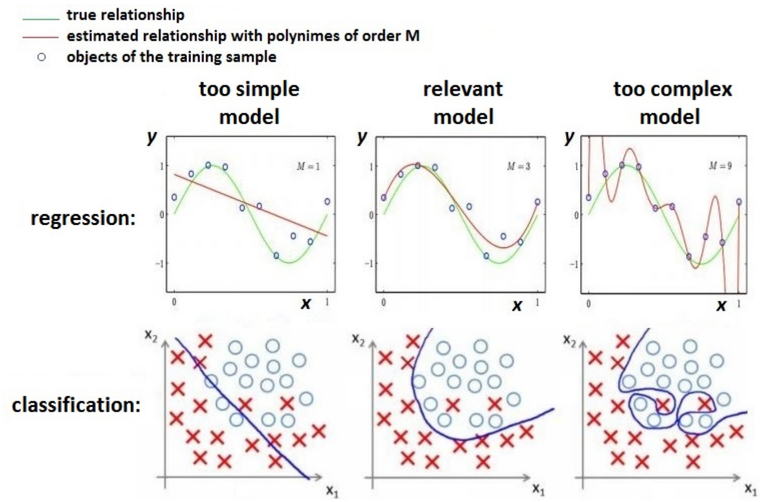
Model that oversimplifies true relationship $\mathcal{X} \to \mathcal{Y}$.

**Too complex (overfitted) model**

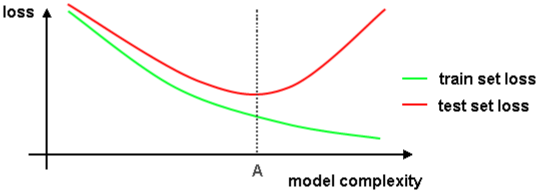
Model that is too tuned on particular peculiarities (noise) of the training set instead of the true relationship $\mathcal{X} \to \mathcal{Y}$.

[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



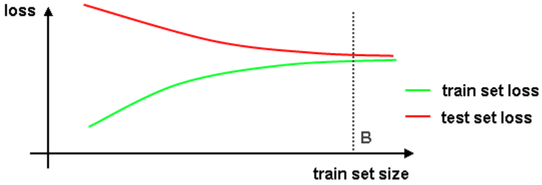
# Loss vs. model complexity




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 transformation

# 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 continuous 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

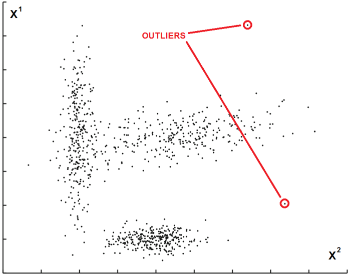
# Comments on imputation

- 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:

- consistency check across different databases
    - 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, $\text{median}\{|x - \text{median}\{x\}\}$
- Outliers can be not errors, but interesting regimes:
  - manual inspection of outliers needed
  - examples:
    - medical data: rare disease
    - network data: hacker attack
    - card transaction data: fraud

[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
    - Data transformation
    - Feature type transformations

# Objects reduction

If $N$ is too large, then

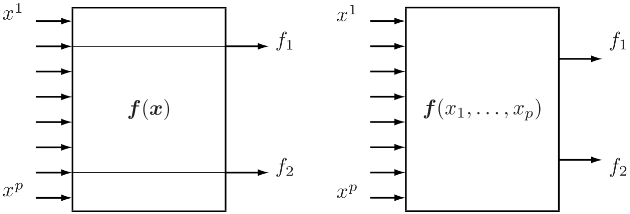
- additional disk/memory/CPU data transfer requirements
- slow-down of optimization in ML methods

Objects reduction:

- random uniform
    - purely random subsampling
    - random with stratification
        - stratification by output (or feature) value
          to preserve output (or object types) distribution
- random non-uniform
    - sample new objects more (in dynamic context)
    - sample rare classes/objects more (underrepresented data)
    - sample harder objects more (mistakes)

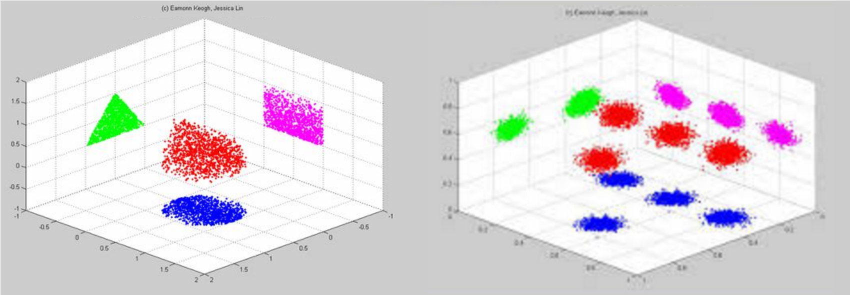
# Feature reduction

- 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
   - Data transformation
   - Feature type transformations

# Normalization of features

- Feature scaling may affect ML model, e.g. K-NN.
- Need equal features impact - make their scatter common.
- Make some features more important - increase their scatter.
- Typical scaling operators:

| Name | Transformation | Properties of result |
|---|---|---|
| Standardization | $u' = \frac{x_j - \text{mean}(u)}{\text{std}(u)}$ | mean=0, std=1 |
| Min-max normalization | $u' = \frac{u - \min(u)}{\max(u) - \min(u)}$ | $\in [0,1]$, 0->0 for sparse data |
| Average normalization | $u' = \frac{u - \text{mean}(u)}{\max(u) - \min(u)}$ | zero mean, range=1 |

- All operators aren't robust to outliers. Propose robust variants.

