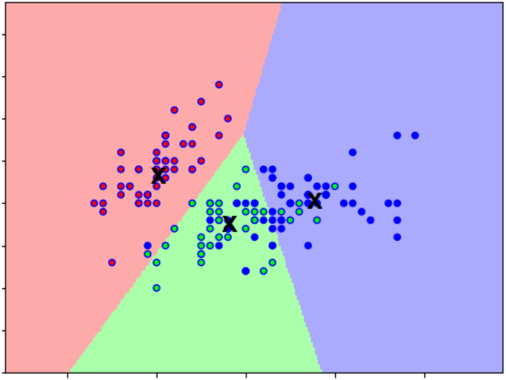
ML Summary

Victor Kitov

v.v.kitov@yandex.ru

Nearest centroids algorithm

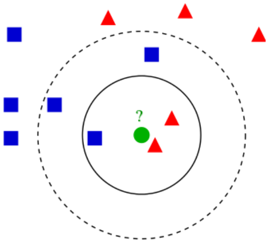
Decision boundaries for 3-class nearest centroids



K-NN (parameters: K, distance metric [may be tuned])

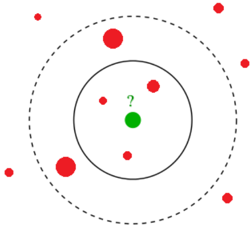
Classification:

1. Find k closest objects to the predicted object x in the training set.
2. Associate x the most frequent class among its k neighbours.



Regression:

1. Find k closest objects to the predicted object x in the training set.
2. Associate x average output of its k neighbours.



Feature transformation

- **One hot encoding**: $f \to (\mathbb{I}[f = c_1], \mathbb{I}[f = c_2], \dots \mathbb{I}[f = c_K])$

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

- **Mean value encoding**: feature $f \to average(y|f)$ (account for overfitting, may average some other feature)

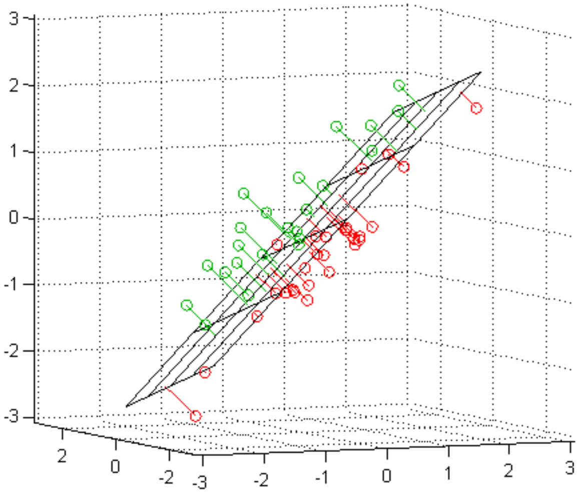
Feature scaling

- Feature scaling change feature importance in most ML methods.
- Standardization:

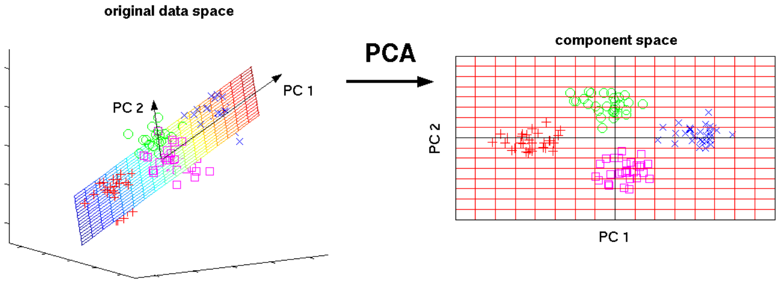
Name	Transformation	Properties of result
Standardization	$u' = \frac{x_j - \text{mean}(u)}{\text{std}(u)}$	mean=0, std=1
Min-max normalization	$u' = \frac{u - \min(u)}{\max(u) - \min(u)}$	$\in [0, 1]$, 0->0 for sparse data
Average normalization	$u' = \frac{u - \text{mean}(u)}{\max(u) - \min(u)}$	zero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

Apply non-linear transformations to features and output:
$x \to \phi(x), \quad y \to \chi(y)$ to change distribution.

