Multidimensional Signal Analysis Part III – Lecture 03

October 30, 2003

Contents

References

[Efron and Tibshirani, 1995] Efron, B. and Tibshirani, R. (1995). Cross-validation and the bootstrap: estimating the error rate of a prediction rule.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York.

[Theodoridis and Koutroumbas, 1999] Theodoridis, S. and Koutroumbas, K. (1999). *Pattern Recognition*. Academic Press, San Diego.

[Wehrens et al., 2000] Wehrens, R., Putter, H., and Buydens, L. M. C. (2000). The bootstrap: a tutorial. *Chemometrics and Intelligent Laboratory Systems*, 54:35–52.

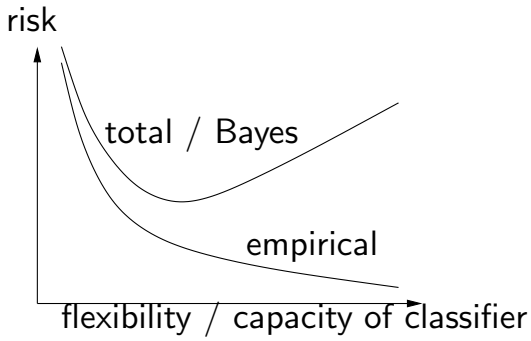
1 Model selection and assessment

[Hastie et al., 2001, 7]

— that is, to identify the best model and to evaluate the final model

Typical observation (the **bias-variance dilemma**):

- models with too little flexibility lead to large prediction error on training set (**empirical risk**) and test set (approximation to **total / Bayes risk**)
- models with too much flexibility lead to low prediction error on training set and large prediction error on test set.



Total risk = empirical risk + optimism

1.1 The bias-variance trade-off

formulated for regression: true mean $\mathbb{E}[y|x]$ is approximated by $f^{\mathcal{T}}(x)$ which depends on the current training set $\mathcal{T}$.

The mean square error is

$$\begin{aligned}
&\mathbb{E}_{\mathcal{T}}\left[\left(f^{\mathcal{T}}(x) - \mathbb{E}[y|x]\right)^2\right] \\
&= \mathbb{E}_{\mathcal{T}}\left[\left(f^{\mathcal{T}}(x) - \mathbb{E}_{\mathcal{T}}[f^{\mathcal{T}}(x)] + \mathbb{E}_{\mathcal{T}}[f^{\mathcal{T}}(x)] - \mathbb{E}[y|x]\right)^2\right] \\
&= \mathbb{E}_{\mathcal{T}}\left[\left(f^{\mathcal{T}}(x) - \mathbb{E}_{\mathcal{T}}[f^{\mathcal{T}}(x)]\right)^2 + \left(\mathbb{E}_{\mathcal{T}}[f^{\mathcal{T}}(x)] - \mathbb{E}[y|x]\right)^2\right] \\
&\quad + \mathbb{E}_{\mathcal{T}}\left[2\left(f^{\mathcal{T}}(x) - \mathbb{E}_{\mathcal{T}}[f^{\mathcal{T}}(x)]\right)\left(\mathbb{E}_{\mathcal{T}}[f^{\mathcal{T}}(x)] - \mathbb{E}[y|x]\right)\right]
\end{aligned}$$

The last term can be multiplied out

$$2\mathbb{E}_{\mathcal{T}}\left[f^{\mathcal{T}}(x)\right]^2 - 2\mathbb{E}_{\mathcal{T}}\left[f^{\mathcal{T}}(x)\right]\mathbb{E}[y|x]$$
$$-2\mathbb{E}_{\mathcal{T}}\left[f^{\mathcal{T}}(x)\right]^2 + 2\mathbb{E}_{\mathcal{T}}\left[f^{\mathcal{T}}(x)\right]\mathbb{E}[y|x] = 0$$

