INFO-F-422 Statistical foundations of machine learning

Supervised learning

Gianluca Bontempi

Machine Learning Group
Computer Science Department
mlg.ulb.ac.be

Nonlinear input/output problems

- So far we have considered supervised problems where the relation between inputs and output is linear

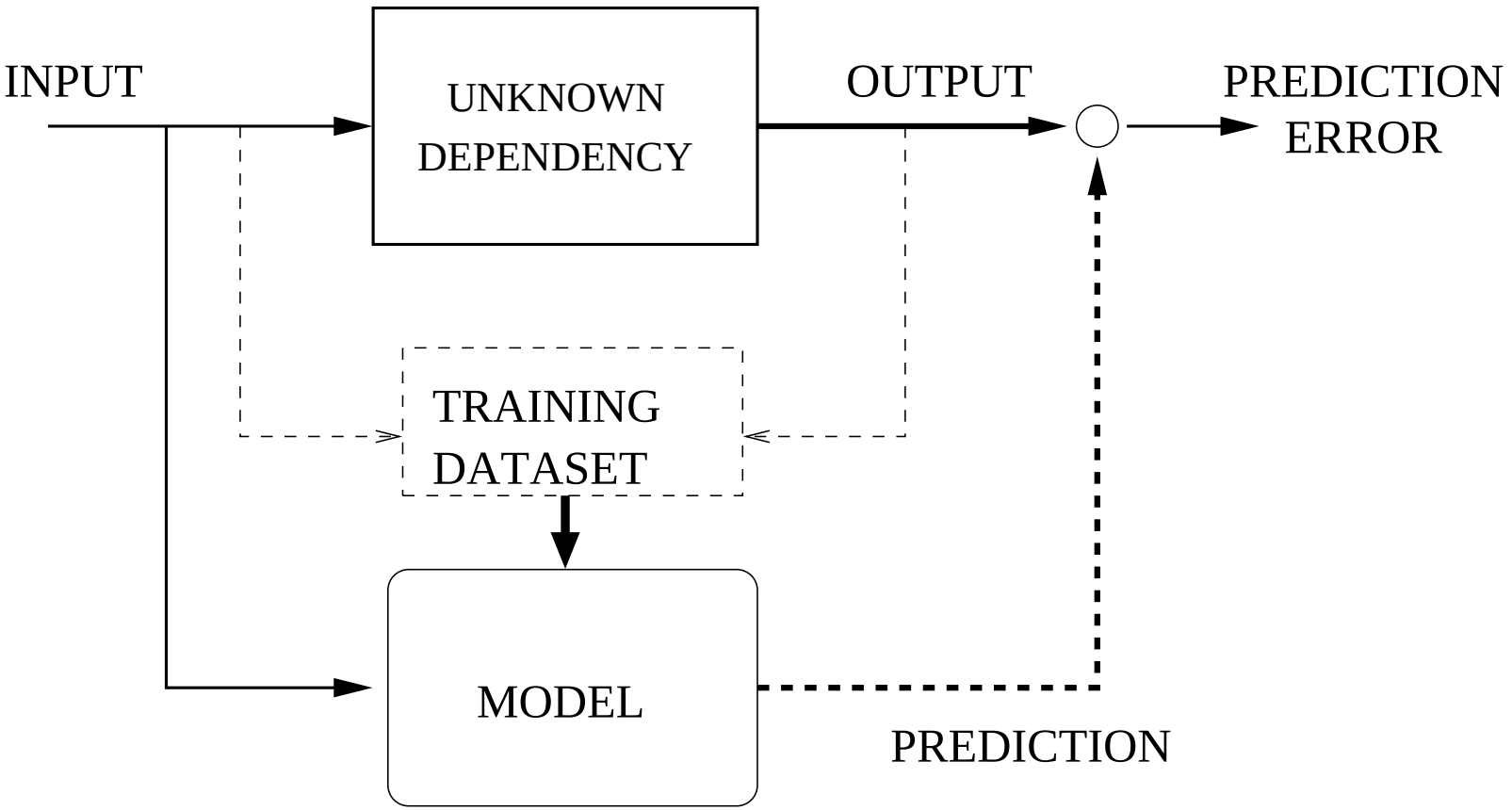
$$\mathbf{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \mathbf{w}$$

- The advantage of linear models are numerous:
  - the least-squares $\hat{\beta}$ estimate can be expressed in an analytical form
  - the least-squares $\hat{\beta}$ estimate can be easily calculated through matrix computation.
  - statistical properties of the estimator can be easily defined.
  - recursive formulation for sequential updating are available.
  - the relation between empirical and generalization error is known.
  - fast leave-one-out (PRESS statistic).

Nonlinear input/output problems

- Unfortunately in real problem it is extremely unlikely that the input and ouput variables are linked by a linear relation.
- Relation is often unknown and only a limited amount of samples is available.
- Using polynomial models is **not effective** since in a problem with n input features, a polynomial of degree d requires the estimation of $O(n^d)$ parameters.
- Nonlinear input/output transformations are effective only if specific domain knowledge.

