Multiclass classification

Multiclass strategies

- Reduce to binary

One-vs-All

$$Y = \{1, \ldots, K\}$$

$$foreach\, k\, in\, Y:$$

$$\quad z_i = 1 \leftrightarrow y_i == k$$

$$\quad f_k = clf.train(X, z)$$

$$y_p(x) = argmax_k f_k(x)$$

- Train a classifier to distinguish each label from the rest
- Total K classifiers
- Prediction: pick label with the highest score

One-vs-One

- Train independent classifier for each pair of labels (i, j)
- Total: K(K-1)/2 classifiers
- Prediction: select most common class via voting

$$y_p(x) = argmax_k \sum_{i=1}^{k} f_{ik}(x)$$

Multiclass strategies

One-vs-All

- Linear in num. of classes

One-vs-One

- Quadratic in num. of classes

Multiclass strategies

One-vs-All

- Linear in num. of classes
- Unbalanced classes problem
- Biases in base models’ scores

One-vs-One

- Quadratic in num. of classes
- Works faster with non-scalable models (e.g. kernels)
- More ambiguity in predictions

Multiclass strategies

- Reduction to binary
- Model extension

Multiclass metrics

Recap: binary classification

$$y_i \in \{0, 1\}$$

$$a_i \in \{0, 1\}$$

Error type	Prediction	Ground truth
True Positive (TP)	1	1
True Negative (TN)	0	0
False Positive (FP)	1	0
False Negative (FN)	0	1

Multiclass metrics

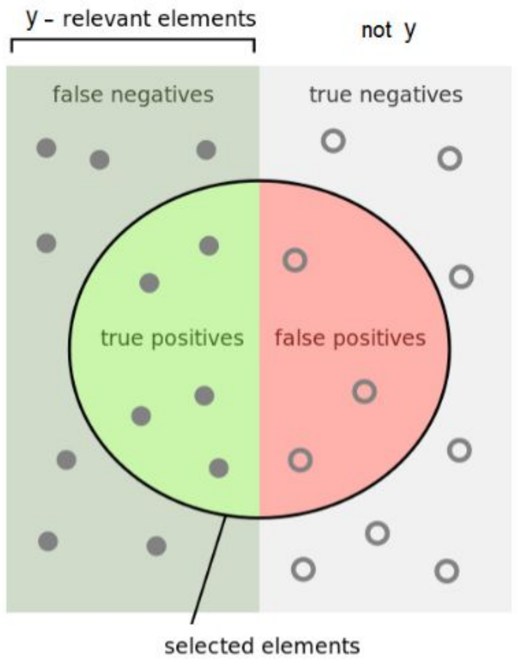
- Let’s extend binary classification metrics

for each class $y \in Y$:

TP_y - True positive predictions

FP_y - False positive predictions

FN_y - False negative predictions



Micro-averaging

$$\text{Precision: } P = \frac{\sum_y \text{TP}_y}{\sum_y(\text{TP}_y + \text{FP}_y)};$$

$$\text{Recall: } R = \frac{\sum_y \text{TP}_y}{\sum_y(\text{TP}_y + \text{FN}_y)};$$

Micro-averaging

$$\text{Precision: } P = \frac{\sum_y \mathrm{TP}_y}{\sum_y(\mathrm{TP}_y + \mathrm{FP}_y)};$$

$$\text{Recall: } R = \frac{\sum_y \mathrm{TP}_y}{\sum_y(\mathrm{TP}_y + \mathrm{FN}_y)};$$

- Does not cover imbalanced classes

Macro-averaging

$$\text{Precision: } P = \frac{1}{|Y|}\sum_{y}\frac{\mathrm{TP}_y}{\mathrm{TP}_y + \mathrm{FP}_y};$$

$$\text{Recall: } R = \frac{1}{|Y|}\sum_{y}\frac{\mathrm{TP}_y}{\mathrm{TP}_y + \mathrm{FN}_y};$$

- Just average class scores

Micro-averaging of macro-averaging?

4 classes; model always outputs 1

C1: TP = 1, FP = 0

C2: TP = 1, FP = 0

C3: TP = 53, FP = 47

C4: TP = 1, FP = 1

Micro-averaging of macro-averaging?

4 classes; model always outputs 1

C1: TP = 1, FP = 0

C2: TP = 1, FP = 0

C3: TP = 53, FP = 47

C4: TP = 1, FP = 1

Precision_micro = 0.53

Precision_macro = 0.76

Feature selection

Feature selection

1. Model-free (statistical)
2. Model-based (instrinsic)
3. Performance-based

Statistical methods

- Correlation

$$R_j = \frac{\sum_{i=1}^{\ell}(x_{ij} - \bar{x}_j)(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{\ell}(x_{ij} - \bar{x}_j)^2 \sum_{i=1}^{\ell}(y_i - \bar{y})^2}}$$

Statistical methods

- T-score (binary classification)

$$R_j = \frac{|\mu_0 - \mu_1|}{\sqrt{\frac{\sigma_0^2}{n_0} + \frac{\sigma_1^2}{n_1}}},$$

Statistical methods

- F-score (multiclass)

$$R_j = \frac{\sum_{k=1}^{K} \frac{n_j}{K-1}(\mu_j - \mu)^2}{\frac{1}{\ell-K}\sum_{k=1}^{K}(n_j - 1)\sigma_j^2},$$

Model-based

- Linear model

$$h(x, w) = x^T w = \sum x_i w_i$$

Model-based

- Linear model

$$h(x, w) = x^T w = \sum x_i w_i$$

- Weights are proportional to corresponding features' impact on prediction
- Do not forget about feature scaling!

