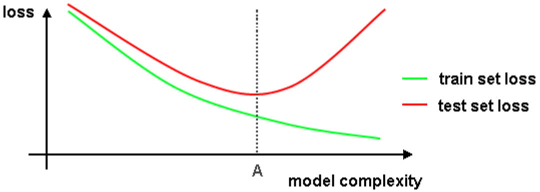
Ensemble learning, bias-variance decomposition

Victor Kitov

v.v.kitov@yandex.ru

Yandex
School of Data Analysis.

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

Table of Contents

Bias-variance decomposition

- True relationship $y = f(x) + \varepsilon$
- This relationship is estimated using random training set $(X, Y) = \{(x_n, y_n),\ n = 1, 2 \ldots N\}$
- Recovered relationship $\widehat{f}(x)$, x-some fixed constant
- Noise ε is independent of any X, Y, $\mathbb{E}\varepsilon = 0$

Bias-variance decomposition

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

- Intuition: $MSE = \text{bias}^2 + \text{variance} + \text{irreducible error}$
  - darts intuition

Proof of bias-variance decomposition

Define for brevity of notation $f = f(x)$, $\widehat{f} = \widehat{f}(x)$, $\mathbb{E} = \mathbb{E}_{X,Y,\varepsilon}$.

$$
\begin{aligned}
\mathbb{E}\left(\widehat{f} - f\right)^2 = \mathbb{E}\left(\widehat{f} - \mathbb{E}\widehat{f} + \mathbb{E}\widehat{f} - f\right)^2 &= \mathbb{E}\left(\widehat{f} - \mathbb{E}\widehat{f}\right)^2 + \left(\mathbb{E}\widehat{f} - f\right)^2 \\
&+ 2\mathbb{E}\left[(\widehat{f} - \mathbb{E}\widehat{f})(\mathbb{E}\widehat{f} - f)\right] \\
&= \mathbb{E}\left(\widehat{f} - \mathbb{E}\widehat{f}\right)^2 + \left(\mathbb{E}\widehat{f} - f\right)^2
\end{aligned}
$$

We used that $(\mathbb{E}\widehat{f} - f)$ is a constant w.r.t. X, Y and hence $\mathbb{E}\left[(\widehat{f} - \mathbb{E}\widehat{f})(\mathbb{E}\widehat{f} - f)\right] = (\mathbb{E}\widehat{f} - f)\mathbb{E}(\widehat{f} - \mathbb{E}\widehat{f}) = 0$.

$$
\begin{aligned}
\mathbb{E}\left(\widehat{f} - y\right)^2 &= \mathbb{E}\left(\widehat{f} - f - \varepsilon\right)^2 = \mathbb{E}\left(\widehat{f} - f\right)^2 + \mathbb{E}\varepsilon^2 - 2\mathbb{E}\left[(\widehat{f} - f)\varepsilon\right] \\
&= \mathbb{E}\left(\widehat{f} - \mathbb{E}\widehat{f}\right)^2 + \left(\mathbb{E}\widehat{f} - f\right)^2 + \Delta
\end{aligned}
$$

Here $\mathbb{E}\left[(\widehat{f} - f)\varepsilon\right] = \mathbb{E}\left[(\widehat{f} - f)\right]\mathbb{E}\varepsilon = 0$ since ε is independent of X, Y.

Table of Contents

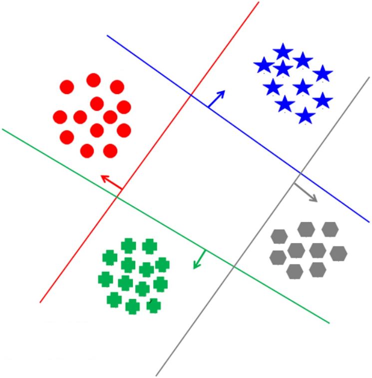
Ensemble learning

- Ensemble model - model using predictions of other models.
- Example: stacking
  - suppose we have base models $\widehat{y}_1 = f_1(x), ...\widehat{y}_M = f_M(x)$.
  - stacking: $\widehat{y}(x) = G(f_1(x), ...f_M(x))$
- Used in
  - supervised methods: regression, classification, collaborative filtering.
  - unsupervised methods: clustering, dimensionality reduction.