Supervised learning



Supervised learning: the actors

- Multidimensional input $x \in \mathcal{X} \subset \mathbb{R}^n$ and scalar output $y \in \mathbb{R}$.
- Finite number of noisy input/output observations (*training set* D_N).
- No a priori knowledge on the process underlying the data.
- Test set of input values for which an accurate *generalization* or *prediction* of the output is required.

Formulation of a learning task

- Data *generator* of i.i.d input vectors $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$ according to some unknown (but fixed) probability distribution $F_{\mathbf{x}}(x)$.
- *Target* operator, which transforms the input $\mathbf{x}$ into the output value $\mathbf{y} \in \mathcal{Y} \subset \Re$ according to some unknown (but fixed) conditional distribution $F_{\mathbf{y}}(y|x)$.
- *Training set* $D_N = \{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots, \langle x_N, y_N\rangle\}$ made of N pairs $\langle x_i, y_i\rangle \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ independent and identically distributed (i.i.d) according to the joint distribution

$$F_{\mathbf{z}}(z) = F(\langle x, y\rangle)$$

The regression plus noise form

- Stochastic input/output relation

$$\mathbf{y} = f(x) + \mathbf{w}$$

  where $f(x) = E[\mathbf{y}|\mathbf{x} = x]$ is a deterministic function and $\mathbf{w}$ is the noise or (random error) is independent of $\mathbf{x}$ and $E[\mathbf{w}] = 0$.
- **Training set** $\{\langle x_i, y_i \rangle : i = 1, \ldots, N\}$, where $x_i = [x_{i1}, \ldots, x_{in}]^T$, generated according to the previous model.
- Goal: find a model $h(x)$ which is able to give a good approximation of the unknown $f(x)$.
- But how to choose h, if unknown data distribution and limited training set?

What is a good model?

- To assess $h(x)$ we need a **cost (or loss) function** C associated with $f(x)$ and $h(x)$.
- Cost function measures the discrepancy between the outputs of the functions $f(\cdot)$ and $h(\cdot)$, given an input x.
- Cost function can have different forms, e.g. quadratic

$$C(f(x), h(x, \alpha)) = (f(x) - h(x, \alpha))^2$$

- If the input/output relationship were deterministic (no noise) the error of $h(x, \alpha)$ is

$$\int_{\mathcal{X}} C(f(x), h(x, \alpha)) dx$$

- Unfortunately, the stochastic case is more complex.

The MSE of a nonlinear model

- Learning algorithm: given dataset D_N returns the parameters α_N of model $h(x, \alpha_N)$.
- $\mathbf{D}_N$ is a random vector then discrepancy between $f(\cdot)$ and $h(\cdot, \alpha_N)$ have to be computed in statistical terms.
- For a given x, **mean-square error** is quadratic cost C averaged over all training sets with N samples and all test samples $\mathbf{y}$:

$$\mathrm{MSE}(x) = E_{\mathbf{D}_N,\mathbf{y}}[\mathbf{C}|x] = E_{\mathbf{D}_N,\mathbf{y}}[(\mathbf{y} - h(x, \boldsymbol{\alpha}_N(D_N)))^2 \,|x]$$

The MISE of a nonlinear model

- If we integrate over the domain $\mathcal{X}$,

$$\text{MISE} = E_{\mathbf{x}}[\text{MSE}(x)] = \int_{\mathcal{X}} \text{MSE}(x) dF_{\mathbf{x}}(x)$$

- *Expected test error* or *expected prediction error*: expectation over everything that is random, i.e. the input, training set D_N (of given size N) and test set.
- MSE and the MISE can be computed analytically only if we know the distribution underlying the data.
- What can we do if this is not the case?

Generalization and overfitting

- Goal: **generalize**, i.e. able to return good predictions for input sets independent of the training set.
- Are models with null empirical risk good?
- Typically NOT: doing very well on the training set could mean doing badly on new data (**overfitting**).
- Using the same data for training a model and assessing it is a wrong procedure, since **over optimistic** assessment of the generalization capability.

Bias and variance of a model (I)

Parametric identification algorithm is an estimator $\hat{\boldsymbol{\theta}} = h(x, \boldsymbol{\alpha}_N)$ of the unknown quantity $\theta = f(x) = E[\mathbf{y}|x]$.

