Model Assessment and Selection

- Error estimate - train and test error
- Loss functions
  - square error loss
  - 0-1 loss
  - log likelihood (crossentropy)
- Evaluation functions
  - train/test split
  - crossvalidation, one-leave-out
  - AIC, BIC - penalized training error estimate
  - (bootstrap) method
- Bias-variance tradeoff.

Vanilla Machine Learning without Hyperparameter Tuning

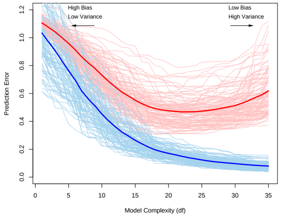
```
procedure FIND_ML_MODEL( $X, y$ )
    $X\_train0, y\_train, X\_test0, y\_test$
                        $\leftarrow train\_test\_split(X, y, train\_size = 2/3)$
    $preproc \leftarrow$ preprocessing method (standardization, ...)
    $X\_train = preproc.fit\_transform(X\_train0, y\_train)$
    $X\_test = preproc.transform(X\_test0)$
    $model = LinearRegression()$

    for  do
        for  do


        end for
    end for


    $model.fit(X\_train, y\_train)$
    return $model, test\_error(model.predict(X\_test), y\_test)$
end procedure
```

Vanilla Machine Learning with Hyperparameter Tuning

```
1: procedure FIND_ML_MODEL( $X, y$ )
2:     $X\_train0, y\_train, X\_test0, y\_test$
3:                         $\leftarrow train\_test\_split(X, y, train\_size = 2/3)$
4:     $preproc \leftarrow$ preprocessing method (standardization, ...)
5:     $X\_train = preproc.fit\_transform(X\_train0, y\_train)$
6:     $X\_test = preproc.transform(X\_test0)$
7:     $model = Ridge()$
8:     $X\_folds = split\_to\_folds(X\_train, y\_train)$          ▷ To select $\lambda$
9:     for each $fold$ in $X\_folds$ do
10:        for each $\lambda$ in $hyperparemeter\_grid$ do
11:            $model(\lambda).fit(X\_folds \setminus \{fold\})$
12:            $err(\lambda, fold) = error(model.predict(fold.X), fold.y)$
13:        end for
14:    end for
15:    $err(\lambda) = mean_{fold \in X\_folds}(err(\lambda, fold))$
16:    $\lambda^* = argmin_\lambda\ err(\lambda)$
17:    $model \leftarrow model(\lambda^*).fit(X\_train, y\_train)$
18:    return $model, test\_error(model.predict(X\_test), y\_test)$
19: end procedure
```

Model Assessment and Selection

- We assume i.i.d. data
  - independently (independent samples)
  - identically (the same distribution)
  - distributed
- Assume many iid datasets
  - 1 line= train/test curve for 1 set of samples (data)
  - data used to fit the model **Training error** is $\overline{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$,
  - data not used to fit the model **Test error** is $err = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$,
  - Test error is a point estimate of the generalization error.
  - Dark red line is an average of test errors, a more robust estimate.



Definition (Generalization error)

Generalization error is the expected prediction error over an independent test sample

$$Err = \mathbb{E}[L(Y, \hat{f}(X))]$$

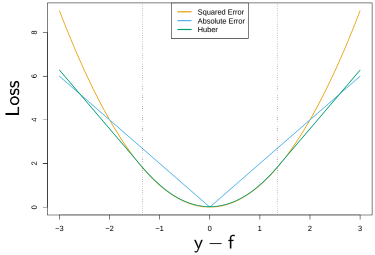
where both X and Y are drawn randomly from their joint distribution (population).