Principal components form plane of best fit

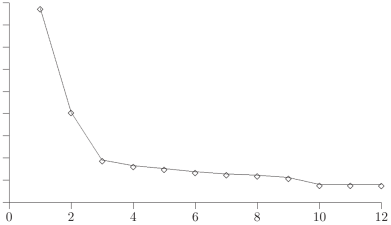


Data visualization



Explained variance ratio

Take most significant components until their explained variance ratio falls sharply down:

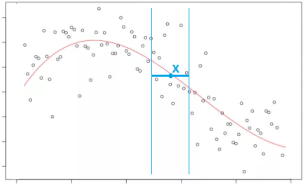


Linear regression

$$f(x, \beta) = x^T \beta$$

$\sum_{n=1}^{N} \left(x_n^T \beta - y_n\right)^2 \to \min_\beta$	linear regression
$\sum_{n=1}^{N} \left(x_n^T \beta - y_n\right)^2 + \lVert\beta\rVert_2^2 \to \min_\beta$	ridge regression
$\sum_{n=1}^{N} \left(x_n^T \beta - y_n\right)^2 + \lVert\beta\rVert_1 \to \min_\beta$	LASSO regression
$\sum_{n=1}^{N} \left(\phi(x_n)^T \beta - y_n\right)^2 + \lVert\beta\rVert_1 \to \min_\beta$	transformed features
$\sum_{n=1}^{N} (f_\beta(x_n) - y_n)^2 \to \min_\beta$	non-linear extension
$\sum_{n=1}^{N} \mathcal{L}(x_n^T \beta - y_n) \to \min_\beta$	different loss-function

Nadaraya-Watson regression



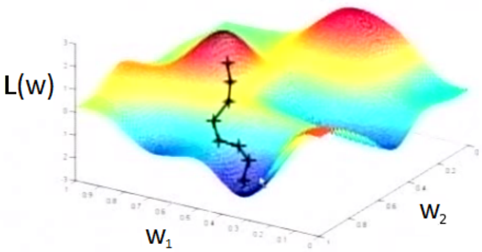
$$\widehat{y}(x)=\frac{\sum_{i=1}^{N}y_iw_i(x)}{\sum_{i=1}^{N}w_i(x)}=\frac{\sum_{i=1}^{N}y_iK\left(\frac{\rho(x,x_i)}{h(x)}\right)}{\sum_{i=1}^{N}K\left(\frac{\rho(x,x_i)}{h(x)}\right)}$$

Gradient descend optimization

- Gradient/stochastic gradient descend - iterative movement in steepest descent of the function :

$$w_{t+1} := w_t - \varepsilon_t \frac{1}{N} \sum_{i=1}^{N} \nabla_w \mathcal{L}(x_i, y_i | w_n)$$

$$w_{t+1} := w_t - \varepsilon_t \frac{1}{K} \sum_{n \in I} \nabla_w \mathcal{L}(x_n, y_n | w_t)$$



Gradient descend optimization

- If $\mathcal{L}(u)$-convex => $L(w)$-convex => local optimum is global optimum, $L'(w) = 0 \iff w$ - global minimum.
- SGD requires $\varepsilon_t \to 0$ for $t \to \infty$
- Modifications: momentum, Nesterov momentum, AdaGrad, RMSProp, Adam.

Classifiers

From binary to multiclass classification:

$\widehat{y}(x) = \arg\max_c g_c(x)$	general multiclass
$\widehat{y}(x) = \arg\max_c w_c^T x$	linear multiclass
$\widehat{y}(x) = \text{sign}\, g(x)$	general binary
$\widehat{y}(x) = \text{sign}\, w^T x$	linear binary

- one-vs-all
- one-vs-one

Margin

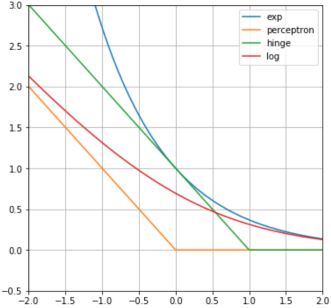
Quality of classification is evaluated with margin (higher is better).