Like for all estimators $\hat{\boldsymbol{\theta}}$ we can define bias

$$\text{Bias}[h(x, \boldsymbol{\alpha}_N)] = E[h(x, \boldsymbol{\alpha}_N)] - f(x)$$

and variance

$$\text{Var}\left[\hat{\boldsymbol{\theta}}\right] = \text{Var}\left[h(x, \boldsymbol{\alpha}_N)\right]$$

Bias and variance of a model (II)

Since $\mathbf{y} = f(x) + \mathbf{w} = E_{\mathbf{y}}[\mathbf{y}|x] + \mathbf{w}$, $E[\mathbf{w}] = 0$ and $E[\mathbf{w}^2] = \sigma^2_{\mathbf{w}}$

$$
\begin{aligned}
\text{MSE}(x) &= E_{\mathbf{D}_N,\mathbf{y}}\left[\mathbf{C}(x,y)\right] = E_{\mathbf{D}_N,\mathbf{y}}\left[(\mathbf{y} - h(x,\boldsymbol{\alpha}_N))^2\right] = \\
&= E_{\mathbf{D}_N,\mathbf{y}}\left[(\mathbf{y} - E_{\mathbf{y}}[\mathbf{y}|x] + E_{\mathbf{y}}[\mathbf{y}|x] - h(x,\boldsymbol{\alpha}_N))^2\right] = \\
&= E_{\mathbf{D}_N,\mathbf{y}}\big[(\mathbf{y} - E_{\mathbf{y}}[\mathbf{y}|x])^2 + 2\mathbf{w}(E_{\mathbf{y}}[\mathbf{y}|x] - h(x,\boldsymbol{\alpha}_N)) + \\
&+ (E_{\mathbf{y}}[\mathbf{y}|x] - h(x,\boldsymbol{\alpha}_N))^2\big] = \\
&= E_{\mathbf{y}}\left[(\mathbf{y} - E_{\mathbf{y}}[\mathbf{y}|x])^2\right] + E_{\mathbf{D}_N}\left[(h(x,\boldsymbol{\alpha}_N) - E_{\mathbf{y}}[\mathbf{y}|x])^2\right] = \\
&= \sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}\left[(h(x,\boldsymbol{\alpha}_N) - E_{\mathbf{y}}[\mathbf{y}|x])^2\right] = \\
&= \sigma^2_{\mathbf{w}} + && \text{“noise”} \\
&+ (E_{\mathbf{D}_N}[h(x,\boldsymbol{\alpha}_N)] - E_{\mathbf{y}}[\mathbf{y}|x])^2 + && \text{“squared bias”} \\
&+ E_{\mathbf{D}_N}\left[(h(x,\boldsymbol{\alpha}_N) - E_{\mathbf{D}_N}[h(x,\boldsymbol{\alpha}_N)])^2\right] && \text{“variance”}
\end{aligned}
$$

Assumptions: quadratic cost, N fixed, f fixed, x fixed, $\boldsymbol{\alpha}_N$ random since function of $\mathbf{D}_N$.

The bias/variance trade-off

1. Noise variance: it cannot be avoided (lower bound on the error) no matter how well we estimate $f(x)$, unless $\sigma^2_{\mathbf{w}} = 0$.
2. Bias of estimator: difference in x between the average of the outputs of the hypothesis functions over the set of possible D_N and $f(x)$.
3. Variance of estimator: variability of the guessed $h(x, \alpha_N)$ as one varies over training sets of fixed dimension N. Measures how sensitive the algorithm is to changes in the data set, regardless to the target. It is null if the prediction is always the same whatever the training set.

The bias/variance trade-off (II)

- In qualitative terms

$$\text{MSE}(x) = \sigma^2_{\mathbf{w}} + \text{squared bias} + \text{variance}$$

