# Assignment 2

Ornella MEFFOVOUNG NGNITEDEM

2025-12-13

## Exercice 1: Practica SML on DNA Microarrays

### 1. Shape, size, dimmensionality

The dataset contains n = 79 observations and p = 500 input variables. since p is much larger then n, this is a typical small n , large p microarray dataset, with a high-dimensional input space and a high risk of overfitting. This as a high-dimensional dataset, which can be a challenge for many machine learning algorithms, especially those like 1kNN than can be sensitive to the curve of dimensionality.

### 2. Type of Data

Categorical Variable: The variable $Y$ is the response variable, indicating whether a subject has cancer or not. This variable is used to predict the class of a subject based on the values of the other variables.

Numeric Variable: The 500 numeric variables represent the gene expression levels, which are continuous measurements of gene expression in tissue samples. These are real-valued variables that capture the intensity of gene expression.

### 3.PLot of distribution of teh response for this datasets and comment

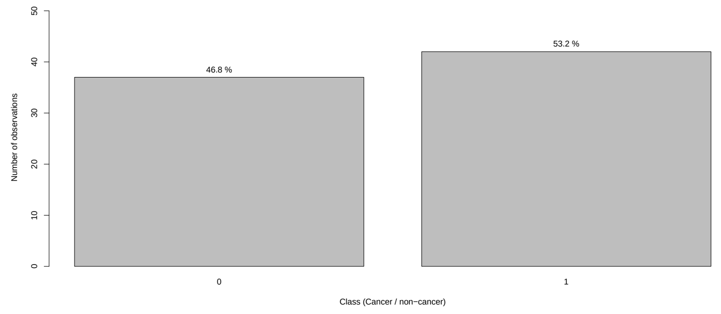

Figure 1: : Distribution of the response Y

The barplot shows that the two classes 0 = non-cancer, 1 = cancer have almost the same number of observations, so the response distributions is roughly balanced. this means there is no strong class imbalance problem, and using the overall misclassification rate as a performance measure is appropriate for this dataset.

### 4.Identify the 9 individually most powerful predictor variables

According to the Kruskal–Wallis test statistics, the 9 individually most powerful predictor variables with respect to the response are: X217844_at, X211935_at, X212640_at, X201290_at, X215333_x_at, X201480_s_at, X209454_s_at, X200047_s_at and X214001_x_at.

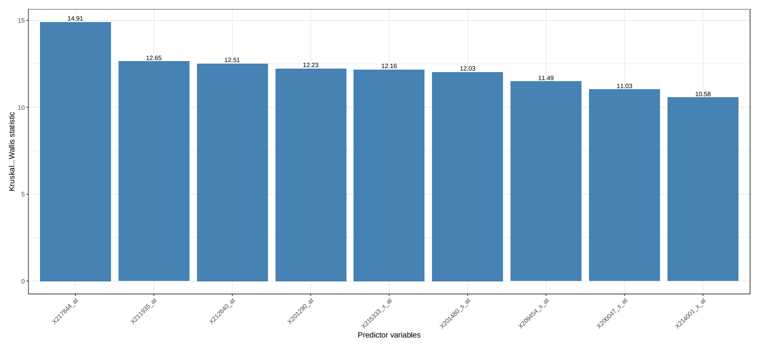

Figure 2: : Top 9 Kruskal–Wallis statistics

## 5. Generate a type="h" plot with the Kruskal-Wallis

The type = h plot for the top 9 variables shows that these genne have the largest Kruskal-Wallis statistics, so they are the most strongly associated with the response. The tallest bar e.g X217844_at correspond to the gene with ths strongest individual discrimination between cancer and non-cancer, and the other eight, although slightly smaller, still show high discriminative power.