$g_y(x) - \max_{c \neq y} g_c(x)$	general multiclass
$w_y^T x - w_c^T x$	linear multiclass
$yg(x)$	general binary
$yw^T x$	linear binary

Margin

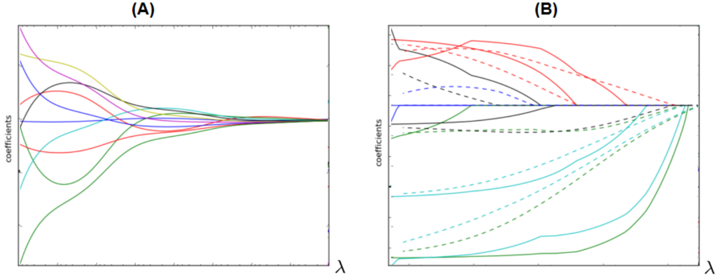
$$\sum_{n=1}^{N} \mathcal{L}\left(M_n\right) = \sum_{n=1}^{N} \mathcal{L}\left(w^T x_n y_n\right) \to \min_w$$

Common loss functions:



Regularization effect

(A) - L_2^2 regilarization, (B)- L_1 regularization (selects features).



Logistic regression (can predict class probabilities)

Binary logistic regression:

$$p(y|x) = \sigma(y\langle w, x\rangle)$$

Multiclass logistic regression:

$$p(\omega_c|x) = softmax(w_c^T x|x_1^T x, ...x_C^T x) = \frac{exp(w_c^T x)}{\sum_i exp(w_i^T x)}$$

Corresponds to linear classifier with logistic loss
$\mathcal{L}(M) = \ln(1 + e^{-M})$

Support vector machines

- Linear classifier, estimated with hinge loss and L_2 regularization:

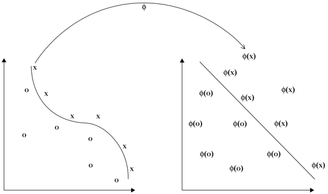
$$\frac{1}{2C}\|w\|_2^2 + \sum_{i=1}^{N}[1 - M_i(w, w_0)]_+ \to \min_{w,w_0}$$

- Maximizes border between classes:

(a) (b)

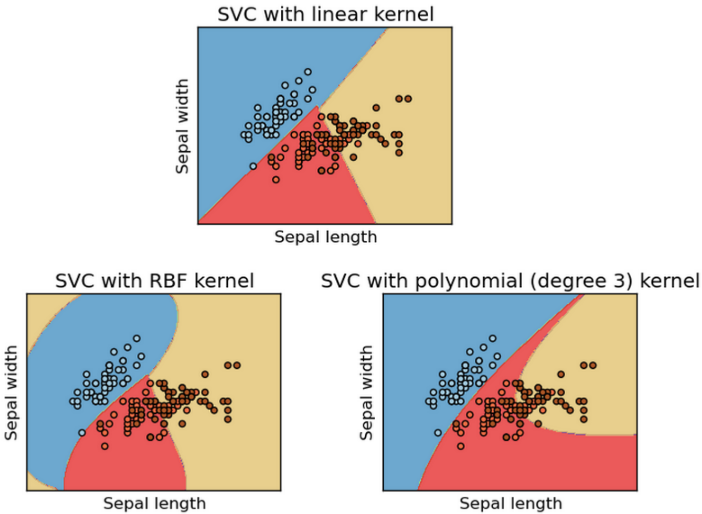
Kernel generalization

$$x \to \phi(x)$$
$$\langle x, x' \rangle \to \langle \phi(x), \phi(x') \rangle = K(x, x')$$



Kernel	Mathematical form
linear	$\langle x, z \rangle$
polynomial	$(\alpha \langle x, z \rangle + \beta)^M$
RBF	$\exp(-\gamma \lVert x - z \rVert^2)$

SVM allows kenrel generalization



Binary classifier evaluation

Error rate	$(TP+TN)\,/\,(P+N)$
Precision	$\frac{TP}{\widehat{P}}$
Recall (=TPR)	$\frac{TP}{P}$
F-measure	$\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$
Weighted F-measure	$\frac{1}{\frac{\beta^2}{1+\beta^2}\frac{1}{Precision}+\frac{1}{1+\beta^2}\frac{1}{Recall}}$