where

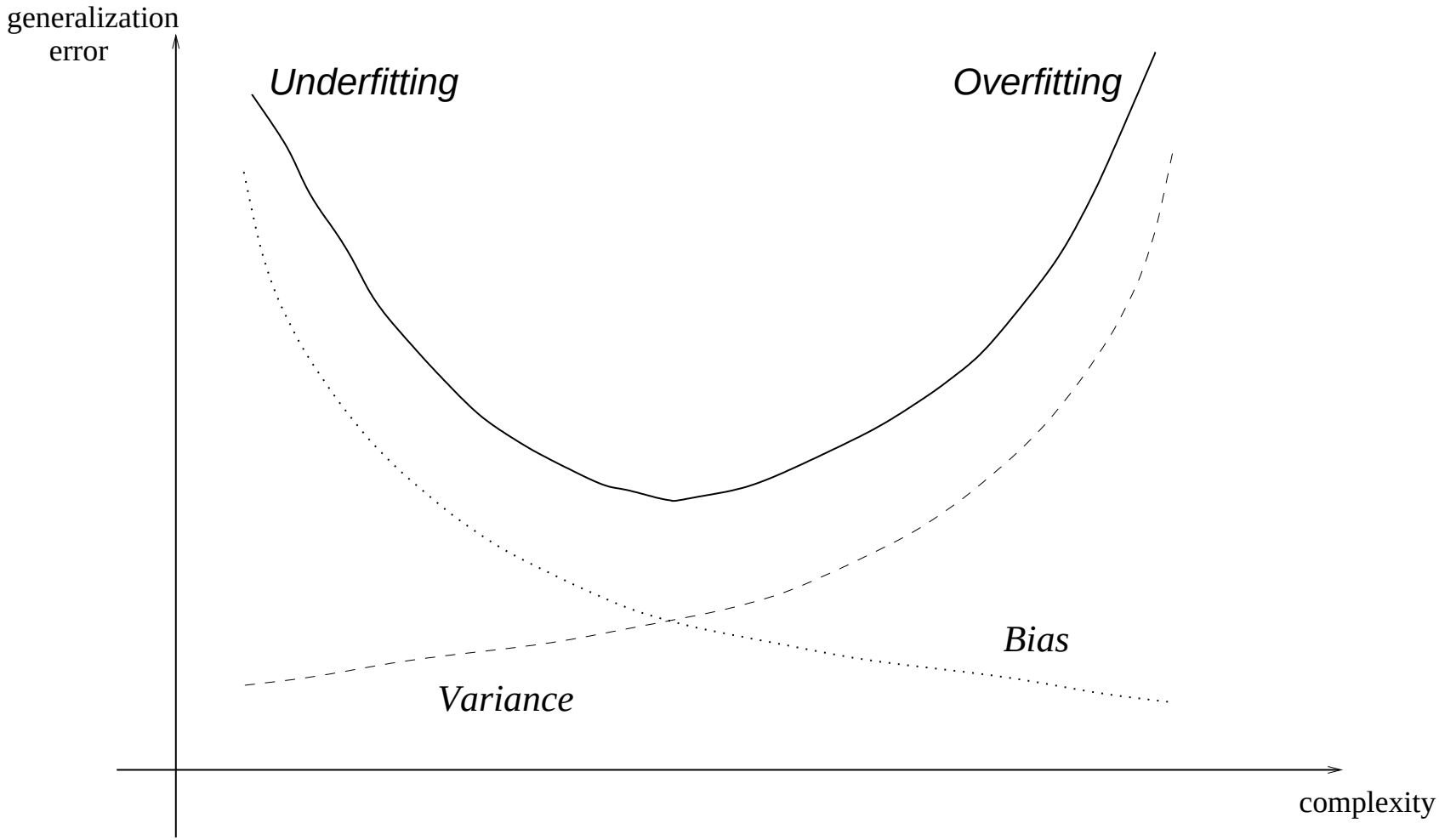
1. the intrinsic noise term reflects the target alone,
2. the bias reflects the target's relation with the learning algorithm and
3. the variance term reflects the learning algorithm alone.

The bias/variance dilemma

- No direct measure of MISE but estimate on the basis of the training set.
- Bias/variance decomposition is relevant in practical learning since it provides a useful hint about what to control to reduce the MISE error .
- Bias measures the lack of representational power of the class of hypotheses. To reduce the bias term we should consider complex hypotheses which can approximate a large number of input/output mappings.
- Variance warns us against an excessive complexity of the model: a class of too powerful hypotheses runs the risk of being excessively sensitive to the noise affecting the training set; therefore, our class could contain the target but it could be practically impossible to find it out on the basis of the available dataset.

- An hypothesis with large bias but low variance *underfits* the data while an hypothesis with low bias but large variance *overfits* the data.
- In both cases, the hypothesis gives a poor representation of the target and a reasonable trade-off needs to be found.
- The task of the model designer is to search for the optimal trade-off between the variance and the bias term, on the basis of the available training set.
- See script `StatLearn/biasvar_vis.py`.

Bias/variance trade-off



The learning procedure

A learning procedure aims at two main goals:

1. to choose a parametric family of hypothesis $h(x, \alpha)$ which contains or gives good approximation of the unknown function f (**structural identification**).
2. within the family $h(x, \alpha)$, to estimate on the basis of the training set D_N the parameter α_N which best approximates f (**parametric identification**).

In order to accomplish that, a learning procedure is made of two *nested loops*:

1. an external structural identification loop which goes through different model structures
2. an inner parametric identification loop which searches for the best parameter vector within the family structure.

Parametric identification

Parametric identification relies on ERM (Empirical Risk Minimization) principle

$$\alpha_N = \alpha(D_N) = \arg\min_{\alpha \in \Lambda} \widehat{\mathrm{MISE}}_{\mathrm{emp}}(\alpha)$$

which minimizes the **empirical risk or training error**

$$\widehat{\mathrm{MISE}}_{\mathrm{emp}}(\alpha) = \frac{\sum_{i=1}^{N} (y_i - h(x_i, \alpha))^2}{N}$$

constructed on the basis of the data set D_N.

Parametric identification (II)

- Requires a procedure of multivariate optimization in the space of parameters.
- Complexity of the optimization depends on the form of $h(\cdot)$.
- In some cases the parametric identification problem may be an NP-hard problem.
- Thus, we must resort to some form of heuristic search.
- Examples: linear least-squares for linear models and backpropagated gradient-descent for feedforward neural networks.