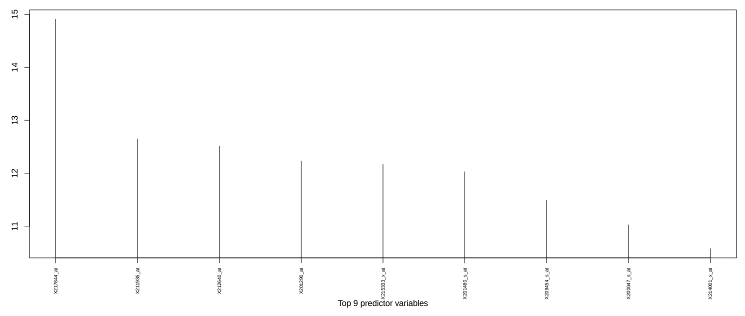

Figure 3: : Top 9 Kruskal-Wallis type h

## 6. Boxplots of 9 most with respect tothe response and comment

For the 9 most powerful variables, the boxplots reveal clear differences in expression between the two classes for several genes. In particular, some genes tend to have systematically higher (or lower) median expression in one class and the interquartile ranges overlap only partially, confirming that these genes are strongly associated with the response and have good individual discriminative power.

## 7. Build the classification tree with cp=0.01

- ### Plot the tree you just built

The classification tree built with cp = 0.01 is relatively small and shallow, indicating a moderate level of pruning that helps to avoid overfitting in this high-dimensional microarray setting.

- ### Determine the number of terminal nodes

The tree has 4 terminal nodes.

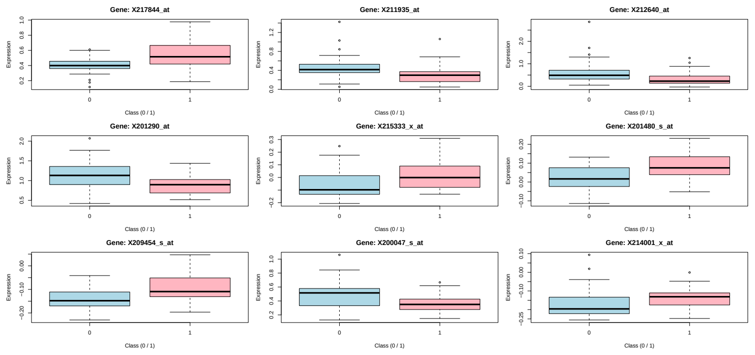

Figure 4: : Boxpots of top 9

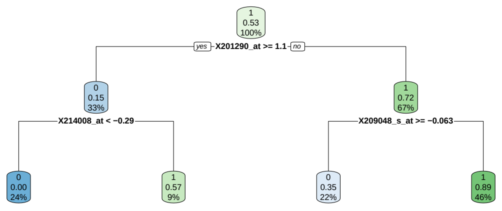

Figure 5: : Classification tree cp=0.01

- **Write down in mathematical form region 2 and Region 4.**

Region 2 corresponds to node 4 (child of node 2).

$$R_2 = \{x : X_{201290\_at} \geq 1.1 \text{ and } X_{214008\_at} \geq -0.29\}\,.$$

Region 4 corresponds to node 7 (right child of node 3).

$$R_4 = \{x : X_{201290\_at} < 1.1 \text{ and } X_{209048\_s\_at} < -0.063\}\,.$$

- **Comment on the variable at the root of the tree in light of the Kruskal-Wallis statistic**

THe root variable of the tree is X201290_at, which is one of the 9 genes with the highest Kruskal_Wallis statistics. This means it is very discriminative on its own, so it is natural that the tree uses it as the first split to separate cancer from non-cancer.

## 8. Generate the comparative boxplots of the 9 weakest variable with respect to the response and comment on what you observe.

For the 9 variables with the smallest Kruskal-Wallis statistics, teh comparative boxplots for classes 0 and 1 look very similar. The median and interquartile ranges of both classes largely overlap, with only small differences conpared to the withinclass variability, indicating that these genes have very week individual discriminative power and carry little useful information to separate cancer from non-cancer.