Loss functions for regression

Definition

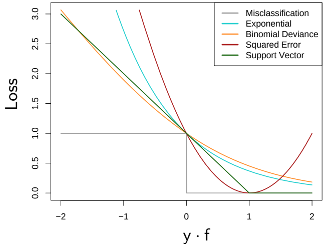
- **Square error loss** $L(y, \hat{y}) = (y - \hat{y})^2$
- **Absolute error loss** $L(y, \hat{y}) = |y - \hat{y}|$
- **Huber error loss**

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \textit{for } |y - \hat{y}| \le \delta, \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2, & \textit{otherwise.} \end{cases}$$



Classifier Assessment and Selection

- Qualitative response G taking one of K values labeled as $1, \dots, K$.
- Typically $\hat{G}(X) = \arg\max_k \hat{p}_k(X)$.
- Figure for binary response encoded $\{-1, +1\}$



Definition (Loss functions for classification)

- **0-1 loss, misclassification**

$$L(G, \hat{G}(X)) = I(G \neq \hat{G}(X))$$

- **log-likelihood, cross-entropy, deviance**

$$L(G, \hat{p}(X)) = -2I(G = k) \log \hat{p}_k(X)$$

$$\mathbb{E}[L_{\{0,1\}}(g, \hat{p})] = -2[p \log \hat{p} + (1-p) \log(1-\hat{p})]$$

- two classes encoded $\{-1, +1\}$
  - **exponential**

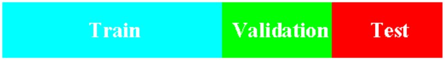
$$e^{-yf}$$

  - **support vector**

$$max(0, 1 - yf(x))$$

- Non-negative loss matrix $L \in \mathbb{R}^{K \times K}$ with 0 on the diagonal.

Data Rich Situation



If we have enough data, we split the dataset

- **Train** train the model
- **Validate** select appropriate model parameter α, λ, usually the model complexity, cost penalty
- **Test** estimate the test error on an independent sample.

Recommended ratios:

- $\frac{1}{2} : \frac{1}{4} : \frac{1}{4}$ with the validation set
- $\frac{2}{3} : \frac{1}{3}$ without the validation need.

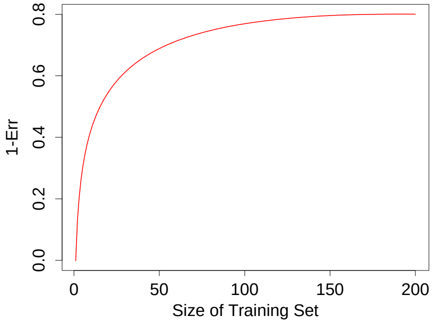
Stratified Selection

- Assume small disease prevalence.
    - We split the data healthy/sick and split each group separately.
- Consider a model over different branches of your company:
    - to estimate a new branch, all data from some branches should be selected as test ones
    - not a random sample from each branch.

Other methods only since we almost never have enough of the data.

Learning curve

- Do we have enough data?
- The learning curve suggests whether additional data would improve our model:
- curve flat close to maximal data size: small expected improvement.



Error Estimation with a few data

Amount of data needed depends on:

- true function complexity
- the noise ratio.

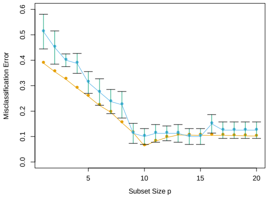
The estimation method depends on the purpourse:

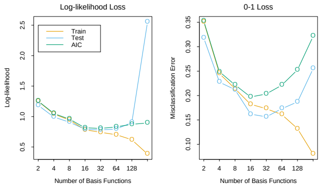
Model selection

- absolute value is not necessary, the difference is crucial
- any method can be used (AIC, BIC, cross-validation,. . .)

Test error estimation

- absolute value is necessary
- direct estimation (cross-validation, one-leave-out) preferred.



