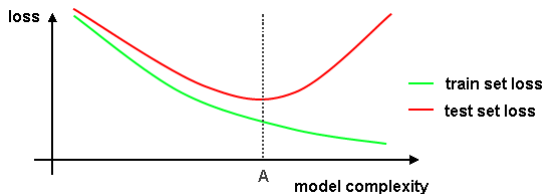


# Ensemble learning, bias-variance decomposition

Victor Kitov

[v.v.kitov@yandex.ru](mailto:v.v.kitov@yandex.ru)

## 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

- 1 Bias-variance decomposition
- 2 Ensemble learning
- 3 Fixed integration schemes for classification
- 4 Stacking
- 5 Train set sampling ensemble methods

## Bias-variance decomposition

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

### Bias-variance decomposition

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

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

# Table of Contents

- 1 Bias-variance decomposition
- 2 Ensemble learning
  - Ensemble learning use cases
- 3 Fixed integration schemes for classification
- 4 Stacking
- 5 Train set sampling ensemble methods

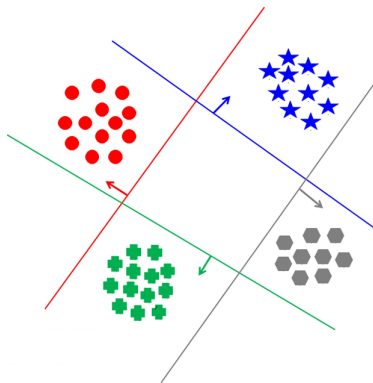
# Ensemble learning

- Ensemble model - model using predictions of other models.
- Example: stacking
  - suppose we have base models  $\hat{y}_1 = f_1(x), \dots, \hat{y}_M = f_M(x)$ .
  - stacking:  $\hat{y}(x) = G(f_1(x), \dots, f_M(x))$
- 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:



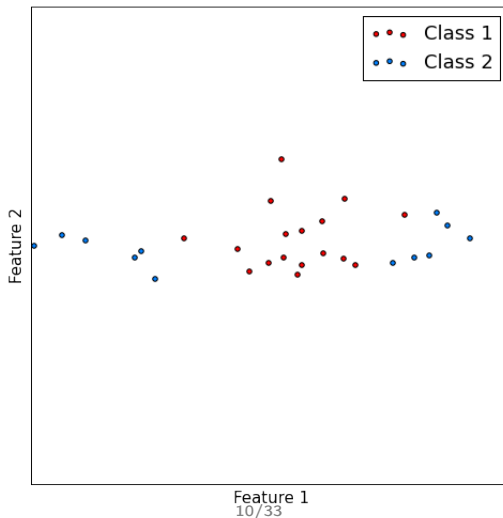


# Solve underfitting

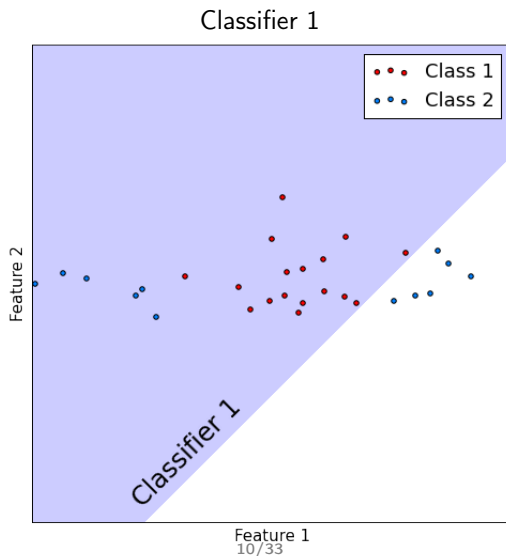
- Suppose  $f_1(x), \dots, f_M(x)$  are too simple and underfit.
- May increase complexity by applying  $G(f_1(x), \dots, f_M(x))$