so that only

variance

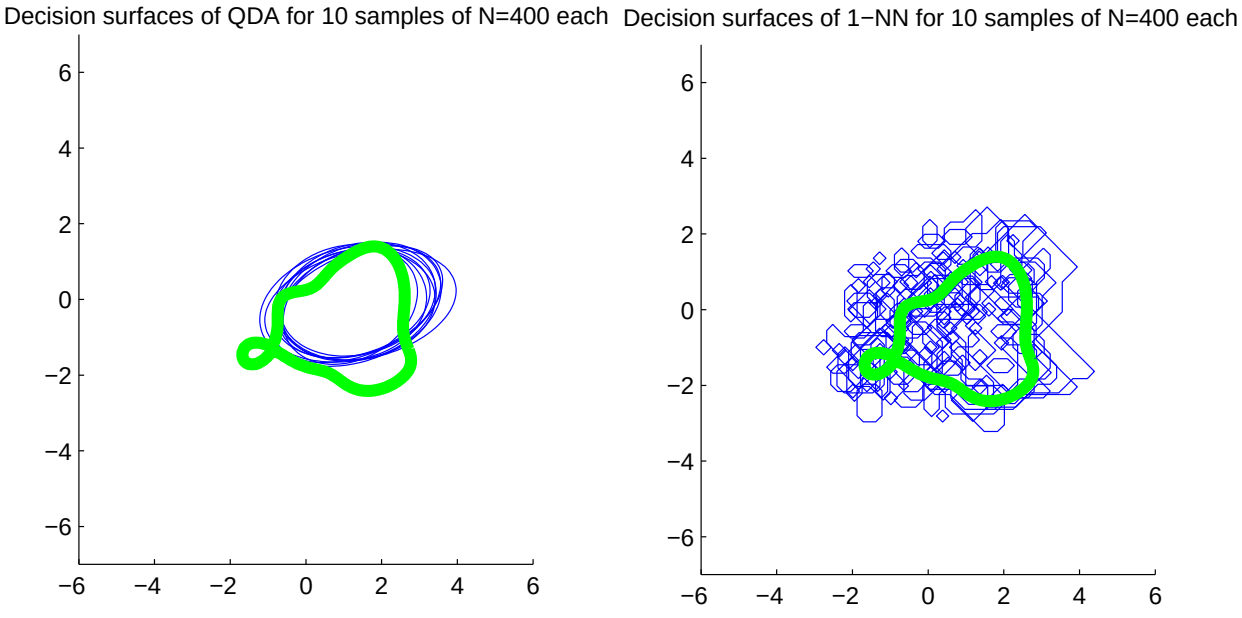
$$\mathbb{E}_{\mathcal{T}}\left[\left(f^{\mathcal{T}}(x) - \mathbb{E}_{\mathcal{T}}[f^{\mathcal{T}}(x)]\right)^2\right]$$

and **squared bias**

$$\left(\mathbb{E}_{\mathcal{T}}[f^{\mathcal{T}}(x)] - \mathbb{E}[y|x]\right)^2$$

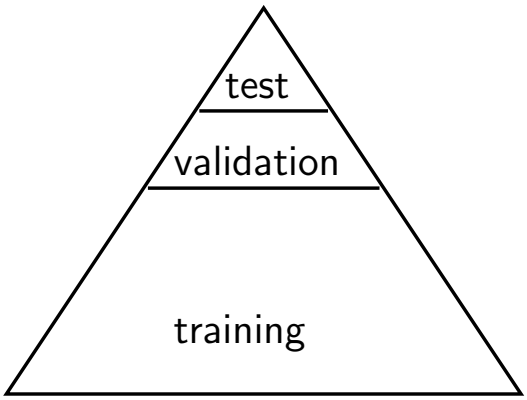
remain.

Example from the "complex" two-class data set: Bayes decision boundary with decision boundaries from QDA (some bias, low variance) and 1-NN (no bias, high variance) for 10 different training sets.