2. Ensemble learning
    - Ensemble learning use cases

Multiclass classification using binary classifiers

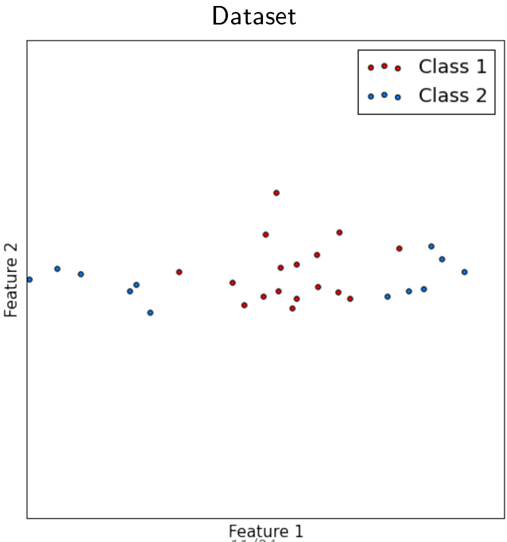
Multiclass classification with one-vs-rest, one-vs-one, error correcting codes schemes:



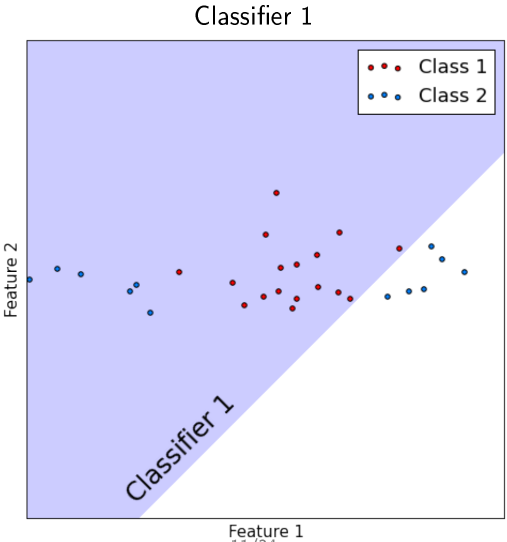
Solve underfitting

- Suppose $f_1(x), ...f_M(x)$ are too simple and underfit.
- May increase complexity by applying $G(f_1(x), ...f_M(x))$

