STK-IN4300
Statistical Learning Methods in Data Science

Riccardo De Bin

debin@math.uio.no

Outline of the lecture

- Random Forests
  - Definition of Random Forests
  - Analysis of Random Forests
  - Details of Random Forests
- Adaptive Nearest Neighbours
- Random Forests and Adaptive Nearest Neighbours

Definition of Random Forests: from bagging to random forests

Increase the performance of a tree by reducing the variance

$$\downarrow$$

bagging: $\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*}(x)$

where

- $\hat{f}^{*}(x)$ is a tree estimate based on a bootstrap sample;
- B is the number of bootstrap samples.

The average of B identically distributed r.v. with variance σ^2 and positive pairwise correlation ρ has variance

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

Definition of Random Forests: main idea

Issue:

- the pairwise correlation between bootstrap trees limits the advantages of reducing the variance by averaging;

Solution:

- at each split, only consider a random subgroup of input variables for the splitting;
- the size of the subgroup, $m \leqslant p$, is a tuning parameter;
- often a default value is used:
  - classification: $\lfloor\sqrt{p}\rfloor$;
  - regression: $\lfloor p/3\rfloor$.

Definition of Random Forests: algorithm

For $b = 1$ to B:

(a) Draw a bootstrap sample Z^* from the data;

(b) Grow a random tree T_b to the bootstrap data, by recursively repeating steps (i), (ii), (iii) for each terminal node until the minimum node size n_{min} is reached:

(i) randomly select $m \leqslant p$ variables;

(ii) pick the best variable/split point only using the m selected variables;

(iii) split the node in two daughter nodes.

The output is the ensemble of trees $\{T_b\}_{b=1}^{B}$.

Definition of Random Forests: classification vs regression

Depending on the problem, the prediction at a new point x:

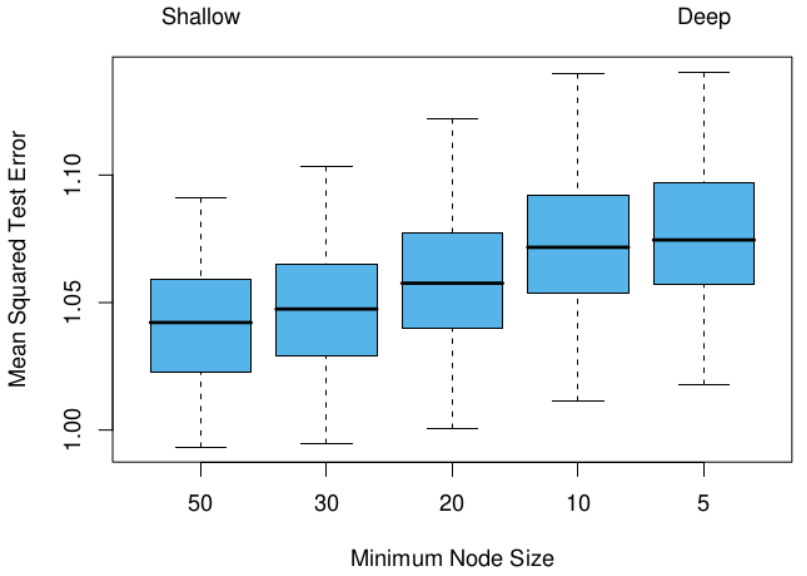
- regression: $\hat{f}_{\text{rf}}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x;\Theta_b)$,
  - where $\Theta_b = \{R_b, c_b\}$ characterizes the tree in terms of split variables, cutpoints at each node and terminal node values;
- classification: $\hat{C}_{\text{rf}}^{B} = \text{majority vote}\{\hat{C}_b(x;\Theta_b)\}_1^B$,
  - where $\hat{C}_b(x;\Theta_b)$ is the class prediction of the random-forest tree computed on the b-th bootstrap sample.

Definition of Random Forests: further tuning parameter n_{min}

The step (b) of the algorithm claims that a tree must be grow until a specific number of terminal nodes, n_{min};

- additional tuning parameter;
- Segal (2004) demonstrated some gains in the performance of the random forest when this parameter is tuned;
- in the textbook it is argued that it is not worth adding a tuning parameter, because the cost of growing the tree, up to the complete tree, is small.