Automatic differentiation

- Derivatives are crucial to perform gradient-based non linear parametric identification.
- Traditionally, gradient-based approaches relied on manual derivation of analytical derivatives.
- Automatic differentiation is technique to numerically evaluate the derivative of a function given by computer program
- AD relies on incorporating derivative values in the variables of a computer program and redefining the operators to propagate derivatives as well by using the chain rule.
- Recent availability of libraries (e.g. Tensor Flow, Theano, PyTorch) providing automatic differentiation functionalities.

See Baydin et al. JMLR article *Automatic differentiation in ML: a survey*.

Validation techniques

- **Testing:** sample independent of D_N and distributed according to the same distribution is used to assess the quality. In practice, additional input/output observations are rarely available.
- **Holdout:** partitions the data D_N into two mutually exclusive subsets: training set D_{tr} and holdout (or test set) $D_{N_{ts}}$.
- **k-fold Cross-validation:** D_N is randomly divided into k mutually exclusive test partitions of approximately equal size. Cases not found in each test partition are used for training. Average error over all the k partitions is the cross-validated error rate.

The K-fold cross-validation

1. split the dataset D_N into k roughly equal-sized parts.
2. For the kth part $k = 1, \ldots, K$, fit the model to the other $K - 1$ parts of the data, and calculate the prediction error of the fitted model when predicting the k-th part of the data.
3. Do the above for $k = 1, \ldots, K$ and average the K estimates of prediction error.

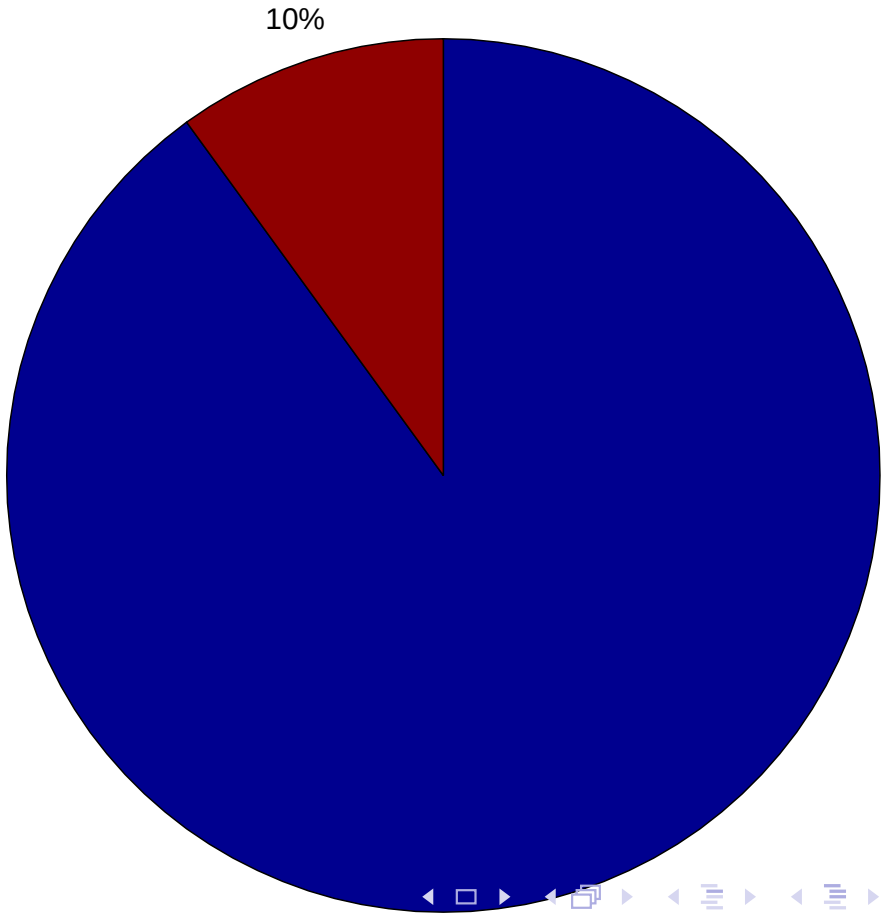
Let $k(i)$ be the part of D_N containing the ith sample.

$$\widehat{\text{MISE}}_{\text{cv}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i^{-k(i)})^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - h(x_i, \alpha^{-k(i)})\right)^2$$

where $\hat{y}_i^{-k(i)}$ denotes the fitted value for the ith observation returned by the model estimated with the $k(i)$th part of the data removed.

10-fold cross-validation

$K = 10$: at each iteration 90% of data are used for training and the remaining 10% for the test.