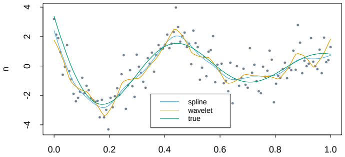


Bias Variance Decomposition

- Assume $Y = f(X) + \epsilon$ where $\mathbb{E}(\epsilon) = 0$ and $Var(\epsilon) = \sigma_\epsilon^2$.
- we derive the expected prediction error of the regression fit $\hat{f}(X)$ at an input point $X = x_0$ using squared-error loss:

$$\begin{aligned} Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\ &= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}(x_0) - f(x_0)]^2 + \mathbb{E}[\hat{f}(x_0) - \mathbb{E}\hat{f}(x_0)]^2 \\ &= \sigma_\epsilon^2 + Bias^2(\hat{f}(x_0)) + Var(\hat{f}(x_0)) \\ &= \text{ Irreducible Error } + \text{ Bias}^2 + \text{Variance.} \end{aligned}$$

- **bias** what is the model type unable to fit
- **model variance** a measure of the 'overfit' to train data



Bias Variance Decomposition

K-Nearest neighbour

$$\begin{aligned} Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\ &= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}(x_0) - f(x_0)]^2 + \mathbb{E}[\hat{f}(x_0) - \mathbb{E}\hat{f}(x_0)]^2 \\ &= \sigma_\epsilon^2 + [f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})]^2 + \frac{\sigma_\epsilon^2}{k}. \end{aligned}$$

Linear fit

$$\begin{aligned} Err(x_0) &= \mathbb{E}[(Y - \hat{f}(x_0))^2 | X = x_0] \\ &= \sigma_\epsilon^2 + [\mathbb{E}\hat{f}_p(x_0) - f(x_0)]^2 + \|\mathbf{h}(x_0)\|\sigma_\epsilon^2 \\ \mathbf{h}(x_0)\mathbf{y} &= (x_0^T(\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T})\mathbf{y} = \hat{f}_p(x_0). \end{aligned}$$

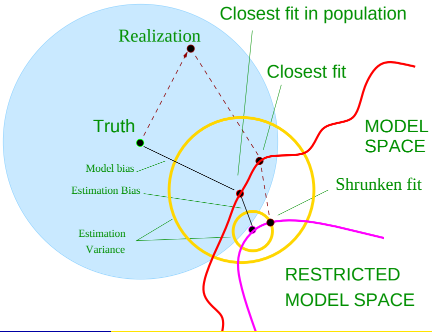
- hence $Var[\hat{f}_p(x_0)] = \|\mathbf{h}(x_0)\|\sigma_\epsilon^2$ and its average is $\frac{p}{N}\sigma_\epsilon^2$, hence

$$\frac{1}{N}\sum_{i=1}^{N} Err(x_i) = \sigma_\epsilon^2 + \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - \mathbb{E}\hat{f}(x_i)]^2 + \frac{p}{N}\sigma_\epsilon^2,$$

Penalized model complexity

- Consider ridge regression fit $\hat{f}_\lambda(x_0)$ with the fit
$$\mathbf{h}(x_0)\mathbf{y} = x_0^T(\mathbf{X^T X} + \lambda I)^{-1}\mathbf{X^T y}$$
- we break the bias more finely. Let β_* be the fit with $\lambda = 0$, $\beta_* = \arg\min_\beta \mathbb{E}(f(X) - \beta^T X)^2$.

$$\begin{aligned}\mathbb{E}_{x_0}[f(x_0) - \mathbb{E}\hat{f}_\lambda(x_0)]^2 &= \mathbb{E}_{x_0}[f(x_0) - \beta_*^T x_0]^2 + \mathbb{E}_{x_0}[\beta_*^T x_0 - \mathbb{E}\hat{\beta}_\lambda^T x_0]^2 \\ &= Ave[\text{Model Bias}]^2 + Ave[\text{Estimation Bias}]^2.\end{aligned}$$



Example: Bias-Variance Tradeoff

- 50 observations, 20 predictors, uniformly distributed in the hypercube $[0,1]^{20}$.