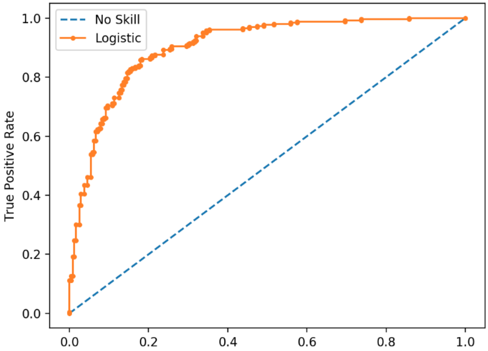
Probability evaluation: $\prod_{i=1}^{N}\widehat{p}(y_i|x_i)$ or $\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{p}_n-\widehat{\mathbf{p}}_n\|^2$

ROC curve

- Classification with variable eagerness to assign $\widehat{y} = +1$:

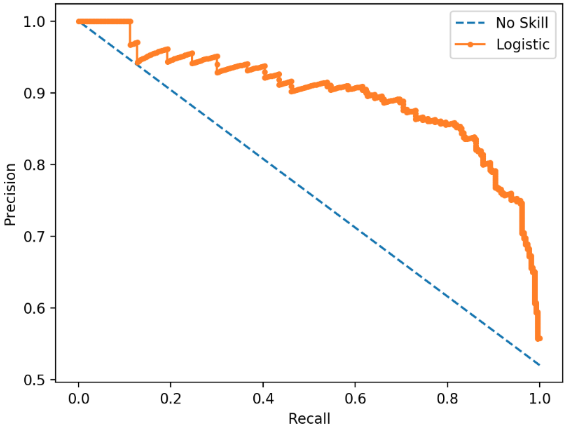
$$\widehat{y}(x) = \text{sign}(g(x) - \alpha)$$

- $TPR = TPR(\alpha)$, $FPR = FPR(\alpha)$.
- ROC curve is a function TPR(FPR):

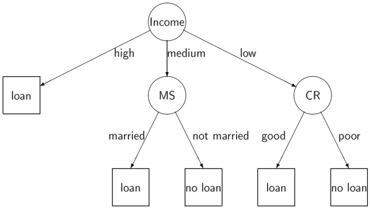


Precision-recall curve: precision(recall)

$$Precision = \frac{TP}{\widehat{P}} \qquad Recall = \frac{TP}{P}$$



Decision trees

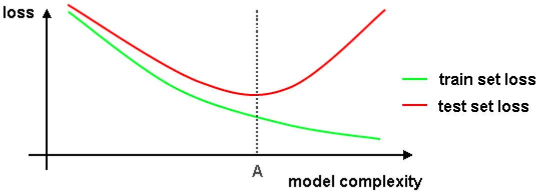


$$\Delta I(t) = I(t) - I(t_L)\frac{N(t_L)}{N(t)} - I(t_R)\frac{N(t_R)}{N(t)} \qquad \text{quality of split}$$

$$\widehat{i}_t,\ \widehat{h}_t = \arg\max_{i,h} \Delta I(t) \qquad \text{maximal quality improvement}$$

- At leaves: mean (regression), mode (classification) for symmetric costs $\mathbb{I}[\widehat{y} \neq y]$
- Tree depth: rule based +pruning.

Loss vs. model complexity



- left to A: simple model, underfitting, high bias
- right to A: complex model, overfitting, high variance

Bias-variance decomposition:

$$\mathbb{E}_{X,Y,\varepsilon}\{[\widehat{f}(x) - y(x)]^2\} = \left(\mathbb{E}_{X,Y}\{\widehat{f}(x)\} - f(x)\right)^2 + \mathbb{E}_{X,Y}\left\{[\widehat{f}(x) - \mathbb{E}_{X,Y}\widehat{f}(x)]^2\right\} + \mathbb{E}\varepsilon^2$$

Ambiguity decomposition

For $F(x)=\sum_{m=1}^{M}w_m f_m(x)$, $w_m\ge 0$, $\sum_m w_m=1$. Then

$$\underbrace{\left(F(x)-y\right)^2}_{\text{ensemble error}}=\underbrace{\sum_m w_m\left(f_m(x)-y\right)^2}_{\text{base learner error}}-\underbrace{\sum_m w_m\left(f_m(x)-F(x)\right)^2}_{\text{ambiguity}}$$

Ensemble may be accurate even when base models are inaccurate, but **disagree with each other**.