Leave-one-out cross validation

- Cross-validation algorithm with $K = N$ is **leave-one-out**.
- For each ith point, $i = 1, \ldots, N$,
  1. we carry out the parametric identification, leaving that observation out of the training set,
  2. we compute the predicted value for the ith observation, denoted by $\hat{y}_i^{-i}$

Estimate of MISE prediction error is

$$\widehat{\mathrm{MISE}}_{\mathrm{Loo}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i^{-i})^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - h\left(x_i, \alpha^{-i}\right)\right)^2$$

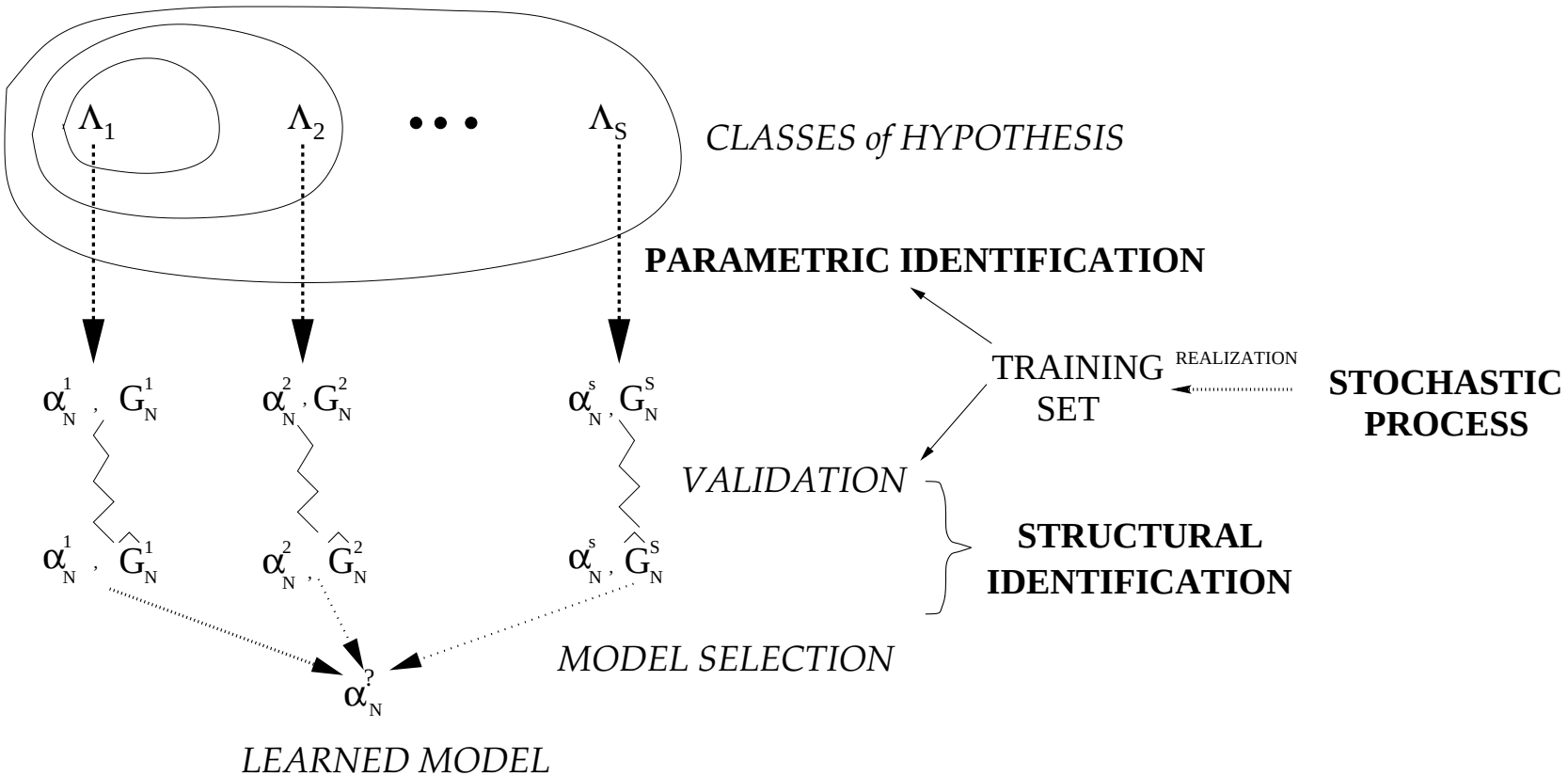
where α^{-i} are parameters returned by parametric identification on the training set with the ith sample set aside.

See script `StatLearn/loo_vis.py`.

Model selection

- Model selection is the final choice of the model structure in after model generation and validation.
- Choice may be subjective and dependent on multiple criteria, like quantitative measures, personal experience and computational effort.
- Here we will consider only quantitative criteria. Two possible approaches:
  1. the *winner-takes-all* approach
  2. the *combination of estimators* approach.

Model selection



Winner-takes-all

The best hypothesis is selected in the set $\{\alpha_N^s\}$, with $s = 1, \ldots, S$, according to

$$\tilde{s} = \arg \min_{s=1,\ldots,S} \widehat{\mathrm{MISE}}^s$$

A model with complexity $\tilde{s}$ is trained on the whole dataset D_N and used for future predictions.