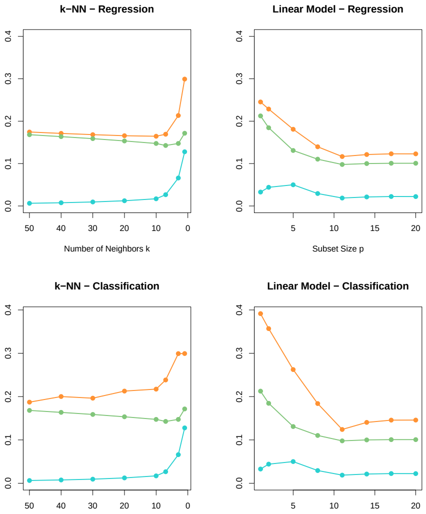
Left: Y is 0 if $X_1 \leq \frac{1}{2}$ and 1 if $X_1 > \frac{1}{2}$ and we apply k-NN

Right: Y is 1 if $\sum_{j=1}^{10} X_j > 5$ and 0 otherwise, and we use best subset linear regression of size p.

orange: error (mean square resp. 0-1)

green: squared bias

blue: estimation variance



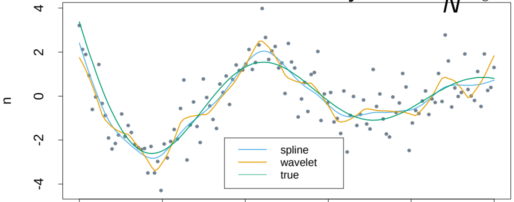
- For classification, only the error differ compared to regression.

Optimism of the Training Error Rate

- **training error** $\overline{err} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$
- usually less than the true error $Err = \mathbb{E}[L(Y, \hat{f}(X))]$.
- we keep x_i points fixed, we take new sample of $\mathbf{y}$ at these points.
- **in-sample error** $Err_{in} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{\mathbf{y}}\mathbb{E}_{Y^{new}} L(Y_i^{new}, \hat{f}(x_i))$.
- **optimism** $op \leftarrow Err_{in} - \mathbb{E}_{\mathbf{y}}(\overline{err})$.
- for squared error, 0-1, and other loss functions: $op = \frac{2}{N}\sum_{i=1}^{N} cov(\hat{y}_i, y_i)$
- For a linear fit with $d = p$ inputs of basis functions and additive error model $Y = f(X) + \epsilon$ it simplifies

$$\sum_{i=1}^{N} cov(\hat{y}_i, y_i) = d\sigma_\epsilon^2$$

$$Err_{in} = \mathbb{E}_{\mathbf{y}}\overline{err} + 2\frac{d}{N}\sigma_\epsilon^2 = C_p = AIC_{gauss}..$$



The Optimism (Just for fun)

$$y_i - \hat{y}_i = (y_i - f(x_i)) + (f(x_i) - \mathbb{E}\hat{f}(x_i)) + (\mathbb{E}\hat{f}(x_i) - \hat{y}_i).$$

- We get six terms of the sum of squares:

$$\begin{array}{ll} A_1 = \sum_i (y_i - f(x_i))^2 & D_1 = 2\sum_i (y_i - f(x_i))(f(x_i) - \mathbb{E}\hat{f}(x_i)) \\ B = \sum_i (f(x_i) - \mathbb{E}\hat{f}(x_i))^2 & E = 2\sum_i (f(x_i) - \mathbb{E}\hat{f}(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i) \\ C = \sum_i (\mathbb{E}\hat{f}(x_i) - \hat{y}_i)^2 & F_1 = 2\sum_i (y_i - f(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i) \end{array}$$

- For the new sample Y, taking the expectation.
- $A_2 = \sum_i \mathbb{E}_{Y^0}(Y_i^0 - f(x_i))^2$
- and similarly D_2, F_2.

$$\begin{aligned} N(Err_{in} - \overline{err}) &= (A_2 + B + C + D_2 + E + F_2) - (A_1 + B + C + D_1 + E + F_1) \\ &= (A_2 - A_1) + (D_2 - D_1) + (F_2 - F_1) \end{aligned}$$