# Non-linear feature transformations

- Feature with skewed distribution with large rare values:

$$u' = \log(1+u), \qquad u' = u^p,\, 0 \le p < 1$$

- For uniformly distributed output ($F(\cdot)$-c.d.f. of $u$)

$$u' = F(u)$$

- For normally distributed output ($\Phi^{-1}(\cdot)$-inverse function to c.d.f. of $\mathcal{N}(0,1)$)[5].

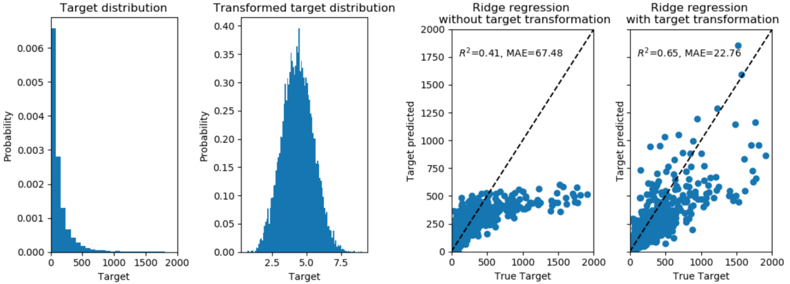
$$u' = \Phi^{-1}(F(u))$$

- Object normalization $x' \to x/\left\|x\right\|, \quad x \in \mathbb{R}^D$.
    - when feature ratios are more important than absolute values.
    - example:
        - $x$ - counts of words within document
        - $x'$ - frequencies of words within document
        - documents of different length become comparable!

[5] Prove that. See sklearn.preprocessing.QuantileTransformer.

# Transformation of output[6]

$$y' = \ln(1+y)$$

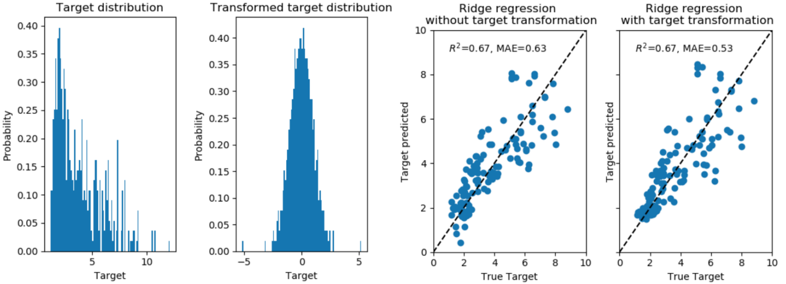



$$MAE = \frac{1}{N}\sum_{n=1}^{N}|\widehat{y}_n - y_n|, \qquad R^2 = 1 - \frac{(1/N)\sum_{n=1}^{N}|\widehat{y}_n - y_n|}{(1/N)\sum_{n=1}^{N}|\,\mathrm{mean}(y_n) - y_n|}$$

[6] See scikit-learn demo.

# Transformation of output[7]

$y' = \Phi^{-1}(F(y))$, $F(\cdot)$- c.d.f. of $y$, $\Phi^{-1}(\cdot)$-inverse function to c.d.f. of $\mathcal{N}(0,1)$.




$$MAE = \frac{1}{N}\sum_{n=1}^{N}|\widehat{y}_n - y_n|, \qquad R^2 = 1 - \frac{(1/N)\sum_{n=1}^{N}|\widehat{y}_n - y_n|}{(1/N)\sum_{n=1}^{N}|\,\text{mean}(y_n) - y_n|}$$

[7] See scikit-learn demo.

# Possible features types

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

# Numeric->categorical (discretization)

1. Split feature domain into intervals $[b_1, b_2], [b_2, b_3], ...[b_K, b_{K+1}]$
2. $u \to u' \in \mathbb{R}^K$

$$u' = (\mathbb{I}[u \in [b_1, b_2]], \mathbb{I}[u \in [b_2, b_3]], ...\mathbb{I}[u \in [b_K, b_{K+1}]])$$

- Loose some information.
- Makes model pay attention to special groups (e.g. by age - students, working, pensioners).
- Intervals selection:
  - equal length of each interval
  - equal density of points in each interval

# Categorical->numeric

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

$$u' = (\mathbb{I}[u = c_1], \mathbb{I}[u = c_2], \dots \mathbb{I}[u = c_K])$$

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

- Original $u$ is then replaced by $u'$, total number of features increases by $K-1$.