1.2 Model assessment

Given enough data, split it:



If not enough data, estimate generalization performance

- analytically (Akaike information criterion, etc.)
- by reusing samples:
  - cross-validation
  - bootstrap

1.3 Akaike information criterion (AIC) and variants

If not enough data, estimate generalization performance analytically:

$$\begin{aligned} \text{deviance} &:= 2 \text{ log likelihood ratio} \\ &= 2 \log \frac{\mathcal{L}(\text{true model})}{\mathcal{L}(\text{current model})} \end{aligned}$$

Estimate of deviance based on training set is too small (because parameters were chosen to minimize it on the training set) by 2·(# independent free parameters)$/N$ iff the true model is contained in the parametric family.

If the true model is outside the parametric family, AIC chooses ever larger models as more data becomes available.

AIC is asymptotically equivalent to cross-validation.

BIC Bayes information criterion: tougher penalization of complex models, useful if true model is outside parametric family; like AIC, but with "$\log N$" instead of "2".

NIC network information criterion: estimate # free parameters in nonlinear models, from Hessian of log-likelihood

MDL minimum description length: penalize deviance by minimum description length of binary code needed to represent a model (formally identical to BIC)

SRM structural risk minimization: see upcomping lecture

1.4 Cross-validation (CV)

simple and powerful

- Partition data into K pieces. Predict class membership in one piece by training on all others. Repeat.
- Typically, $K = 5 \ldots 10$.
- For $K = N$(# training feature vectors), **leave-one-out (LOO) cross-validation** results.
- Choosing $K \ll N$ is faster and has lower variance but more bias[a], especially for small N.

[a] The estimate becomes overly pessimistic because not all available data are used for training. Bias can be estimated using the "jackknife".

1.5 Bootstrap

- Sample from training set *with* replacement. Train a classifier on that set, and calculate "leave-one-out prediction error" on samples that are *not* in the set.
- Repeat 50-100 times, and estimate parameters
- Obtain point or interval estimates for the above parameters [Wehrens et al., 2000]

1.5.1 The .632 estimator for prediction error

Compute

$(1 - 0.632)\times$(training error) $+ 0.632\times$(leave-one-out prediction error)

- balances optimism (from inclusion of training error) with pessimism (because not all of the data is used in training)
- works well as long as there is no overfitting

Why .632?

Probability that object from the training set is in a bootstrap sample is

$$\begin{aligned} & 1 - \text{probability that it is not included} \\ = \quad & 1 - \left(\frac{N-1}{N}\right)^N \\ \simeq \quad & 1 - \frac{1}{e} = 0.632 \end{aligned}$$

1.5.2 The .632+ estimator for prediction error

The above estimate breaks down in overfitting. An amendment is

$$(1-\hat{\omega})\times(\text{training error}) + \hat{\omega}\times(\text{leave-one-out prediction error})$$

with

$$\begin{aligned}\hat{\omega} &= \frac{.632}{1-.368\hat{R}} \\ 0 \leq \hat{R} &= \frac{\text{leave one out prediction error} - \text{training error}}{\text{no information error rate} - \text{training error}} \leq 1\end{aligned}$$