- $\mathbb{E}(A_1) = \mathbb{E}(A_2) = N\sigma_\epsilon^2$,
- $\mathbb{E}(D_1) = 2\sum_i (\mathbb{E}(y_i) - f(x_i))(f(x_i) - \mathbb{E}\hat{f}(x_i)) = 0$ since $\mathbb{E}(y_i) = f(x_i)$
- $\mathbb{E}(D_2) = 0$ as well.
- $F_2 = 2\sum_i \mathbb{E}_{Y^0}\left[(Y_i^0 - f(x_i))(\mathbb{E}\hat{f}(x_i) - \hat{y}_i)\right] = 0$
  - as $\mathbb{E}(Y_i^0) = f(x_i)$ and Y_i^0 and $\hat{y}_i$ are independent.
- $\mathbb{E}(F_1) = -2\sum_i^N cov(y_i, \hat{y}_i)$.

The Covariance (Just for fun)

$$\begin{aligned} cov(\hat{\mathbf{y}}, \mathbf{y}) &= cov((\mathbf{X}(\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T})\mathbf{y}, \mathbf{y}) = (\mathbf{X}(\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}\mathbf{X}^\mathbf{T}) cov(\mathbf{y}, \mathbf{y}) \\ cov(y, y) &= \sigma_\epsilon^2. \end{aligned}$$

- The values $cov(\hat{y}_i, y_i)$ are the diagonal values of the above matrix $cov(\hat{\mathbf{y}}, \mathbf{y})$. Thus

$$\begin{aligned} \sum_{i=1}^{N} cov(\hat{y}_i, y_i) &= trace(\mathbf{X}(\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}(\mathbf{X}^T))\sigma_\epsilon^2 \\ &= trace((\mathbf{X}^\mathbf{T}\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X}))\sigma_\epsilon^2 \\ &= trace(I_d)\sigma_\epsilon^2 = d\sigma_\epsilon^2. \end{aligned}$$

- Similarly,

$$\sum_{i=1}^{N} cov(\mathbf{S}_\lambda y_i, y_i) = trace(\mathbf{S}_\lambda).$$

https://www.waxworksmath.com/Authors/G_M/Hastie/WriteUp/Weatherwax_Epstein_Hastie_Solution_Manual.pdf

AIC Akaike Information Criterion

Definition (AIC)

The **AIC Akaike Information Criterion** is defined

- Logistic regression, binomial log likelihood

$$AIC = -\frac{2}{N} loglik + 2\frac{d}{N}.$$

- Gaussian model with variance σ_ϵ^2

$$AIC(\lambda) = \overline{err(\lambda)} + 2\frac{d(\lambda)}{N}\sigma_\epsilon^2.$$

  the sum of the training error $\overline{err(\lambda)}$ and the complexity penalty for $d, d(\lambda)$ model parameters, N samples, $loglik$ the logarithm of likelihood.

- The effective number of parameters
  - for a linear fit $\widehat{\mathbf{y}} = \mathbf{S}\mathbf{y}$ is $d(\mathbf{S}) = trace(\mathbf{S})$, the sum of the diagonal elements.
- For 0-1 loss does not hold in general, only as approximation.

BIC Bayesian Information Criterion

$$\begin{aligned} P(\mathcal{M}_m|\mathbf{Z}) &= \frac{P(\mathbf{Z}|\mathcal{M}_m)\cdot P(\mathcal{M}_m)}{P(\mathbf{Z})} \\ &\propto P(\mathbf{Z}|\mathcal{M}_m)\cdot P(\mathcal{M}_m) \\ &\propto P(\mathcal{M}_m)\cdot \int P(\mathbf{Z}|\theta_m,\mathcal{M}_m)P(\theta_m|\mathcal{M}_m)d\theta_m \end{aligned}$$