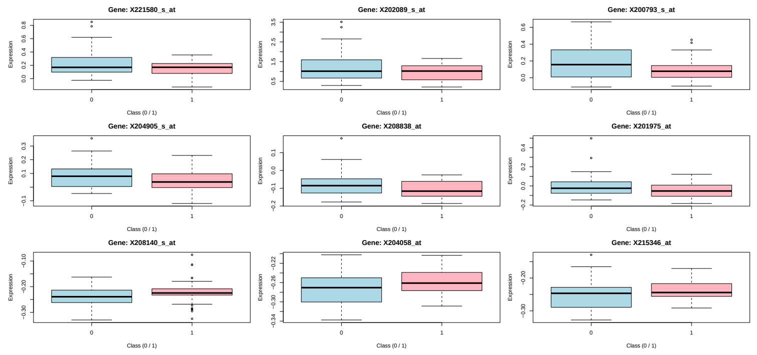

Figure 6: : Comparatif Boxpots of the 9 weakest variable

## 9. Generate the correlation plot of the predictor variables and comment extensively one what they reveal, if anything.

the 9 strongest genes are highly correlated, so they contain a lot of redundant information. several pairs have strong positive correlations around 0.5, 0.7 and somme moderate negative one around -0.4, -0.5, indicating multicolilinearity and suggesting that fewer genes could already capture most of th predictive signal.

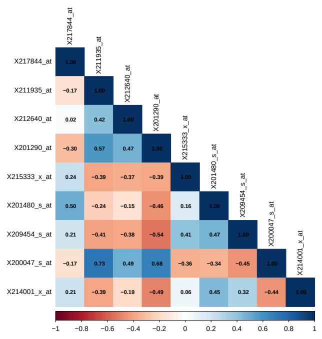

Figure 7: : the correlation plot of the predictor variables

## 10. Compute the eigendecomposition of the correlation matrix and comment on the ratio $\lambda_{max}/\lambda_{min}$.

The eigenvalues are $\lambda_{max} \approx 4.01$ and $\lambda_{min} \approx 0.24$, so the ratio is about $\lambda_{max}/\lambda_{min} \approx 17$, This relatively large ratio indicates noticeable multicollinearity among the top 9 genes (some directions in the predictor space have much larger variance than others), but not an extreme degeneracy of the correlation matrix.

## 11. Using the whole data for training and the whole data for test, building the above six learning machines, then plot the comparative ROC curves on the same grid

When training and testing on the whole dataset,all six ROC curves lie well above the diagonal, with high true positive rates for small false positive rates. THe pruned trees (cp=0.05 and cp=0.1) and the 7NN/9NN classifiers perform slightly better than 1NN, bur in

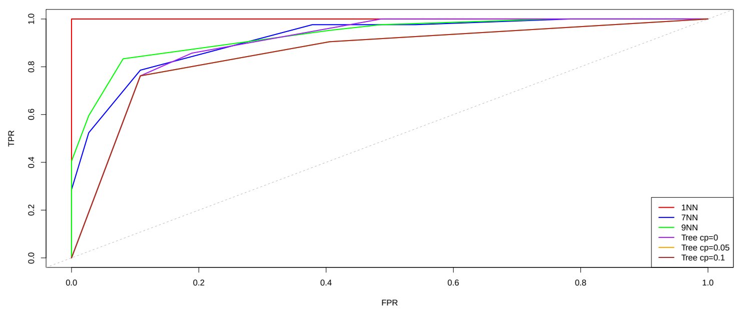

Figure 8: : Courbes ROC

## 12. Plot all the three classification tree grown, using the prp function for the package rpart.plot

### Interpretation of the three trees

All three classification trees lead to almost the same decision rule; they only differ slightly in depth and complexity.

- In all three trees, the root split is on the variable `X201290_at` with threshold `1.1`.
  - When $X201290\_at \geq 1.1$, the model mostly predicts class 0 (higher probability of class 0).
  - When `X201290_ at< 1.1`, the model mostly predicts class 1 (higher probability of class 1).
- The complexity parameter `cp` controls how strongly the tree is pruned.
  - With `cp = 0`, the tree keeps an additional split on `X214008_at` in the left branch, so this version is slightly more detailed.
  - With `cp = 0.05` and `cp = 0.1`, this extra split is removed and only a split on `X209048_at` remains, which produces simpler trees that are almost identical.
- Overall, `X201290_at` is the key variable driving the classification, while `X209048_at` (and `X214008_at` in the unpruned tree) only provide small refinements to the predictions.