Definition of Random Forests: further tuning parameter n_{min}

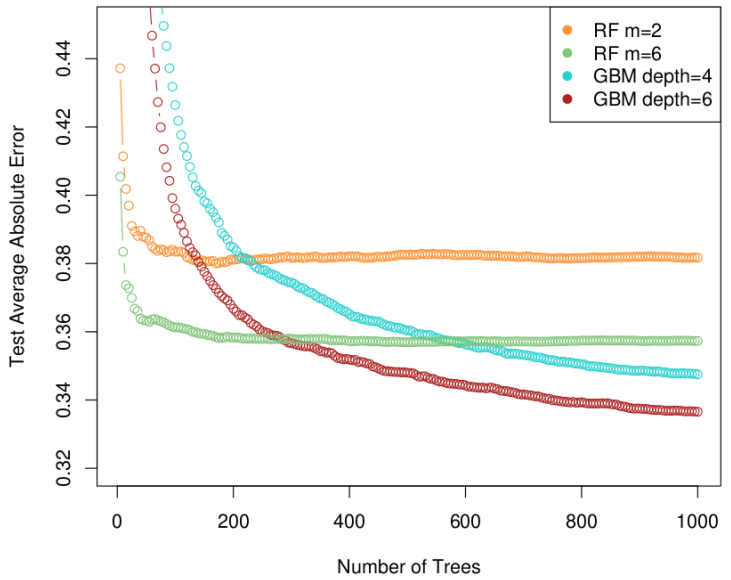


Definition of Random Forests: more on the tuning parameters

In contrast, the textbook shows that the default choice for the tuning parameter is not always the best. Consider the California Housing data:

- aggregate data of 20.460 neighbourhoods in California;
- response: median house value (in $\$100.000$);
- eight numerical predictors (input):
  - `MedInc`: median income of the people living in the neighbour;
  - `House`: house density (number of houses);
  - `AveOccup`: average occupancy of the house;
  - `longitude`: longitude of the house;
  - `latitude`: latitude of the house;
  - `AveRooms`: average number of rooms per house;
  - `AveBedrms`: average number of bedrooms per house.

Definition of Random Forests: more on the tuning parameters



Definition of Random Forests: more on the tuning parameters

Note that:

- $\lfloor 8/3 \rfloor = 2$, but the results are better with $m = 6$;
- the test error for the two random forests stabilizes at $B = 200$, no further improvements in considering more bootstrap samples;
- in contrast, the two boosting algorithms keep improving;
- in this case, boosting outperforms random forests.

Analysis of Random Forests: estimator

Consider the random forest estimator,

$$\hat{f}_{\text{rf}}(x) = \lim_{B\to\infty} \frac{1}{B} \sum_{b=1}^{B} T_b(x;\Theta_b) = E_{\Theta}[T(x;\Theta)],$$

a regression problem and a square error loss.

To make more explicit the dependence on the training sample Z, the book rewrite

$$\hat{f}_{\text{rf}}(x) = E_{\Theta|Z}[T(x;\Theta(Z))].$$

Analysis of Random Forests: correlation

Consider a single point x. Then,

$$\text{Var}[\hat{f}_{\text{rf}}(x)] = \rho(x)\sigma^2(x),$$

where:

- ρ is the sampling correlation between any pair of trees,

$$\rho(x) = corr[T(x;\Theta_1(Z)), T(x;\Theta_2(Z))]$$

  where $\Theta_1(Z)$ and $\Theta_2(Z)$ are a randomly drawn pair of random forest trees grown to the randomly sampled Z;
- $\sigma^2(x)$ is the sampling variance of any single randomly drawn tree,

$$\sigma^2(x) = Var[T(x;\Theta_2(Z))].$$

Analysis of Random Forests: correlation

Note:

- $\rho(x)$ is **NOT** the average correlation between $T_{b_1}(x;\Theta_{b_1}(Z=z))$ and $T_{b_2}(x;\Theta_{b_2}(Z=z))$, $b_1 \neq b_2 = 1,\ldots,B$ that form a random forest ensemble;
- $\rho(x)$ is the theoretical correlation between a $T_{b_1}(x;\Theta_1(Z))$ and $T_{b_2}(x;\Theta_2(Z))$ when drawing Z from the population and drawing a pair of random trees;
- $\rho(x)$ is induced by the sampling distribution of Z and Θ.