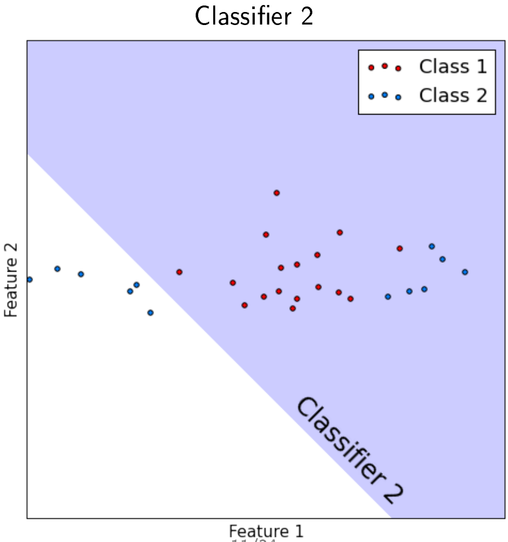
Example



Example

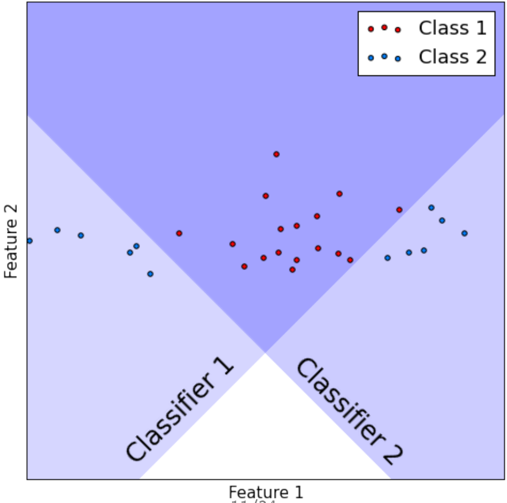

Classifier 1
Class 1
Class 2
Classifier 1
Feature 2
Feature 1

Example


Classifier 2
Class 1
Class 2
Feature 2
Feature 1
Classifier 2

Example

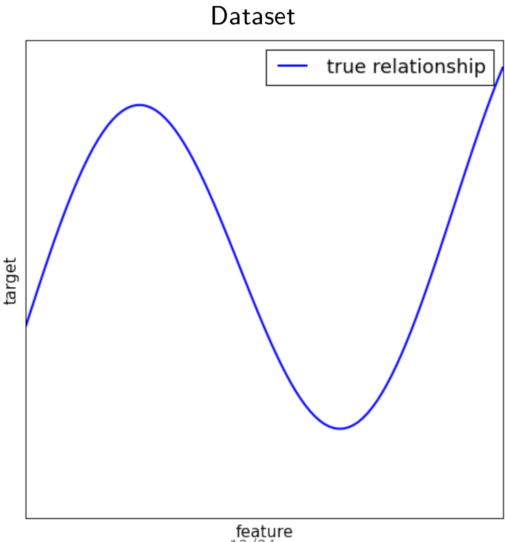
Classifier 1 and classifier 2 combined using AND rule



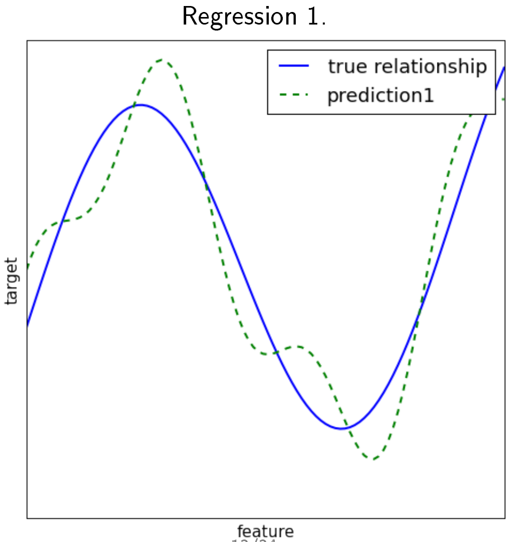
Solve overfitting

- $f_1(x), ...f_M(x)$ overfit (have high variance)
  - decision trees on different training sets
  - neural networks estimasted with different initial conditions
- Regression: average their variability to get more robust estimate:
$$\widehat{y}(x) = \frac{1}{M}\sum_{m=1}^{M} f_m(x)$$
- Classification: majority voting.