# Example

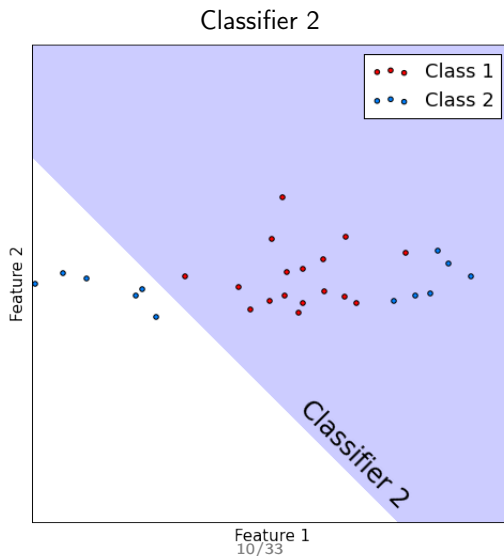
Dataset



# Example

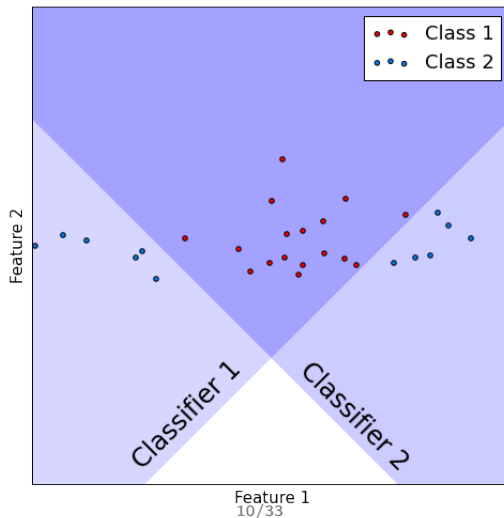


# Example



# Example

Classifier 1 and classifier 2 combined using AND rule



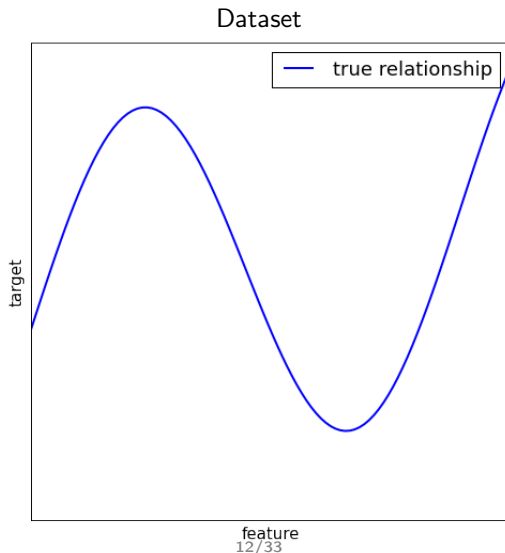
# Solve overfitting

- $f_1(x), \dots, f_M(x)$  overfit (have high variance)
  - decision trees on different training sets
  - neural networks estimated with different initial conditions
- Regression: average their variability to get more robust estimate:

$$\hat{y}(x) = \frac{1}{M} \sum_{m=1}^M f_m(x)$$

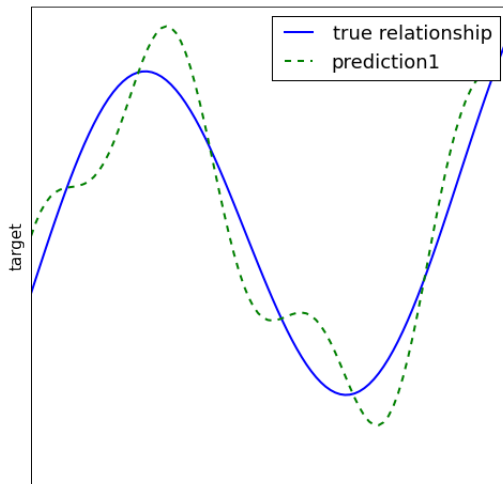
- Classification: majority voting.

## Regression: high variance



# Regression: high variance

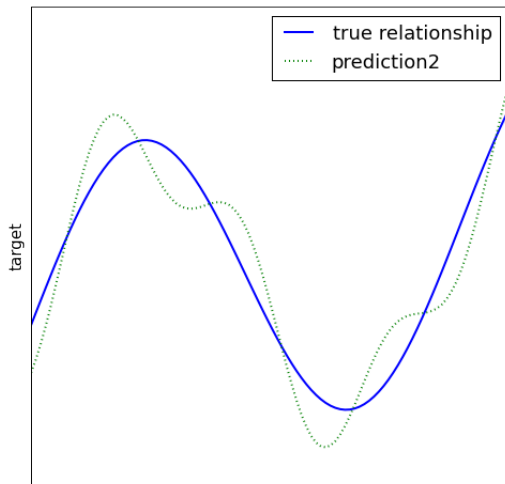
Regression 1.