Laplace approximation to the integral gives with $\hat{\theta}_m$ the ML estimate of θ:

$$\begin{aligned} \log P(\mathbf{Z}|\mathcal{M}_m) &= \log P(\mathbf{Z}|\hat{\theta}_m,\mathcal{M}_m) - \frac{d_m}{2}\cdot \log N + O(1) \\ &= loglik - \frac{d_m}{2}\cdot \log N + O(1) \end{aligned}$$

Definition (Bayesian Information Criterion (BIC))

$$BIC_m = -2loglik_m + (\log N)\cdot d_m$$

BIC may be used to compare the model posterior probabilities $\frac{e^{\frac{1}{2}\cdot BIC_m}}{\sum_{\ell=1}^{M} e^{\frac{1}{2}\cdot BIC_\ell}}$.

Crossvalidation

Crossvalidation

- Split the data into K roughly equal-sized parts. Usually, $K = 5$ or 10 or $K = N$
  - For $K = 10$ it is called **tenfold** crossvalidation.
  - For $K = N$ it is called **one-leave-out** crossvalidation.
  - $\kappa : \{1, \dots, N\} \to \{1, \dots, K\}$ is the partition function
  - $\hat{f}^{-k}$ is the fitted function with kth part removed.
- For $k = 1, \dots, K$
  - For the kth part we fit the model to the other $K - 1$ parts of the data, and calculate the prediction error of the fitted model when predicting the kth part of the data.
- Average the error estimates.

$$CV = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i)).$$

10 times 10 Crossvalidation

Parameter Tuning by Crossvalidation

Parameter Tuning by Crossvalidation

- Given a set of models $f(x, \alpha)$ indexed by a tuning parameter α, we define

$$CV(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, \alpha)).$$

- The $CV(\alpha)$ provides an estimate of the test error curve and we find the tuning parameter $\hat{\alpha}$ that minimizes it.
- Our final model is $f(x, \hat{\alpha})$.

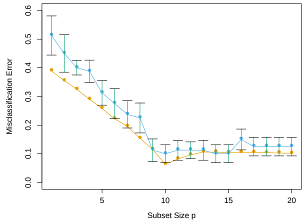


FIGURE 7.9. *Prediction error (orange) and tenfold cross-validation curve (blue) estimated from a single training set, from the scenario in the bottom right panel of Figure 7.3.*

Definition (One standard error rule)

We choose the most parsimonious model whose error is no more than one standard error above the error of the best model.

Here, $p = 9$.

Generalized Crossvalidation

- For a linear fitting method we can write

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}.$$

- For many linear fitting methods,

$$\frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{f}^{-i}(x_i)]^2 = \frac{1}{N}\sum_{i=1}^{N}[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}]^2,$$

- The **generalized crossvalidation** *GCV* approximation is

$$GCV = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - trace(\mathbf{S})/N}\right]^2.$$

Bad and good Crossvalidation

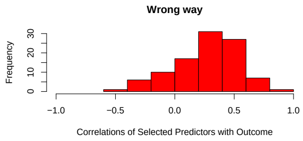
Our data

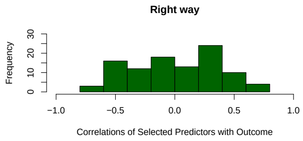
- $p = 500$ dimensions
- $N = 20$
- random data, y independent of x.

Wrong way of the crossvalidation

1. choose 100 good predictors
2. Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.
3. Use the classifier to predict the class labels for the samples in fold k.

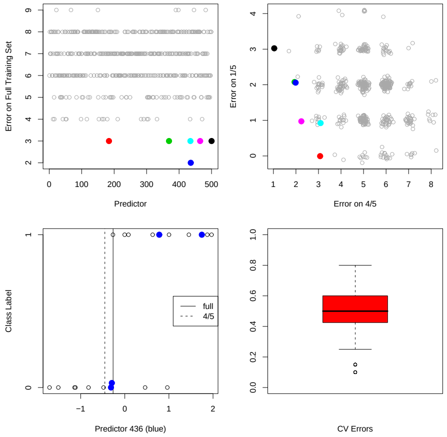
- This approach achieves the crossvalidation error 3%.
- What is the true error?
- What is the correct crossvalidation approach?