Analysis of Random Forests: correlation

Consider the following simulation model,

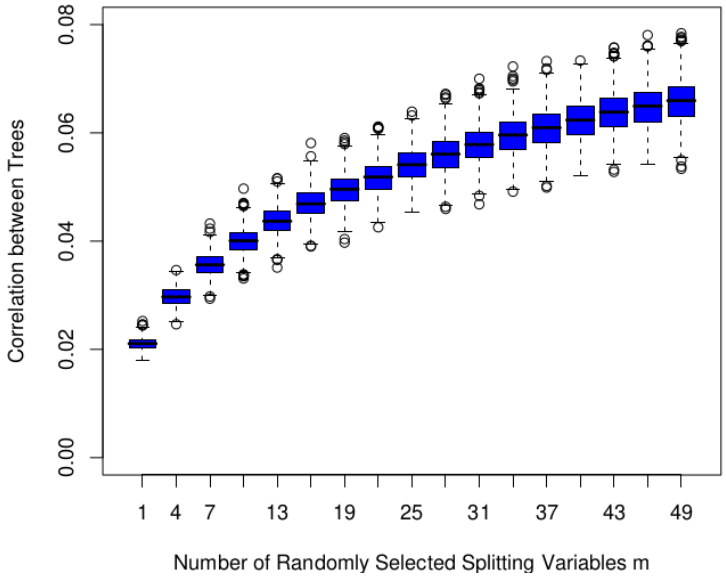
$$Y = \frac{1}{\sqrt{50}} \sum_{j=1} 50X_j + \epsilon$$

where X_j, $j = 1,\ldots,50$ and ϵ are i.i.d. Gaussian.

Generate:

- training sets: 500 training sets of 100 observations each;
- test sets: 1 set of 600 observations.

Analysis of Random Forests: correlation



Analysis of Random Forests: variance

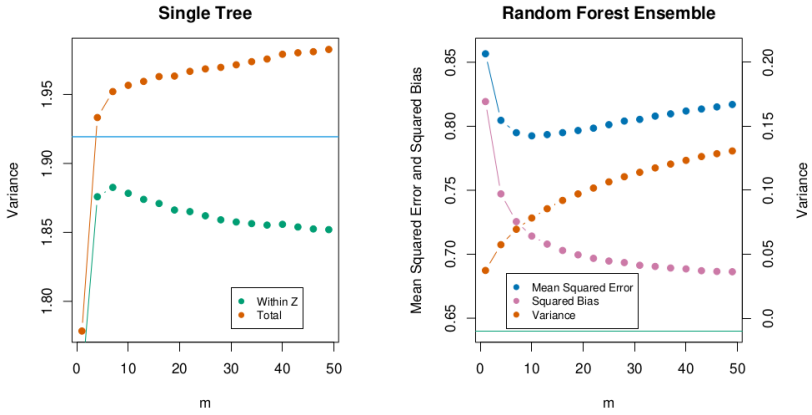
Consider now the variance of the single tree, $\text{Var}[T(x;\Theta(Z))]$. It can be decomposed as

$$\underbrace{\text{Var}_{\Theta,Z}[T(x;\Theta(Z))]}_{\text{total variance}} = \underbrace{\text{Var}_Z\left[E_{\Theta|Z}[T(x;\Theta(Z))]\right]}_{\text{Var}_Z \hat{f}_{\text{rf}}(x)} + \underbrace{E_Z\left[\text{Var}_{\Theta|Z}[T(x;\Theta(Z))]\right]}_{\text{within-Z variance}}$$

where:

- $\text{Var}_Z \hat{f}_{\text{rf}}(x)$: sampling variance of the random forest ensemble,
  - decreases with m decreasing;
- within-Z variance: variance resulting from the randomization,
  - increases with m decreasing;

Analysis of Random Forests: variance



Analysis of Random Forests: variance

As in bagging, the bias is that of any individual tree,

$$\begin{aligned}\text{Bias}(x) &= \mu(x) - E_Z[\hat{f}_{\text{rf}}(x)], \\ &= \mu(x) - E_Z\left[E_{\Theta|Z}[T(x;\Theta(Z))]\right]\end{aligned}$$

It is typically greater than the bias of an unpruned tree:

- randomization;
- reduced sample space.

General trend: larger m, smaller bias.