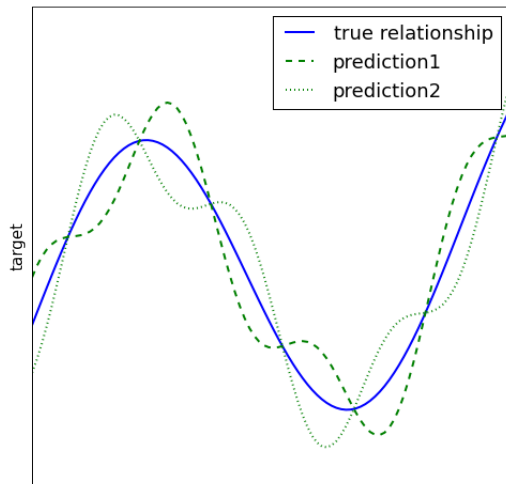
# Regression: high variance

Regression 2.



# Regression: high variance

Average of regression 1 and regression 2 gives better prediction.



# Majority voting of classifiers

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

# Convex loss

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

- Take convex loss  $\mathcal{L}(\hat{y} - y)$ , such as absolute or square.
- Take  $f_1(x), \dots, f_M(x)$  with weights  $w_1, \dots, w_M$ .
- For any fixed  $x$  consider 2 prediction strategies:
  - 1 sample  $m \sim \text{Categorical}(\alpha_1, \dots, \alpha_M)$ ,  $\hat{y}(x) = f_m(x)$ .
  - 2  $\hat{y}(x) = \sum_{m=1}^M w_m f_m(x)$
- Second strategy is better than first<sup>1</sup>, averaged over different sample outcomes  $m$ .

---

<sup>1</sup>Prove that.

# Ambiguity decomposition

## Ambiguity decomposition:

consider predicting fixed  $(x, y)$  with ensemble for

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

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

Ensemble is accurate when:

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

# Proof of ambiguity decomposition

**Proof:**

$$\begin{aligned} \sum_m w_m (f_m(x) - F(x))^2 &= \sum_m w_m (f_m(x) - y + y - F(x))^2 \\ &= \sum_m w_m (f_m(x) - y)^2 + \sum_m w_m (y - F(x))^2 + 2 \sum_m w_m (f_m(x) - y) (y - F(x)) \\ &= \sum_m w_m (f_m(x) - y)^2 + (F(x) - y)^2 + 2 (y - F(x)) \sum_m w_m (f_m(x) - y) \\ &= \sum_m w_m (f_m(x) - y)^2 + (F(x) - y)^2 + 2 (y - F(x)) (F(x) - y) \\ &= \sum_m w_m (f_m(x) - y)^2 + (F(x) - y)^2 - 2 (F(x) - y)^2 \\ &= \sum_m w_m (f_m(x) - y)^2 - (F(x) - y)^2 \end{aligned}$$

# 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

- 1 Bias-variance decomposition
- 2 Ensemble learning
- 3 Fixed integration schemes for classification**
- 4 Stacking
- 5 Train set sampling ensemble methods



# Classifiers output **labels**

- **Binary classification:** output  $+1 \Leftrightarrow$ 
  - 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

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

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

- 1 Bias-variance decomposition
- 2 Ensemble learning
- 3 Fixed integration schemes for classification
- 4 Stacking**
- 5 Train set sampling ensemble methods

## Stacking algorithm

### Input:

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

# Stacking algorithm

**Input:**

- original training set  $T = \{(x_n, y_n)\}_{n=1}^N$
  - base learners  $f_1(x), \dots, f_M(x)$  and  $G(\cdot)$ .
- 1 Set generated training set  $T' = \{\}$
  - 2 Split training set into  $K$  folds:  $T_1, T_2, \dots, T_K$ .