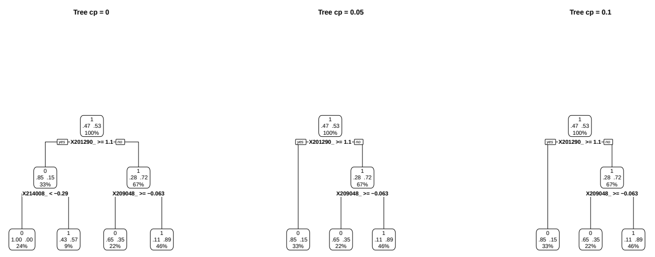

Figure 9: : Treen cp

## 13. Comment succinctly on what the ROC curves reveal for this data and argue in light of the theory whether or not that was to be expected.

The ROC curves lie clearly above the diagonal, so all six classifiers have good discriminative ability on this dataset. Moreover, the curves are almost overlapping, which means that changing kk for kNN or the complexity parameter cpcp for the trees barely affects performance, as theory predicts when several reasonable classifiers approximate the same underlying decision boundary.

## 14. Using set.seed(19671210) along with a 7/10 training 3/10 test basic stochastic holdout split of the data, compute S = 100 replicated random splits of the test error for all the above learning machines.

- **Plot the comparative boxplots (be sure to properly label the plots)**

### Analysis of Classification Errors

The graph shows the classification errors for different models:

- The models **1NN, 7NN, and 9NN** have similar errors, around 0.3.
- The decision trees with **regularization** (**`cp=0.05` and `cp=0.1`**) show lower errors, with better **stability**.
- `cp=0` shows a higher error, indicating overfitting.

### Conclusion

**Regularized decision trees** perform better than kNN, especially with a higher **`cp`** value.

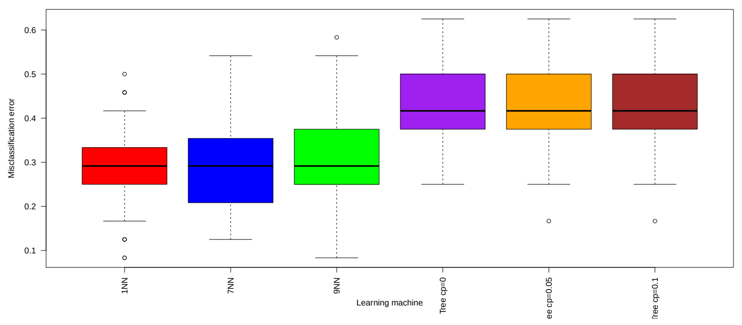

Figure 10: :Test error over 100 random 70/30 splits

- **Comment on the distribution of the test error in light of (implicit) model complexity.**

#### Erreur de test et complexité des modèles

The boxplots show that, overall, the test error remains fairly high: it is around 30–35% for the kNN models and rather 40–45% for the trees. The trees, even the most complex ones ($cp = 0$), therefore appear slightly less accurate than the kNN models on this dataset.

From the complexity point of view, 1NN is the most flexible model: it adapts very closely to the data, which sometimes allows it to reach a very low error, but it is also the most unstable from one split to another. In contrast, 7NN, 9NN and the more heavily pruned trees ($cp = 0.05$ and 0.1) are simpler: their test errors vary less, but their average performance is not really better, which nicely illustrates the classical trade-off between model complexity and stability.

- **Perform a basic analysis of variance (ANOVA) on those test errors and comment!**

### ANOVA on test errors

The one-way ANOVA gives $F \approx 63.4$ with a p-value $< 2 \times 10^{-16}$ for the method factor. This shows that the average test errors of the six learning machines are clearly not all equal: some models have significantly lower mean error than others.

In practical terms, this confirms what the boxplots already suggest: kNN and tree models do not perform at exactly the same level, and these differences are far too large to be explained by random variation across the 100 splits. Pairwise comparisons or a close look at the boxplots are then needed to pinpoint which specific methods are best.