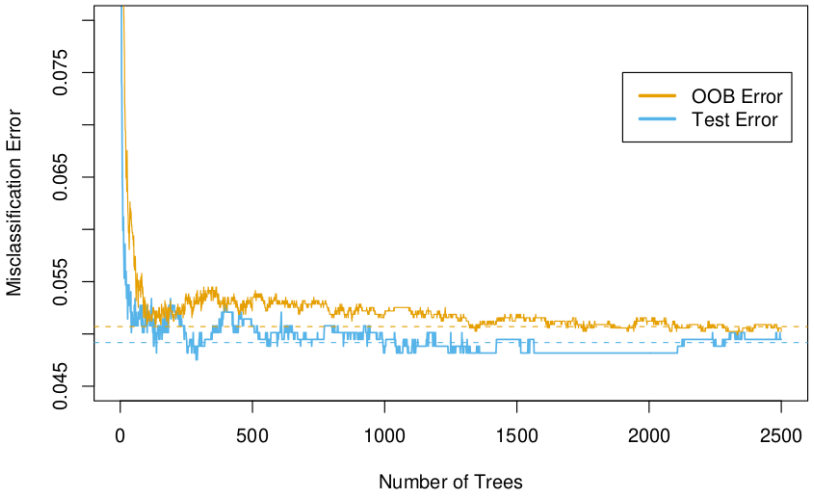
Details of Random Forests: out of bag samples

An important feature of random forests is its use of **out-of-bag** (OOB) samples:

- each tree is computed on a bootstrap sample;
- some observations $z_i = (x_i, y_i)$ are not included;
- compute the error by only averaging trees constructed on bootstrap samples not containing z_i → OOB error.

- OOB error is almost identical to N-fold cross-validation;
- random forests can be fit in one sequence.

Details of Random Forests: out of bag samples



Details of Random Forests: variable importance

A measure for the relative importance of each variable can be constructed. For each tree,

$$\mathcal{I}_\ell^2(T) = \sum_{t=1}^{J-1} \hat{\imath}_\ell^2 \mathbb{1}(v(t) = \ell)$$

where:

- computed for all variables X_ℓ, $\ell = 1, \ldots, p$;
- for each internal node t, $t = 1, \ldots, J-1$;
- $v(t)$ is the variable selected for the partition in two regions;
- $\hat{\imath}_\ell^2$ is the estimated improvement due to the split (from a common value for whole region to two values for two regions).

Details of Random Forests: variable importance

Extension to random forest (and boosting):

$$\mathcal{I}_\ell^2 = \frac{1}{B} \sum_{b=1}^{B} \mathcal{I}_\ell^2(T_b).$$

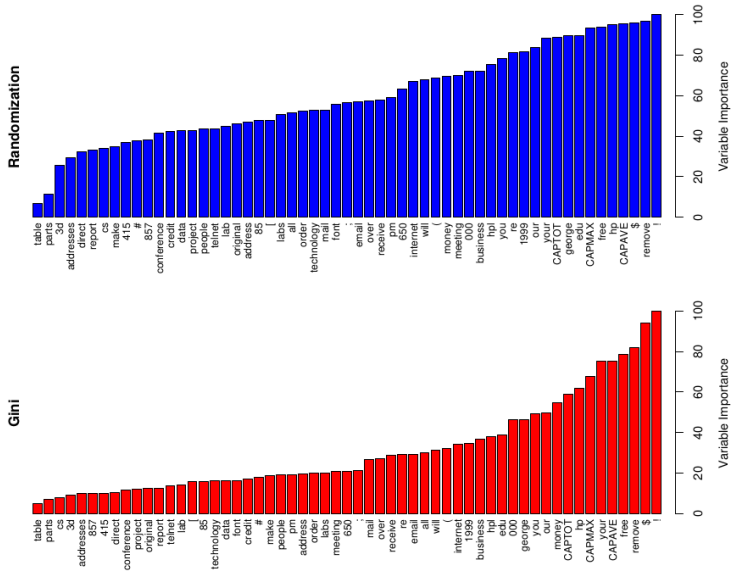
- much more reliable, due to the stabilizing effect of averaging;
- the measure is relative:
  - the value for the variable with highest importance is set to 100;
  - the other values are rescaled.