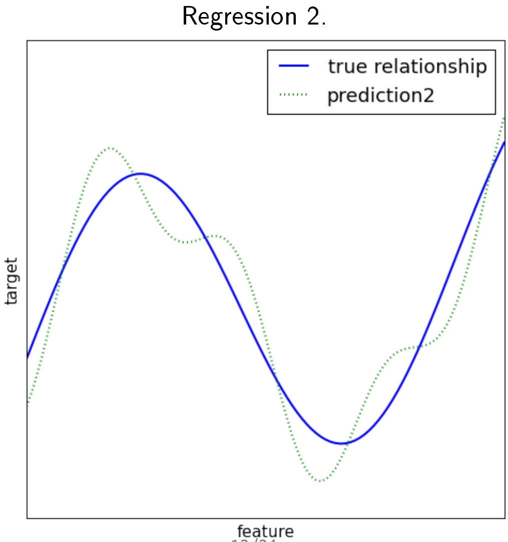
Regression: high variance



Regression: high variance

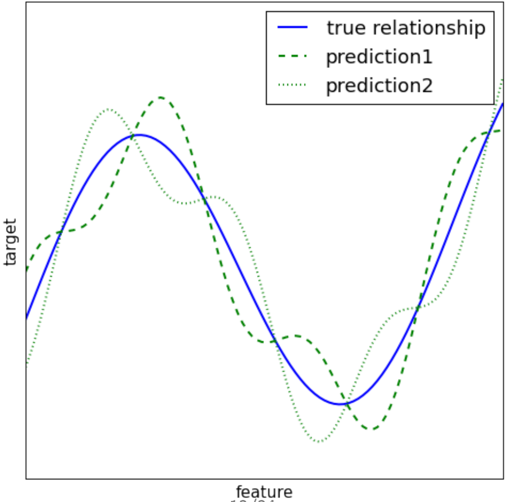


Regression: high variance



Regression: high variance

Average of regression 1 and regression 2 gives better prediction.



Majority voting of classifiers

- Consider M classifiers $f_1(x), ...f_M(x)$, performing binary classification.
- Let probability of mistake be constant $p \in (0, \frac{1}{2})$: $p(f_m(x) = y) = p \, \forall m$
- Suppose all models make mistakes or correct guesses independently of each other.
- Let $G(x)$ be majority voting combiner.
- Then $p(G(x) \neq y) \to 0$ as $m \to \infty$

Convex loss

Convex loss promotes the usage of averaged prediction instead of individual ones.

- Take convex loss $\mathcal{L}(\widehat{y}-y)$, such as absolute or square.
- Take $f_1(x),...f_M(x)$ with weights $w_1,...w_M$.
- For any fixed x consider 2 prediction strategies:

1. sample $m \sim Categorical(\alpha_1,...\alpha_M)$, $\widehat{y}(x)=f_m(x)$.
2. $\widehat{y}(x)=\sum_{m=1}^{M} w_m f_m(x)$

- Second strategy is better than first[1], averaged over different sample outcomes m.

[1] Prove that.

Ambiguity decomposition

Ambiguity decomposition:
consider predicting fixed (x, y) with ensemblefor
$F(x)=\sum_{m=1}^{M} w_m f_m(x)$, $w_m \geq 0$, $\sum_m w_m = 1$. Then

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

Ensemble is accurate when:

- $f_m(x)$ are accurate
- and/or there is huge disagreement in base learners predictions.

Proof of ambiguity decomposition

Proof:

$$\sum_m w_m (f_m(x) - F(x))^2 = \sum_m w_m (f_m(x) - y + y - F(x))^2$$

$$= \sum_m w_m (f_m(x) - y)^2 + \sum_m w_m (y - F(x))^2 + 2 \sum_m w_m (f_m(x) - y)(y - F(x))$$

$$= \sum_m w_m (f_m(x) - y)^2 + (F(x) - y)^2 + 2(y - F(x)) \sum_m w_m (f_m(x) - y)$$

$$= \sum_m w_m (f_m(x) - y)^2 + (F(x) - y)^2 + 2(y - F(x))(F(x) - y)$$

$$= \sum_m w_m (f_m(x) - y)^2 + (F(x) - y)^2 - 2(F(x) - y)^2$$

$$= \sum_m w_m (f_m(x) - y)^2 - (F(x) - y)^2$$

Data promotes ensembles

Data may promote the use of ensembles when it is divided into separate groups with different regularities.

- Flat price prediction:
  - purpose-for living: model depending on comfort, living tastes, etc.
  - purpose-for investment: another model depending on exchange rates, interest rates, stock growth, etc.
- Face detection on images:
  - one model detects face with frontal view
  - another model detects face with profile view
- Person identification using diverse information:
  - by voice, by face, by behaviour patterns, etc.