## 15. Comment extensively on the most gemeral observation and lesson you gleaned from this exploration.

### General lessons from the exploration

Overall, several quite different models kNN and trees reach very similar ROC curves and test errors, so there is no single magic algorithm that clearly dominates on this dataset. The more complex methods 1NN, weakly pruned trees are a bit more variable across random splits, while smoother models 7NN, 9NN, more pruned trees are more stable but not dramatically better in average error, which nicely illustrates the bias–variance trade-off.

Repeating the 70/30 split 100 times and running an ANOVA on the test errors shows that some performance differences between methods are real and not just due to random splits, but they remain modest. The main practical lesson is that using a simple, reasonably tuned model and validating it carefully across many splits is more important than trying to squeeze out tiny improvements with highly complex models.

### Exercise 2 statement

We consider a binary response $Y \in \{-1, +1\}$ and a k-nearest neighbors learning machine with exponential kernel weights

$$K(x, x_i) = w_i(x) = \frac{\exp\big(-\gamma d(x, x_i)\big)}{\sum_\ell \exp\big(-\gamma d(x, x_\ell)\big)},$$

where $\gamma > 0$ and $\alpha_i = \mathbf{1}(x_i \in V_k(x))$ indicates membership in the $k$-nearest neighborhood $V_k(x) \subset D \subset \mathcal{X}$.

The classifier is

$$\hat{f}_{k\mathrm{NN}}(x) = \mathrm{sign}\Big(\sum_{i=1}^{n} Y_i\, \alpha_i K(x, x_i)\Big),$$

and we define

$$\pi(x) = \sum_{i=1}^{n} \mathbf{1}(Y_i = 1)\, \alpha_i K(x, x_i).$$

### 1. Why $\hat{f}_{k\mathbf{NN}}(x)$ predicts the label of $x$

For each $x$, only the $k$ nearest neighbours have $\alpha_i = 1$, all others have $\alpha_i = 0$. The kernel $K(x, x_i)$ is a positive weight that is larger for closer neighbours and normalized so that the weights of the neighbours sum to 1. so the term $\sum_{i=1}^{n} Y_i\, \alpha_i K(x, x_i)$ is a weighted average of the labels of the $k$ nearest neighbors, and that its sign is positive when the weighted majority label is $+1$ and negative when the majority is $-1$. Hence $\hat{f}_{k\mathrm{NN}}(x)$ returns the predicted label of $x$.

### 2. What $\pi(x)$ is estimating

$$\pi(x) = \sum_{i=1}^{n} \mathbf{1}(Y_i = 1)\, \alpha_i K(x, x_i)$$

corresponds to the normalized weighted fraction of neighbors with label $y_i = +1$, the neighbour weights $K(x, x_i)$ are normalized to sum to 1 over the neighbourhood, and therefore estimates the conditional probability $\mathbb{P}(Y = +1 \mid X = x)$ based on the local kNN kernel weighting around $x$.

## 3. Relationship between $\hat{f}_{k\mathbf{NN}}(x)$ and $\hat{g}_{k\mathbf{NN}}(x)$

when we split the weighted sum over the two classes :

$$\sum_{i=1}^{n} y_i \alpha_i K(x, x_i) = \sum_{i:y_i=1} \alpha_i K(x, x_i) - \sum_{i:y_i=-1} \alpha_i K(x, x_i).$$

we have the first term is $\pi_d(x)$, the second term equals $1 - \pi_d(x)$ because the neighbour weights sum to 1. so

$$\sum_{i=1}^{n} y_i \alpha_i K(x, x_i) = \pi_d(x) - (1 - \pi_d(x)) = 2\pi_d(x) - 1.$$

Hence

$$\hat{f}^{k\mathrm{NN}}(x) = \mathrm{sign}\big(2\pi_d(x) - 1\big).$$

Now $2\pi_d(x) - 1 > 0$ if and if $\pi_d(x) > \frac{1}{2}$ implies

$$\hat{f}^{k\mathrm{NN}}(x) = +1 \quad \text{exactly when} \quad \pi_d(x) > \tfrac{1}{2},$$

and $-1$ otherwise, which is equivalent to

$$\tilde{g}^{k\mathrm{NN}}(x) = 2\,\mathbf{1}\big(\pi_d(x) > \tfrac{1}{2}\big) - 1.$$

conclusion $\hat{f}^{k\mathrm{NN}}(x)$ and $\tilde{g}^{k\mathrm{NN}}(x)$ implement the same decision rule written in two different ways.