Details of Random Forests: variable importance

Alternatively, one can use the OOB samples:

- measure the prediction strength of each variable;
- at each split, for each tree
  - the prediction error is computed on the OOB sample;
  - same procedure on randomly permuted values of the OOB sample for the variable used for the split;
  - the decrease of accuracy is registered;
- the decrease of accuracy is averaged over all splits and trees;
- also a relative measure (highest 100, other rescaled).

Details of Random Forests: variable importance



Details of Random Forests: variable importance

Remarks:

- the two rankings are similar (top: OOB; bottom: $\mathcal{I}^2$);
- importances in the top plot are more uniform over the variables:
  - it does not measure the effect on prediction were this variable not available, because if the model was refitted without the variable, other variables could be used as surrogates
- all variables have some importance;
- smaller the m, higher the chances that all variables are selected at some point.

Details of Random Forests: m and sparsity

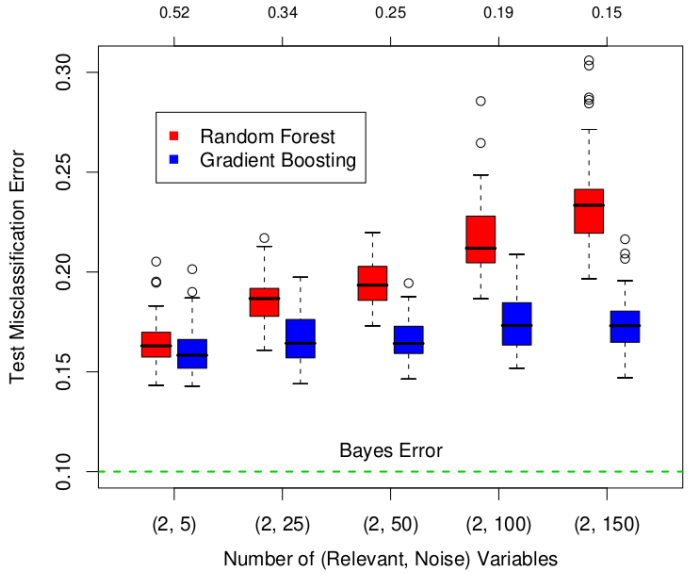
For the same reason, if the number of relevant predictors is small,

- the performance of random forests worsens when the number of noisy predictors increase;
- high chance to not include any relevant predictors in the random selection.

When the number of relevant predictors is sufficiently large,

- random forests are surprisingly robust to increases in the number of noisy predictors.

Details of Random Forests: m and sparsity



Final comments on Random Forests: remarks

Advantages of random forests:

- usually have good prediction ability;
- very little tuning is required;
- interpretability in terms of variable importance.

Nevertheless:

- difficulties in portability;
- no model behind;
- worse performance (in the book's examples) than boosting.

Final comments on Random Forests: remarks

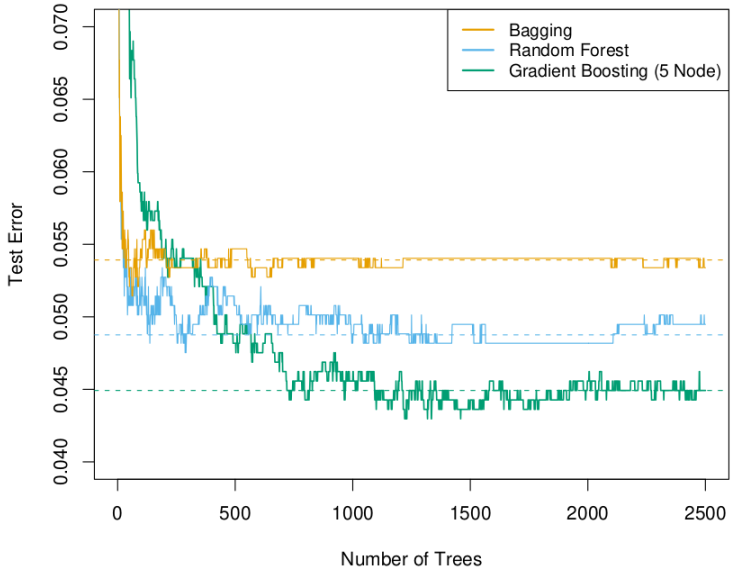
Final example 1:

- spam data;
- test error for bagging, random forests and boosting with increasing number of trees.