Model-based

- Decision trees
- **Decrease in impurity**

$$Imp(j) = \sum_t I\{j_t = j\} p(t) \Delta_i(t)$$

$$\Delta_i(t) = i(t) - \frac{N(t_L)}{N} i(T_L) - \frac{N(t_R)}{N} i(T_R)$$

Model-based

- Random Forest
- **Mean decrease impurity**
- **Out-of-bag score**

$$\mathrm{OOB} = \sum_{i=1}^{\ell} L\left(y_i, \frac{1}{\sum_{n=1}^{N}[x_i \notin X_n]} \sum_{n=1}^{N} [x_i \notin X_n] b_n(x_i)\right)$$

Performance-based

- Train model on various subsets and select those which perform best
- Estimate quality on hold-out set for not overfitting

J - a set of features, $||J|| = j$

μ_J - model trained only on J parameters

$$Q(J) = Q(\mu_J, X_{test})$$

$$Q(J) \to min$$

Performance-based

Full Search

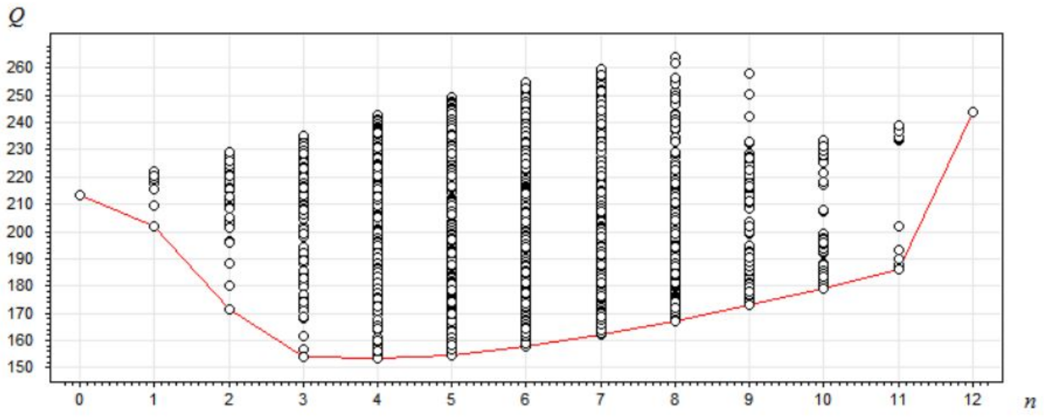
$$J = \varnothing$$

$$for\, j\, in\, 1..n:$$

$$J = argmin_{\|J\|=j} Q(\mu_J, X_{test})$$

$$if\; Q(\mu_J) < Q^* \;:\; J^* = J;\; Q^* = Q(\mu_J)$$

$$if\, \|J\| > \|J^*\| + d:\;\; return\, J^*$$



Performance-based

Full search

Pros:

- Simplicity
- Optimal solution

Cons:

- O(2^n)
- Prone to overfitting

Performance-based

Greedy addition

$$J = \varnothing$$

$$for\, j\, in\, 1..n:$$

$$f_j = argmin_{f \in F \setminus J}\; Q(\mu_{J \bigcup f}, X_{test})$$

$$J = J \bigcup f_j$$

$$if\; Q(\mu_J) < Q^*\; :\; J^* = J;\; Q^* = Q(\mu_J)$$

$$if\, j > \|J^*\| + d:\;\; return\, J^*$$

Performance-based

Greedy addition

Pros:

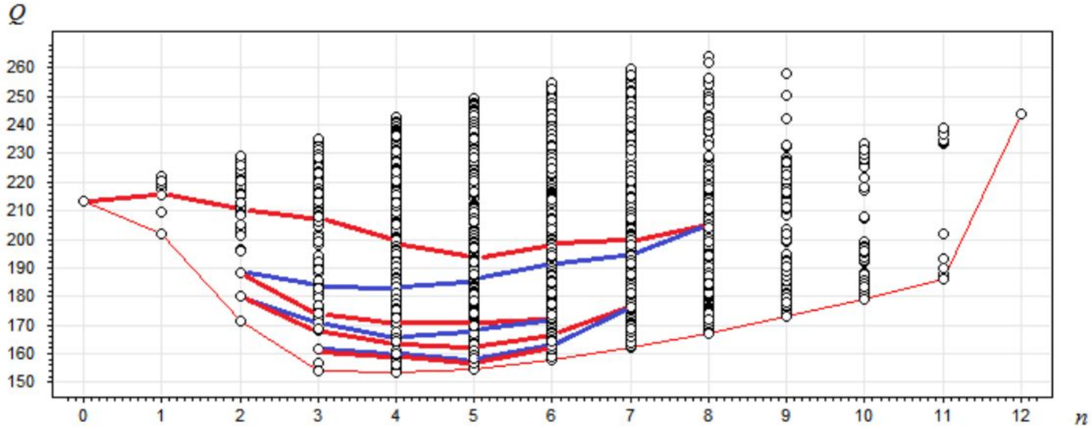
- $O(n^2)$
- Fast incremental algorithms (step-wise regression)

Cons:

- Tends to include odd features

Performance-based

Greedy addition / deletion (add-del)



Summary

- Multiclass classification
  - Classification strategies (One-vs-All, One-vs-One)
  - Metrics averaging
- Feature selection
  - Statistical
  - Model-based (intrinsic)
  - Performance-based

Summary

The following awesome materials were used:

- ML lectures by K.Vorontsov: materials
- HSE ML course by E.Sokolov: materials

Thank you for your attention!