## 4. Output space $\mathcal{Y}$

The hypothesis space

$$\mathcal{H} = \Big\{ f : X \to Y \;\big|\; f(x) = \mathbf{1}\big(\pi_d(x) > \tfrac{1}{2}\big)\ \forall x \in X \Big\}.$$

Containt functions whose outputs are the indicator $1(\cdot)$, so the values are 0 or 1 such that functions $f$ take values in $\{0, 1\}$, so here one can take $\mathcal{Y} = \{0, 1\}$.
We are working with $0 - 1$ coding of the labels rather than $-1, 1$ then $\mathcal{Y} = \{-1, +1\}$.

## 5. Another similar hypothesis space

THe hypothesis space studied in basic statisttical learning is a linear classifier in feature space:

$$\mathcal{H}_{\mathrm{lin}} = \Big\{ f : X \to \{-1, +1\} \;\big|\; f(x) = \mathrm{sign}\big(w^\top x + b\big),\ w \in \mathbb{R}^d,\ b \in \mathbb{R} \Big\}.$$

Each function is defined by a simple rule ("predict $+1$ if some quantity of $x$ is above a threshold, otherwise $-1$"), but the quantity is linear in $x$, $w^\top x + b$, instead of a probability estimate $\pi_d(x)$.

## Exercise statement

Let $\mathcal{D} = \{Z_1, Z_2, \ldots, Z_n\}$ where $Z_i = (X_i, Y_i) \in \mathcal{X} \times \mathcal{Y}$ are i.i.d. from $p_Z(z)$.
Consider a random split of $\mathcal{D}$ into a training set $\mathcal{D}_{\mathrm{TR}}$ and a test set $\mathcal{D}_{\mathrm{TE}}$ such that

$$\mathcal{D} = \mathcal{D}_{\mathrm{TR}} \cup \mathcal{D}_{\mathrm{TE}}, \quad n = |\mathcal{D}_{\mathrm{TR}}| + |\mathcal{D}_{\mathrm{TE}}|.$$

Let $f : \mathcal{X} \to \mathcal{Y}$ be a mapping and $\ell(\cdot, \cdot)$ a loss function.

Define the empirical training and test risks

$$\hat{R}_{\mathrm{TR}}(f) = \frac{1}{|\mathcal{D}_{\mathrm{TR}}|} \sum_{i=1}^{n} \ell(Y_i, f(X_i)) \mathbf{1}(Z_i \in \mathcal{D}_{\mathrm{TR}}),$$

$$\hat{R}_{\mathrm{TE}}(f) = \frac{1}{|\mathcal{D}_{\mathrm{TE}}|} \sum_{j=1}^{n} \ell(Y_j, f(X_j)) \mathbf{1}(Z_j \in \mathcal{D}_{\mathrm{TE}}).$$

Let

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}_{\mathrm{TR}}(f).$$

Show that

$$\mathbb{E}\big(\hat{R}_{\mathrm{TR}}(\hat{f})\big) \;\le\; \mathbb{E}\big(\hat{R}_{\mathrm{TE}}(\hat{f})\big).$$

## Hint: definitions

Let $m = |\mathcal{D}_{\mathrm{TR}}|$ and $s = |\mathcal{D}_{\mathrm{TE}}|$.