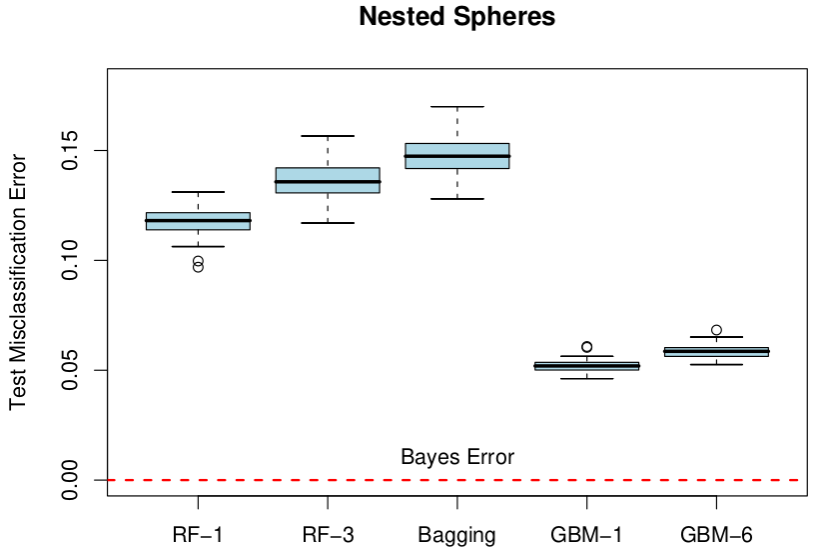
Final example 2:

- $X_1, \ldots, X_{10}$ standard independent Gaussian;
- $Y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > \chi^2_{10}(0.5), \\ -1 & \text{otherwise} \end{cases}$
- 2.000 training observations (1.000 cases for each class), 10.000 test observations.

Final comments on Random Forests: final example 2



Final comments on Random Forests: final example 1



Adaptive Nearest Neighbours: k-nearest neighbours

In the first lecture, we introduced the k-nearest neighbours (kNN),

$$\hat{f}(x) = \frac{1}{k} \sum_{i:x_i \in N_k(x)} y_i,$$

where the idea is to average over the k closest points to x:

- with X continous, Euclidean distance

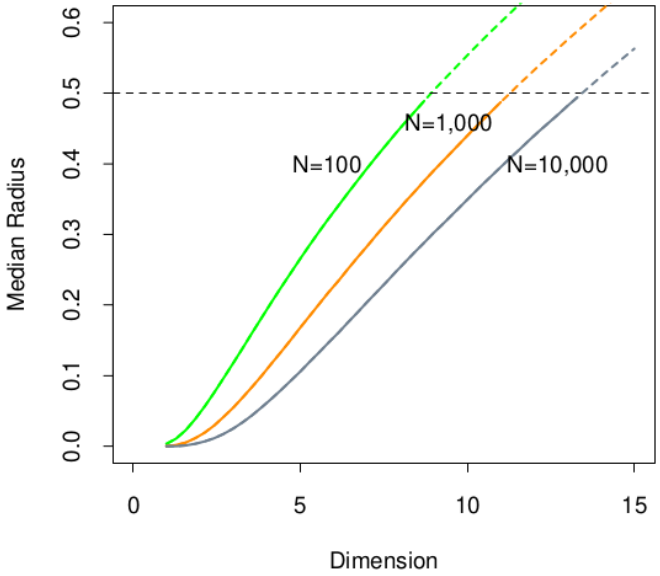
$$d_{(i)} = ||x_{(i)} - x_0||.$$

We also saw that kNN suffers from the curse of dimensionality:

$$d(p, N) = \left(1 - (1/2)^{1/N}\right)^{1/p},$$

where $d(p, N)$ is the median distance between the target point and the first neighbour.

Adaptive Nearest Neighbours: curse of dimensionality



Adaptive Nearest Neighbours: discriminant adaptive nearest neighbours

Consider a two-class classification problem, in which there are:

- two independent continuous variables;
- only one of them is relevant for the classification.