# Stacking algorithm

## Input:

- original training set  $T = \{(x_n, y_n)\}_{n=1}^N$
  - base learners  $f_1(x), \dots, f_M(x)$  and  $G(\cdot)$ .
- 1 Set generated training set  $T' = \{\}$
  - 2 Split training set into  $K$  folds:  $T_1, T_2, \dots, T_K$ .
  - 3 for  $k$  in  $1, 2, \dots, K$ :
    - train  $f_1(x), \dots, f_M(x)$  on  $T \setminus T_k$
    - for  $(x, y)$  in  $T_k$ :
      - augment  $T'$  with sample  $([f_1(x), \dots, f_M(x)], y)$

# Stacking algorithm

## Input:

- original training set  $T = \{(x_n, y_n)\}_{n=1}^N$
- base learners  $f_1(x), \dots, f_M(x)$  and  $G(\cdot)$ .
- ❶ Set generated training set  $T' = \{\}$
- ❷ Split training set into  $K$  folds:  $T_1, T_2, \dots, T_K$ .
- ❸ for  $k$  in  $1, 2, \dots, K$ :
  - train  $f_1(x), \dots, f_M(x)$  on  $T \setminus T_k$
  - for  $(x, y)$  in  $T_k$ :
    - augment  $T'$  with sample  $([f_1(x), \dots, f_M(x)], y)$
- ❹ Train  $G(\cdot)$  on  $T'$ .



# Stacking algorithm

## Input:

- original training set  $T = \{(x_n, y_n)\}_{n=1}^N$
  - base learners  $f_1(x), \dots, f_M(x)$  and  $G(\cdot)$ .
- 1 Set generated training set  $T' = \{\}$
  - 2 Split training set into  $K$  folds:  $T_1, T_2, \dots, T_K$ .
  - 3 for  $k$  in  $1, 2, \dots, K$ :
    - train  $f_1(x), \dots, f_M(x)$  on  $T \setminus T_k$
    - for  $(x, y)$  in  $T_k$ :
      - augment  $T'$  with sample  $([f_1(x), \dots, f_M(x)], y)$
  - 4 Train  $G(\cdot)$  on  $T'$ .
  - 5 Retrain  $f_1(x), \dots, f_M(x)$  on  $T$ .

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

# Comments

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

## Linear stacking (blending)

- Linear stacking:

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

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

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

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

# Table of Contents

- 1 Bias-variance decomposition
- 2 Ensemble learning
- 3 Fixed integration schemes for classification
- 4 Stacking
- 5 Train set sampling ensemble methods**

## 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)<sup>2,3</sup>
- **Random subspace method:**
  - random selection of features (without replacement)
- May apply both methods jointly.

---

<sup>2</sup>what is the probability that observation will not belong to bootstrap sample?

<sup>3</sup>what is the limit of this probability with  $N \rightarrow \infty$ ?

## Bagged trees

In CART trees we solve

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

$S(t)$  for standard decision trees:

```
S = {}  
for each f in {1, ..., D}  
  for each h in unique {x_n^f}_{n: x_n \in t}  
    S := S \cup (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  $\Rightarrow$  higher bias, smaller variance.

$S(t)$  for random forest:

```

 $S = \{\}, K = \alpha D$ 
sample  $d_1, \dots, d_K$  randomly from  $\{1, \dots, D\}$  without replacement.
for each  $f$  in  $d_1, \dots, d_K$ 
  for each  $h$  in  $\text{unique} \{x_n^f\}_{n: x_n \in t}$ 
     $S := S \cup (f, h)$ 
  
```

$S(t)$  for extra random trees:

```

 $S = \{\}, K = \alpha D$ 
sample  $d_1, \dots, d_K$  randomly from  $\{1, \dots, D\}$  without replacement.
for each  $f$  in  $d_1, \dots, d_K$ 
  sample  $h$  randomly from  $\text{unique} \{x_n^f\}_{n: x_n \in t}$ 
   $S := S \cup (f, h)$ 
  
```

## 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.

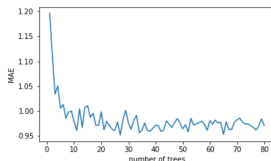


# Comments

- Bagged decision trees, RF, ERT:
  - have straightforward 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.