Define

$$A = \frac{1}{m} \sum_{i=1}^{n} \ell(Y_i, f(X_i))\, \mathbf{1}(Z_i \in \mathcal{D}_{\mathrm{TR}}), \qquad B = \frac{1}{m} \sum_{j=1}^{n} \ell(Y_j, f(X_j))\, \mathbf{1}(Z_j \in \mathcal{D}_{\mathrm{TE}}).$$

Now let

$$\tilde{f} = \arg\min_{f \in \mathcal{F}} \hat{R}_{\mathrm{TE}}(f),$$

and define

$$C = \frac{1}{s} \sum_{i=1}^{n} \ell(Y_i, \tilde{f}(X_i))\, \mathbf{1}(Z_i \in \mathcal{D}_{\mathrm{TE}}), \qquad D = \frac{1}{m} \sum_{j=1}^{n} \ell(Y_j, \tilde{f}(X_j))\, \mathbf{1}(Z_j \in \mathcal{D}_{\mathrm{TR}}).$$

## 1: show that $\mathbb{E}(A) = \mathbb{E}(C)$

By definition,

$$B = \frac{1}{m} \sum_{i=1}^{n} \ell\big(Y_i, \hat{f}(X_i)\big)\, \mathbf{1}\big(Z_i \in D_{\mathrm{TE}}\big), \qquad C = \frac{1}{s} \sum_{i=1}^{n} \ell\big(Y_i, \hat{f}(X_i)\big)\, \mathbf{1}\big(Z_i \in D_{\mathrm{TE}}\big),$$

with $m = |D_{\mathrm{TR}}|$, $s = |D_{\mathrm{TE}}|$.

Each observation is i.i.d. and the train/test split is random and symmetric, so

$$\mathbb{P}\big(Z_i \in D_{\mathrm{TE}}\big) = \frac{s}{n}, \qquad \mathbb{E}\big[\mathbf{1}(Z_i \in D_{\mathrm{TE}})\big] = \frac{s}{n}.$$

Hence

$$\mathbb{E}(B) = \frac{1}{m} \sum_{i=1}^{n} \mathbb{E}\big[\ell(Y_i, \hat{f}(X_i))\big] \frac{s}{n}, \qquad \mathbb{E}(C) = \frac{1}{s} \sum_{i=1}^{n} \mathbb{E}\big[\ell(Y_i, \hat{f}(X_i))\big] \frac{s}{n},$$

which are equal, so $\mathbb{E}(B) = \mathbb{E}(C)$.

## 2: show that $\mathbb{E}(A) = \mathbb{E}(D)$

We have

$$A = \frac{1}{m} \sum_{i=1}^{n} \ell\big(Y_i, \hat{f}(X_i)\big)\, \mathbf{1}\big(Z_i \in D_{\mathrm{TR}}\big), \qquad D = \frac{1}{m} \sum_{j=1}^{n} \ell\big(Y_j, \tilde{f}(X_j)\big)\, \mathbf{1}\big(Z_j \in D_{\mathrm{TE}}\big).$$

The training and test subsets are chosen in the same random way, and $\hat{f}$ minimizes the empirical risk on $D_{\mathrm{TR}}$ while $\tilde{f}$ minimizes the empirical risk on $D_{\mathrm{TE}}$ within the same hypothesis class $\mathcal{H}$. Their joint distributions are therefore symmetric, which implies

$$\mathbb{E}(A) = \mathbb{E}(D).$$

## 3: show that $D \leq B$

By definition,

$$\tilde{f} = \arg\min_{f \in \mathcal{H}} R_T^E(f),$$

so for every $f \in \mathcal{H}$, and in particular for $f = \hat{f}$,

$$R_T^E(\tilde{f}) \leq R_T^E(\hat{f}).$$

But $R_T^E(\hat{f}) = B$ and $R_T^E(\tilde{f}) = D$, so $D \leq B$.

## 4: deduce that $\mathbb{E}(A) \leq \mathbb{E}(C)$

From step 3, $D \leq B$. Taking expectations and using steps 1˘2,

$$\mathbb{E}(D) \leq \mathbb{E}(B) \;\Rightarrow\; \mathbb{E}(A) \leq \mathbb{E}(C),$$

since $\mathbb{E}(A) = \mathbb{E}(D)$ and $\mathbb{E}(B) = \mathbb{E}(C)$.

Because $A = R_T^R(\hat{f})$ and $C = R_T^E(\hat{f})$, this is exactly

$$\mathbb{E}\big(R_T^R(\hat{f})\big) \;\leq\; \mathbb{E}\big(R_T^E(\hat{f})\big).$$