Table of Contents

Classifiers output **labels**

- **Binary classification**: output +1 <=>
  - all classifiers predict +1 (AND rule)
  - at least one classifier predicts +1 (OR rule)
  - at least k classifiers predict +1 (k-out-of-N)
- **Multiclass classification**:
  - predict most popular class (majority vote)

- Extension - **weighted account for classifiers**:
  - weighted majority vote
  - weighted k-out-of-N

Classifiers output **scores**

- Let $g_y^m(x)$ be score of class y by model m.
- Problem: scores are incomparable across models.
- Solution:
  1. define ranking score: $s_y^m(x) = \sum_{c \neq y} \mathbb{I}[g_y^m(x) > g_c^m(x)]$
  2. since $s_y^m(x)$ are comparable, assign

$$\widehat{y}(x) = \arg\max_y \sum_{m=1}^{M} s_y^m(x)$$

  Allows weighted account of classifiers.

Classifiers output **probabilities**

- Let $p_y^m(x)$ be probability of class y by classifier m.
- Possible final predictions:

$$p_y(x) = \frac{1}{M}\sum_{m=1}^{M} p_y^m(x)$$

$$p_y(x) = \text{median}_m\, p_y^m(x)$$

Allows weighted account of classifiers.

Table of Contents

Stacking algorithm

Input:

- original training set $T = \{(x_n, y_n)\}_{n=1}^{N}$
- base learners $f_1(x), ...f_M(x)$ and $G(\cdot)$.

Stacking algorithm

Input:

- original training set $T=\{(x_n,y_n)\}_{n=1}^{N}$
- base learners $f_1(x),...f_M(x)$ and $G(\cdot)$.

1. Set generated training set $T'=\{\}$
2. Split training set into K folds: $T_1,T_2,...T_K$.

Stacking algorithm

Input:

- original training set $T = \{(x_n, y_n)\}_{n=1}^{N}$
- base learners $f_1(x), ...f_M(x)$ and $G(\cdot)$.

1. Set generated training set $T' = \{\}$
2. Split training set into K folds: $T_1, T_2, ...T_K$.
3. for k in 1,2,...K:
   train $f_1(x), ...f_M(x)$ on $T \setminus T_k$
   for (x, y) in T_k:
   augment T' with sample $([f_1(x), ...f_M(x)], y)$

Stacking algorithm

Input:

- original training set $T = \{(x_n, y_n)\}_{n=1}^{N}$
- base learners $f_1(x), ...f_M(x)$ and $G(\cdot)$.

1. Set generated training set $T' = \{\}$
2. Split training set into K folds: $T_1, T_2, ...T_K$.
3. for k in 1,2,...K:
   train $f_1(x), ...f_M(x)$ on $T \setminus T_k$
   for (x, y) in T_k:
     augment T' with sample $([f_1(x), ...f_M(x)], y)$
4. Train $G(\cdot)$ on T'.

Stacking algorithm

Input:

- original training set $T = \{(x_n, y_n)\}_{n=1}^{N}$
- base learners $f_1(x), ...f_M(x)$ and $G(\cdot)$.

1. Set generated training set $T' = \{\}$
2. Split training set into K folds: $T_1, T_2, ...T_K$.
3. for k in 1,2,...K:
   train $f_1(x), ...f_M(x)$ on $T \backslash T_k$
   for (x, y) in T_k:
      augment T' with sample $([f_1(x), ...f_M(x)], y)$
4. Train $G(\cdot)$ on T'.
5. Retrain $f_1(x), ...f_M(x)$ on T.

Output: ensemble $G(f_1(x), ...f_M(x))$.

Comments

- Training $f_1(x), ...f_M(x), G(\cdot)$ on the same data causes overfitting.
- Besides $f_1(x), ...f_M(x)$ $G(\cdot)$ may also depend on
  - original features x
  - internal representations inside f_m such as class scores, probabilities.