- take base models of different types
- train each base model on its own subset of objects and features.

Solving overfitting/overfitting of base models

- Solve overfitting:
  - use simple aggregation model (average, majority voting)
    - e.g. averaging the same model over random training set subsets=bagging
  - average trees on random subsets of features/thresholds=RandomForest/ExtraRandomTrees.
- Solve underfitting:
  - use aggregation model with additional trainable parameters
    - train $f_1(x), ...f_M(x)$ and $G(f_1(x), ...f_M(x))$ on separate sets of objects (stacking)
  - adjust next base model to correct mistakes of previous ensemble (boosting)

Boosting

$$F_M(x) = f_0(x) - c_1 f_1(x) + ... - c_M f_M(x)$$

Regression: $\widehat{y}(x) = F_M(x)$
Binary classification: $score(y|x) = F_M(x)$, $\widehat{y}(x) = \text{sign}\, F_M(x)$
$(f_m(x), c_m)$ are found one after another:

$$(c_m, f_m) := \arg\min_{f,c} \sum_{n=1}^{N} \mathcal{L}(F_{m-1}(x_n) - cf(x_n),\, y_n)$$

For binary classification with $\mathcal{L}(yF(x)) = e^{-yF(x)}$ can solve explicitly (AdaBoost).

Gradient boosting

Gradient descent

$$L(w) \to \min_{w}$$
$$w_m = w_{m-1} - \varepsilon_{m-1} \nabla L(w_{m-1})$$
$$w_M = w_0 - \varepsilon_0 \nabla L(w_0) - \varepsilon_1 \nabla L(w_1) - ... - \varepsilon_{M-1} \nabla L(w_{M-1})$$

Gradient boosting:

$$L(F(x), y) \to \min_{F(x)}$$
$$F_m(x) = F_{m-1}(x) - \varepsilon_{m-1} \nabla f_{m-1}(x)$$
$$F_M(x) = f_0(x) - \varepsilon_0 f_1(x) - \varepsilon_1 f_2(x) - ... - \varepsilon_{M-1} f_{M-1}(x)$$
$$f_i(x) \approx \nabla_F L(F_{i-1}(x), y)$$

Gradient boosting extensions

- Finetune predictions at every leaf of the decision tree.
- Shrinkage - add next base model with smaller weight.
  - increases #base models

$$F_m(x) = F_{m-1}(x) - \alpha\varepsilon_m f_m(x)$$

- Subsampling - train next base learner on subsample of objects and features
  - increases diversity of base models (ambiguity decomposition)
  - increases fitting speed and #base models
- Use second order (quadratic) Taylor approximation of $L(F(x), y)$.
- Tie impurity functions of decision trees to the final loss $L(F(x), y)$.
  - last 2 ideas implemented in xgBoost

Data preparation

- Feature scaling is important.
- Try different encodings of categorical features.
- Feature engineering is important.
  - visualize distributions of x^d, y, (x^i, x^j), (x^i, y).
  - visualize your whole data in 2D using PCA
- Clear outliers, think of extended features, helping model to predict.
- Think of non-linear transformations $\phi(x), \chi(y)$ to make dependency more vivid.

Model fitting

- Start from linear model.
    - quick baseline
    - coefficients reveal dependency in general
    - *may be the best for high feature dimensionality*
- Fit RandomForest to get more
    - *quick way to get advanced baseline*
    - analyze most important features
    - especially good for
        - data of different types
        - non-linear dependencies
- Fit boosting (xgBooost or other extensions)
    - use subsambling & shrinkage
    - accurately adjust #base models
    - *best in many cases, but slowly tuned.*

Model fitting

- Natural measure
- of dissimilarity - try K-NN.
    - of similarity - try kernel trick.
- Finally fit an ensemble (stacking) comprising diverse set of models.
- Analyze poorly predicted objects. What makes them different from the rest?
    - generate additional features
    - or handle them with a separate model
- Use separate validation set for evaluation.
- Extensive usage of the same validation set may cause overfitting.
- Use test set only once to report final accuracy.