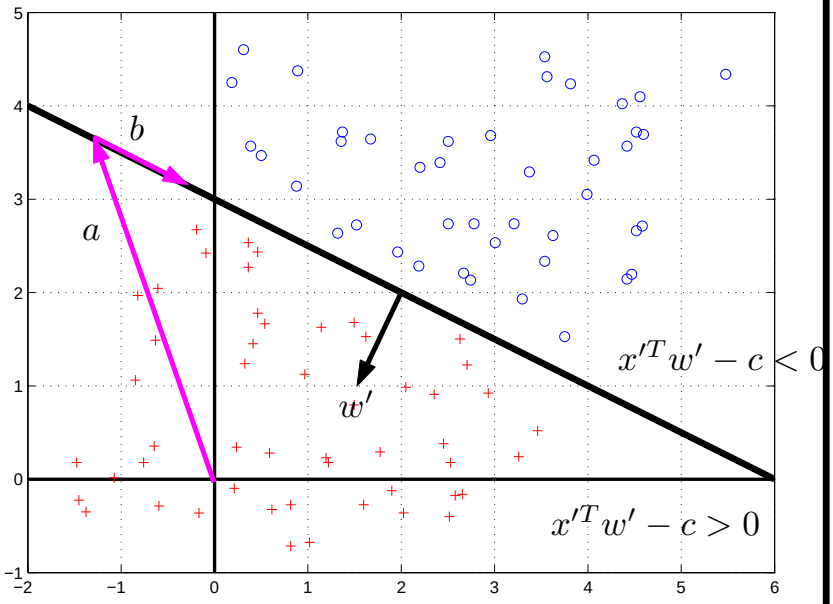
The authors note that "all of this is a reasonable plausibility argument for the 632+ rule, but not much more than that" [Efron and Tibshirani, 1995].

2 The perceptron

[Theodoridis and Koutroumbas, 1999, 3.3]

Discrimination between two classes assumed linearly separable

$$\begin{aligned}x' &= a + \beta b \\ x'^T w' &= \underbrace{a^T w'}_{c} + \beta \underbrace{b^T w'}_{0} \\ x'^T w' - c &= 0\end{aligned}$$



Linear separability given if

$$w'^T x'_i - c > 0 \qquad \forall x'_i \in \text{class } 1$$
$$w'^T x'_i - c < 0 \qquad \forall x'_i \in \text{class } 2$$

Equivalently,

$$w'^T x' - c = [w_1 \quad \cdots \quad w_D] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix} - c = [w_1 \quad \cdots \quad w_D \quad -c] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix}$$
$$= w^T x$$

so that

$$w^T x_i > 0 \qquad \forall x_i \in \text{class } 1$$
$$w^T x_i < 0 \qquad \forall x_i \in \text{class } 2$$

To optimize the linear discriminant function, choose a loss function, e.g.

$$E(w) = \sum_{i \in \text{misclassified}} |w^T x_i|$$
$$= \sum_{i \in \text{misclassified}} k_i w^T x_i$$
$$\text{with} \quad k_i = \begin{cases} +1 \forall x_i \in \text{class } 1 \\ -1 \forall x_i \in \text{class } 2 \end{cases}$$

The loss function is continuous but not differentiable everywhere.

Minimize the loss function using steepest descent

$$w(t+1) = w(t) - \rho_t \frac{\partial E(w)}{\partial w}\bigg|_{w=w(t)}$$

with t – iteration index and ρ – stepsize.

At all points at which the derivative exists,

$$\frac{\partial E(w)}{\partial w} = \sum_{i \in \text{misclassified}} k_i x_i$$

such that

$$w(t+1) = w(t) - \rho_t \sum_{i \in \text{misclassified}} k_i x_i$$

which is defined at all points.

Demo: `nnd4pr`

It can be proven that if the sequence ρ_t is chosen such that

$$\lim_{t\to\infty} \sum_{k=0}^{t} \rho_k^2 < \infty, \quad \lim_{t\to\infty} \sum_{k=0}^{t} \rho_k \to \infty$$

$$\left(\text{e.g. } \rho_t = \frac{\text{const.}}{t}\right)$$

then the algorithm converges in a finite number of steps, but

- convergence can be slow
- and is proven to exist only for linearly separable classes: can fail for non-linearly separable data
  ($\rightarrow$ "pocket-algorithm" which stores the best previous solution)

The steepest descent algorithm introduced above processes the *entire* training set in *each* iteration: it is a **batch algorithm**.

Online or **pattern algorithms** adjust the parameters after each presentation of a single training vector.

Online algorithms are order-dependent. Random selection of training vectors for presentation can help find better local minima.