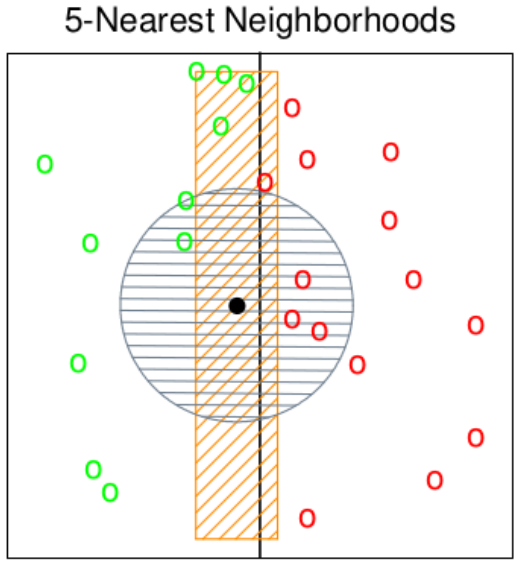
The standard kNN assumes,

- the class probabilities roughly constant in the neighbourhood;
- the neighbourhood "circular".

If we take into account that only one variable is relevant:

- we would stretch the neighbourhood accordingly;
- same variance, reduced bias.
- often the case in high-dimensional data.

Adaptive Nearest Neighbours: discriminant adaptive nearest neighbours



Adaptive Nearest Neighbours: discriminant adaptive nearest neighbours

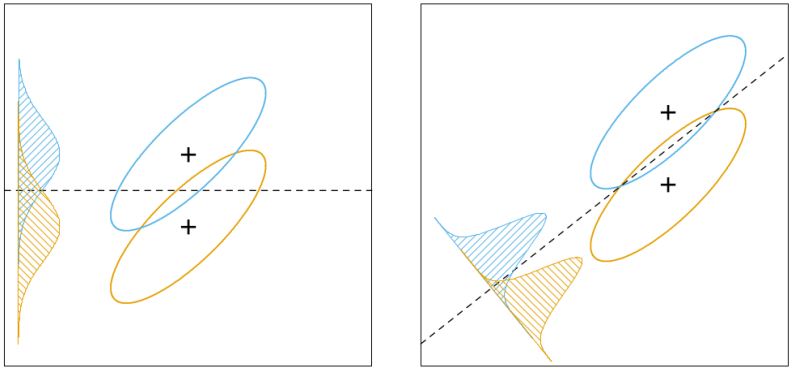
Idea:

- check the class distribution in the neighbourhood;
- decide how to deform the neighbourhood,
  - i.e., how to adapt the metric;
- at each target point the (specific) adapted metric is used.

From the previous figure:

- the neighbourhood should be stretched in the direction orthogonal to that in which the class centroids are lying;
- the direction in which the class probabilities change the least.

Adaptive Nearest Neighbours: discriminant adaptive nearest neighbours



Adaptive Nearest Neighbours: discriminant adaptive nearest neighbours

The discriminant adaptive nearest-neighbours defines the metric

$$D(x, x_0) = (x - x_0)^T \Sigma (x - x_0),$$

for a target point x_0, with

$$\begin{aligned}\Sigma &= W^{-1/2}\left[W^{-1/2} B W^{-1/2} + \epsilon I\right] W^{-1/2} \\ &= W^{-1/2}\left[B^* + \epsilon I\right] W^{-1/2}\end{aligned}$$

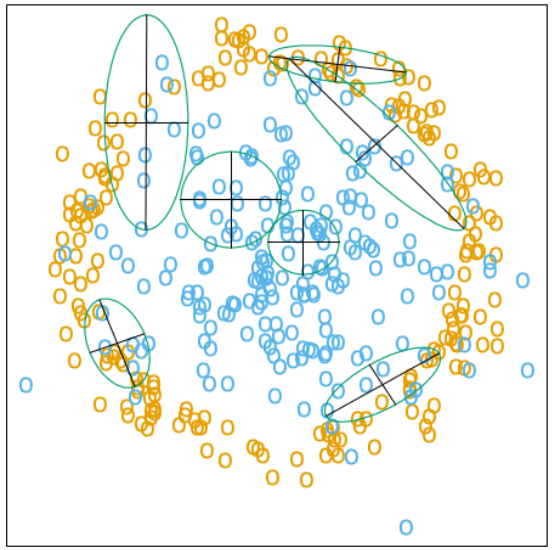
- W is the pooled within-class covariance matrix $\sum_{k=1}^{K} \pi_k W_k$;
- B is the between class cov. matrix $\sum_{k=1}^{K} \pi_k (\bar{x}_k - \bar{x})(\bar{x}_k - \bar{x})^T$;
- both W and K computed for $x_i \in N(x_0)$.