Winner-takes-all pseudo-code

1. for $s = 1, \ldots, S$: (Structural loop)
   - for $j = 1, \ldots, N$
     1.1 Inner parametric identification (for l-o-o):

$$\alpha_{N-1}^{s} = \arg\min_{\alpha \in \Lambda_s} \sum_{i=1:N, i \neq j} (y_i - h(x_i, \alpha))^2$$

     1.2 $e_j = y_j - h(x_j, \alpha_{N-1}^{s})$
   - $\widehat{\mathrm{MISE}}_{\mathrm{Loo}}(s) = \frac{1}{N} \sum_{j=1}^{N} e_j^2$
2. Model selection: $\tilde{s} = \arg\min_{s=1,\ldots,S} \widehat{\mathrm{MISE}}_{\mathrm{Loo}}(s)$
3. Final parametric identification:
   $\alpha_N^{\tilde{s}} = \arg\min_{\alpha \in \Lambda_{\tilde{s}}} \sum_{i=1}^{N} (y_i - h(x_i, \alpha))^2$
4. The output prediction model is $h(\cdot, \alpha_N^{\tilde{s}})$

Winner-takes-all Python -code

```
for s in S:
    ## structural identification loop
    Eloo = []
    for j in range(N):
        ## leave-one-out loop
        Tr_i = np.delete(df, j, axis=0)
        Ts_i = df[j, :].reshape(1, -1)
        ## parametric identification
        h = MLPRegressor(hidden_layer_sizes=(s,))
        h.fit(Tr_i[:, 1:], Tr_i[:, 0])
        Eloo.append(Y[j] - h.predict(Ts_i[:, 1:].reshape(1, -1))
    MISEhat.append(np.mean(np.square(Eloo)))
stilde = S[np.argmin(MISEhat)]
h = MLPRegressor(hidden_layer_sizes=(stilde,))
h.fit(df[:, 1:], df[:, 0])
```

Excerpt from `StatLearn/wta.py`

Training, validation and test

- If we have enough samples, we recommend to randomly divide (e.g. 50%, 25%, 25%) the dataset into three parts: a training set, a validation set, and a test set.
- Training set is used to perform parametric identification
- Validation set is used to estimate prediction error for model selection
- Test set is used for unbiased assessment of the generalization error of the final chosen model.
- NB: the test set should be kept aside (and ideally forgotten) and considered only ONCE at the very end of the data analysis.
- Using the test-set (before the final assessment) would "contaminate" it and make it lose its unbiasedness.

Model combination

- Winner-takes-all approach is intuitively the best approach.
- Recent results show that the accuracy can be improved not by choosing which is expected to predict the best but by creating a model whose output is the **combination** of different models.
- Intuition: any chosen hypothesis $h(\cdot, \alpha_N)$ is only an estimate of the real target and, like any estimate, is affected by a bias and a variance term.
- See the theoretical results on the combination of estimators.

Combination of two estimators

Consider two unbiased estimators $\hat{\boldsymbol{\theta}}_1$ and $\hat{\boldsymbol{\theta}}_2$ of the same parameter θ

$$E[\hat{\boldsymbol{\theta}}_1] = \theta \qquad E[\hat{\boldsymbol{\theta}}_2] = \theta$$

having the same variance

$$\text{Var}\left[\hat{\boldsymbol{\theta}}_1\right] = \text{Var}\left[\hat{\boldsymbol{\theta}}_2\right] = v$$

and being uncorrelated, i.e. $\text{Cov}[\hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2] = 0$.

Combination of two estimators(II)

Let $\hat{\theta}_{\text{cm}}$ be the combined estimator

$$\hat{\theta}_{\text{cm}} = \frac{\hat{\theta}_1 + \hat{\theta}_2}{2}$$

This estimator has the nice properties of being unbiased

$$E[\hat{\theta}_{\text{cm}}] = \frac{E[\hat{\theta}_1] + E[\hat{\theta}_2]}{2} = \theta$$

and with a reduced variance

$$\text{Var}\left[\hat{\theta}_{\text{cm}}\right] = \frac{1}{4}\text{Var}\left[\hat{\theta}_1 + \hat{\theta}_2\right] = \frac{\text{Var}\left[\hat{\theta}_1\right] + \text{Var}\left[\hat{\theta}_2\right]}{4} = \frac{v}{2}$$

This trivial computation shows that the simple average of two unbiased estimators with a non zero variance returns a combined estimator with reduced variance.

Unstable learners

- A learning algorithm is informally called *unstable* if *small* changes in the training data lead to significantly different models and relatively *large* changes in accuracy.
- Unstable learners can have low bias but have typically high variance.
- Unstable methods can have their accuracy improved by *perturbing* (i.e. generating multiple versions of the predictor by perturbing the training set or learning method) and *combining*. Breiman call these techniques P&C methods.
- Combining multiple estimator is a variance reduction technique.
- The *bagging* technique is a P&C technique which aims to improve accuracy for unstable learners by averaging over such discontinuities.
- The philosophy of bagging is to improve the accuracy by reducing the variance.