# Categorical->numeric

**Mean value 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**.

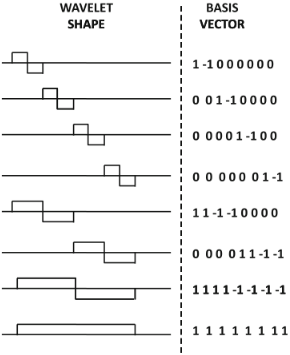
# Time series / spatial -> numeric

- **Time series->numeric:**
  - use discrete wavelet transform (DWT).
- **Spatial data->numeric:**
  - use discrete wavelet transform (DWT).

# Haar wavelet transform

- Suppose we have time series $f(t)$, $t = 1, 2, \dots T$.
  - e.g. temperature measurements from sensor every second
- How can we get compact description of $f(t)$?

- Consider the following set of basis functions $\phi_k(t)$ (Haar wavelets):
- They are orthogonal $\langle \phi_i, \phi_j \rangle = \sum_t \phi_i(t)\phi_j(t) = 0 \quad \forall i \neq j$.
- Represent $f_t = \sum_{k=1}^{K} a_k \phi_k(t)$, so $f_t \to (a_1, a_2, \dots a_K)$.



# Finding wavelet coefficients

Finding wavelet coefficients:

1. Set first coefficient to $\frac{1}{T}\sum_t f(t)$
2. repeat until given resolution achieved:
    1. next wavelet coefficient = 0.5*(difference between average value of time series value on 1st halve and 2nd halves)
    2. recursively apply this approach to 1st and send half of time series

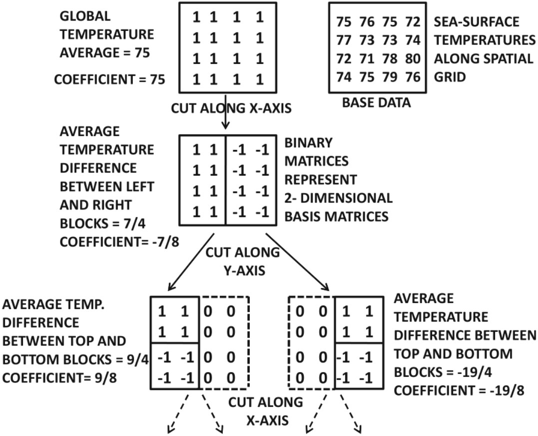
# Dimensionality reduction with wavelets

$$f_t = \sum_{k=1}^{K} a_k \phi(t) = \sum_{k=1}^{K} a_k \left\| \phi_k(t) \right\| \frac{\phi_k(t)}{\left\| \phi_k(t) \right\|}$$

- $\frac{\phi_k(t)}{\left\| \phi_k(t) \right\|}$ are orthonormal, so comparable.
- Leave only coefficients, having $a_k \left\| \phi_k(t) \right\| > \textit{threshold}$.
- When have $P$ time series simultaneously, we can
  - leave coefficients for $\phi_k(t)$ that are on average important for all time series
  - or leave coefficients for each time series independently, set other to 0, get sparse matrix.
    - then we can get economical representation of this matrix with SVD decomposition.

# Wavelets for spatial data

Top levels of the wavelet decomposition for spatial data



# Other transformations

- **Discrete sequence->numeric:**
  1. for each $t$ replace $f_t$ with one-hot encoded $\tilde{f}_t \in \mathbb{R}^K$
  2. for each $k = 1, 2, ...K$: apply wavelet transform to each binary time series $\tilde{f}_t^k$
  3. append wavelet coefficients into single vector representation.
- **Any type->set of numeric points $y_1, ...y_N$, $y_i \in \mathbb{R}^K$:**
  - solve *multidimensional scaling* problem:

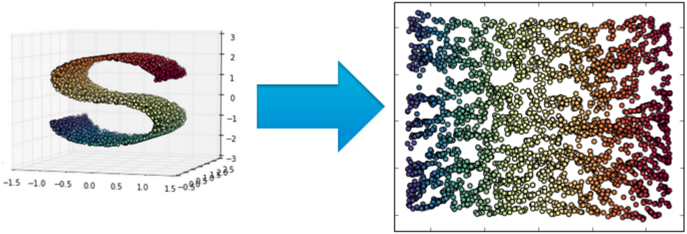
$$\sum_{i,j:\, i>j} (\rho(x_i, x_j) - \|y_i - y_j\|)^2 \to \min_{y_1,...y_N}$$

# Other transformations

- **Time series->discrete sequence:**
  1. Consider time series $f_t$, $t$-time.
  2. Divide time into windows of equal size, $f_t$ $\rightarrow$averaged value on each window
  3. Discretize averaged values using equiwidth or equiwidth discretization.
- **Any type->graph:**
  - each object is represented by a node
  - connection between $x_i, x_j$ exists <=> $x_i, x_j$ are sufficiently close:
    $\rho(x_i, x_j) < threshold$
    $x_i, x_j$ are belong to $K$ nearest neighbours of each other.
  - weight of connection:

$$w_{ij} = e^{-\gamma\rho(x_i,x_j)^2}$$

# Why we may need graphs




Distance along the graph may be useful.

# 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.