Adaptive Nearest Neighbours: discriminant adaptive nearest neighbours

Basically the algorithm:

- first spheres the data with respect to W;
- then stretches the neighbourhood in the zero-eigenvalue directions of B^*,
  - i.e., directions in which the observed class means do not differ;
- from an infinite strip to an ellipsoid due to ϵ,
  - to avoid using points far away;
  - empirically, $\epsilon = 1$ works generally well;
- if only one class, the neighbourhoods remain circular,
  - $B = 0 \rightarrow \Sigma = I$;
  - remember the X have to be scaled.

Adaptive Nearest Neighbours: discriminant adaptive nearest neighbours

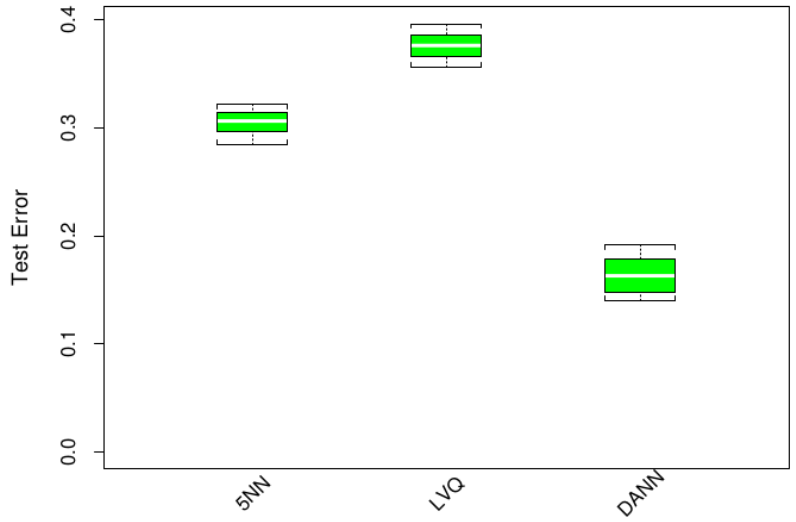


Adaptive Nearest Neighbours: example

- Generate two-class data in ten dimensions;
- all 10 predictors in class 1 are independent standard normal, conditioned on the squared radius being > 22.4 and < 40;
- the predictors in class 2 are independent standard normal without the restriction;
- the first class almost completely surrounds the second class in the full ten-dimensional space;
- 250 observations in each class;
- no pure noise variables.

- Compute the test error in 10 repetitions.

Adaptive Nearest Neighbours: example

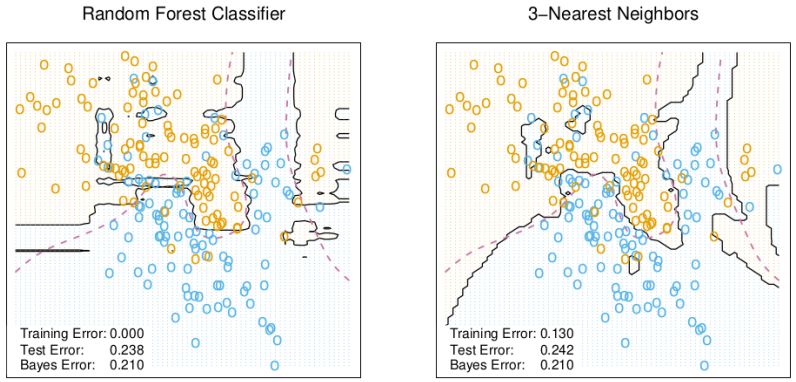


Random Forests and Adaptive Nearest Neighbours: comparison

The random forest classifier can be seen as a weighted version of the k-nearest neighbour classifier:

- for a particular Θ^*, $T(x;\Theta^*(Z))$ is the response value for one of the training samples,
  - if the tree is grown to maximal size;
  - i.e., one observation per leaf.
- the most informative predictors (among those available) are chosen along the path;
- the averaging assigns weights to these training responses;
- those observations close to the target point get assigned weights which combine to form the classification decision.

Random Forests and Adaptive Nearest Neighbours: comparison



References

Segal, M. R. (2004). Machine learning benchmarks and random forest regression. Tech. Rep. 35x3v9t4, UC San Francisco.