Linear stacking (blending)

- Linear stacking:

$$f(x)=\sum_{m=1}^{M}w_m f_m(x)$$

$$\left(\sum_{m=1}^{M}w_m f_m(x_n)-y_n\right)^2\to\min_{\mathbf{w}}$$

- $f_1(x),...f_M(x)$ are correlated (predict the same y) => estimate unstable.
- For more robust estimate solve:

$$\begin{cases}\left(\sum_{m=1}^{M}w_m f_m(x_n)-y_n\right)^2+\lambda\sum_{m=1}^{M}\left(w_m-\frac{1}{M}\right)^2\to\min_{\mathbf{w}}\\ w_1\geq 0,...w_M\geq 0\end{cases}$$

Table of Contents

Bagging & random subspaces

When model overfits to particular training set T, it is useful to generate many training sets $T_1, \dots T_M$, estimate model on each of them and average.

- **Bagging**:
  - random selection of samples (with replacement)[2] [3]
- **Random subspace method**:
  - random selection of features (without replacement)
- May apply both methods jointly.

[2] *what is the probability that observation will not belong to bootstrap sample?*
[3] *what is the limit of this probability with $N \to \infty$?*

Bagged trees

In CART trees we solve

$$\widehat{f},\widehat{h}=\arg\min_{f,h\in S(t)}\Delta I(t)$$

$S(t)$ for standard decision trees:

```
S = {}
for each f in {1,...,D}
    for each h in unique {x_n^f}_{n: x_n ∈ t}
        S := S ∪ (f, h)
```

Bagged decision trees - bagging applied to standard decision trees.

Random forest & extra random trees

- Random forest & extra random trees are bagged decision trees with restricted search through (f, h), controlled by $\alpha \in (0, 1]$.
    - restricted search=>higher bias, smaller variance.

$S(t)$ for random forest:

```
S = {},  K = αD
sample d_1,...d_K randomly from {1,...,D} without replacement.
for each f in d_1,...d_K
   for each h in unique {x_n^f}_{n:x_n∈t}
      S := S ∪ (f,h)
```

$S(t)$ for extra random trees:

```
S = {},  K = αD
sample d_1,...d_K randomly from {1,...,D} without replacement.
for each f in d_1,...d_K
   sample h randomly from unique {x_n^f}_{n:x_n∈t}
   S := S ∪ (f,h)
```

Out-of-bag estimate

Out-of-bag estimate - estimate of expected loss by bagged algorithms.

- from above (pessimistic)
- without need for separate validation set

$$OOB = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}\left(\frac{1}{|I_n|}\sum_{m\in I_n} f_m(x_n), y_n\right)$$

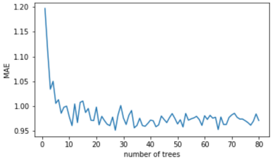
For each (x_n, y_n) we make prediction using only those models $I_n = \{m : (x_n, y_n) \notin T_m\}$ for which (x_n, y_n) is new.

Comments

- Bagged decision trees, RF, ERT:
  - have straighforward parallel implementation
  - but trees are not targeted to correct mistakes of each other
- Trees in RF, ERT may be built on the same training set T
  - due to stochastic $S(t)$ they will be different anyway

Comments

- Let M=# of base learners.
- ERT trains faster than RF, but on average requires higher M.
- Typical dependency between loss of bagging/RF/ERT depending on M:



- We average variability of tree to training set, what is more efficient for higher M.
- To find optimal hyperparameters set small M, find other parameters, then set high M back.

Conclusion

- Bias-variance decomposition gives 2 sources for poor accuracy:
  - bias: for underfitted models
  - variance: for overfitted models
- Stacking with complex aggregating model decreases bias.
  - may add variance
- Stacking with simple aggregating model (averaging, majority vote) decreases variance.