Best Decision Stump Example

- Left top: Best decision stumps on full data
- 1/5 and 4/5 split
- bottom left: The stump fitted on 4/5 produces two missclassified validation data (blue)
- The crossvalidation error in 50 experiments is around 0.5, as it should be.



One–leave–out

- Use all but one data samples for learning.
- Evaluate the error or the hidden sample.
- Repeat for each sample and calculate the average.

Advantage:

- The largest possible training set.
- Deterministic evaluation (no sense to repeat it).

Disadvantage:

- Time consuming.
- May be misleading – take randomly $50 : 50$ generated goal G, the one–leave–out error is 100%.

Summary

- Error estimate - train and test error
- Loss functions
  - square error loss
  - 0-1 loss
  - log likelihood (crossentropy)
- Evaluation functions
  - train/test split
  - crossvalidation, one-leave-out
  - AIC, BIC - penalized training error estimate
  - (bootstrap) method
- Bias-variance tradeoff.

Bootstrap

- Select elements with replacement.
- We have N data samples, we select with replacement N samples – some are selected more than one, some are not selected at all. *The not selected are used for testing.*
- The probability of not-selecting a sample is $\left(1 - \frac{1}{N}\right)^N \approx e^{-1} = 0.368$.
- The error estimate is pessimistic since we learn a model on N samples that come from only 0.632 samples.
  The usual error estimate is:
  $$err = 0.632 \cdot e_{\text{test}} + 0.368 \cdot e_{\text{train}}$$
- Again, may be misled by similar data as one–leave–out.

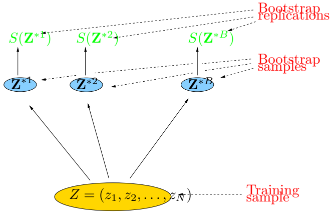


FIGURE 7.12. *Schematic of the bootstrap process. We wish to assess the statistical accuracy of a quantity $S(\mathbf{Z})$ computed from our dataset. B training sets $\mathbf{Z}^{*b}$, $b = 1, \dots, B$ each of size N are drawn with replacement from the original dataset. The quantity of interest $S(\mathbf{Z})$ is computed from each bootstrap training set, and the values $S(\mathbf{Z}^{*1}), \dots, S(\mathbf{Z}^{*B})$ are used to assess the statistical accuracy of $S(\mathbf{Z})$.*

Confusion Matrix

true class \ prediction	$+$	-
$+$	TP – true positive	FN – false negative
-	FP – false positive	TN true negative

Česky se říká správně/falešně positivní/negativní.

Basic measures:

celková správnost	accurancy	$Acc = \frac{TP+TN}{TP+TN+FP+FN}$
chyba	error	$Err = \frac{FP+FN}{TP+TN+FP+FN}$
přesnost	precision	$Prec = \frac{TP}{TP+FP}$
úplnost, sensitivita	recall, sensitivity	$Rec = \frac{TP}{TP+FN}$
specificita	specificity	$Specificity = \frac{TN}{TN+FP}$
F míra	F measure	$F = \frac{2\cdot Prec\cdot Rec}{Prec+Rec} = \frac{2\cdot TP}{2\cdot TP+FP+FN}$
	TP rate	$\frac{TP}{TP+FN}$
	FP rate (=1–Specificity)	$\frac{FP}{FP+TN}$

Additional Remarks

- ROC - curve
  - Precision recall curve
- Interval estimates
- t-test
  - paired t-test
- ANOVA
- Q-Q plot
- $R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \overline{y}_i)^2}$, 1 or 0 instead of nan and infinity.
  - explained variance - identical for zero mean residuals.
- Mc Nemar's test (two classifiers, confusion matrix with values <5; i.e. not χ^2 test).
  - statsmodels.stats.contingency_tables.mcnemar()

Loss Matix, Receiver-Operating-Curve

The cost of missclassification may be different for each class. The general loss specification is a **loss matrix** $L_{kk^{|}}$, an element represent the cost of classifying k as $k^{|}$. Must be zero at the diagonal, non-negative everywhere.

- we can modify $Gini(m) = \sum_{k \neq k^{|}} L_{kk^{|}} \hat{p}_{mk} \hat{p}_{mk^{|}}$
- or weight the data samples k $L_{kk^{|}}$ times (only in binary classification)
- we classify according to $k(m) = argmin_k \sum_l L_{lk} \hat{p}_{ml}$ in the leaves.

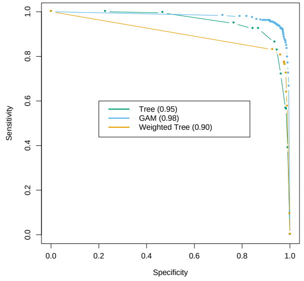


FIGURE 9.6. *ROC curves for the classification rules fit to the* spam *data. Curves that are closer to the north-east corner represent better classifiers. In this case the GAM classifier dominates the trees. The weighted tree achieves better sensitivity for higher specificity than the unweighted tree. The numbers in the legend represent the area under the curve.*

List of topics

1. Linear, ridge, lasso regression, k-neares neighbours,(formulas) overfitting, curse of dimensionality, (LARS)
2. Splines - the base, natural splines, smoothing splines; kernel smoothing: kernel average, Epanechnikov kernel.
3. Logistic regression, Linear discriminant analysis, generalized additive models
4. Train/test error and data split, square error, 0-1, crossentropy, AIC, BIC,(formulas) crossvalidation, one-leave-out CV, wrong estimate example
5. decision trees, information gain/entropy/gini, CART prunning,(formulas)
6. random forest (+bagging), OOB error, Variable importance, boosting (Adaboost(formulas) and gradient boosting), stacking, MARS,
7. Bayesian learning: MAP, ML hypothesis (formulas), Bayesian optimal prediction, EM algorithm
8. Clustering: k-means, Silhouette, k-medoids, hierarchical
9. Apriori algorithm, Association rules, support, confidence, lift
10. Inductive logic programming basic: hypothesis space search, background knowledge, necessity, sufficiency and consistency of a hypothesis, Aleph
11. Undirected graphical models, Graphical Lasso procedure, deviance, MRF
12. Gaussian processes: estimation of the function and its variance (figures, ideas).

List of topics

1. Linear, ridge, lasso regression, k-neares neighbours,(formulas) overfitting, curse of dimensionality, (LARS)
2. Splines - the base, natural splines, smoothing splines; kernel smoothing: kernel average, Epanechnikov kernel.
3. Logistic regression, Linear discriminant analysis, generalized additive models
4. Train/test error and data split, square error, 0-1, crossentropy, AIC, BIC,(formulas) crossvalidation, one-leave-out CV, wrong estimate example
5. decision trees, information gain/entropy/gini, CART prunning,(formulas)
6. random forest (+bagging), OOB error, Variable importance, boosting (Adaboost(formulas) and gradient boosting), stacking, MARS,
7. Bayesian learning: MAP, ML hypothesis (formulas), Bayesian optimal prediction, EM algorithm
8. Clustering: k-means, Silhouette, k-medoids, hierarchical
9. Apriori algorithm, Association rules, support, confidence, lift
10. Inductive logic programming basic: hypothesis space search, background knowledge, necessity, sufficiency and consistency of a hypothesis, Aleph
11. Undirected graphical models, Graphical Lasso procedure, deviance, MRF
12. Gaussian processes: estimation of the function and its variance (figures, ideas).

Table of Contens