Bagging

- Consider a dataset D_N and a learning procedure to build an hypothesis α_N from D_N.
- The idea of **bagging** or **bootstrap aggregating** is to imitate the stochastic process underlying the realization of D_N in order to reduce the variance of $h(\cdot, \boldsymbol{\alpha}_N)$.
- A set of B repeated bootstrap samples $D_N^{(b)}$, $b = 1, \ldots, B$ are taken from D_N.
- A model $\alpha_N^{(b)}$ is built for each $D_N^{(b)}$.
- A final predictor is built by aggregating the B models $\alpha_N^{(b)}$.
- In the regression case the bagging predictor is

$$h_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} h(x, \alpha_N^{(b)})$$

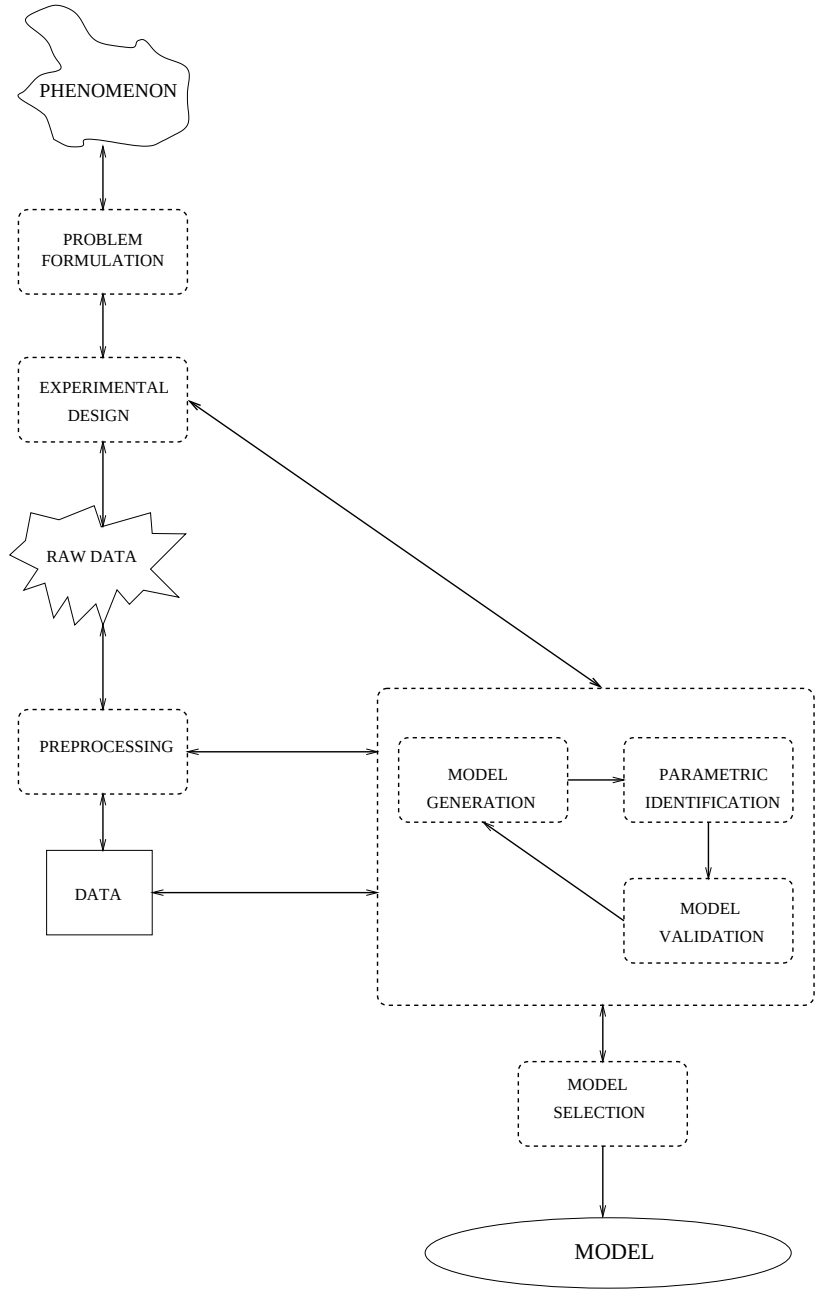
- In the classification case a majority vote is used.

Considerations

- Tests on real and simulated datasets showed that bagging can give a substantial gain in accuracy.
- The vital element is the instability of the prediction method.
- If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.
- On the other hand it can slightly degrade the performance of stable procedures.
- There is a cross-over point between instability and stability at which bagging stops improving.
- How many bootstrap replicates? In his experiments Breiman suggests that $B \approx 50$ is a reasonable figure.
- Bagging demands the repetition of B estimations of $h(\cdot, \alpha_N^{(b)})$ but avoids the use of expensive validation techniques (e.g. cross-validation).

Considerations (II)

- Bagging is an ideal procedure for parallel computing. Each estimation of $h(\cdot, \alpha_N^{(b)})$, $b = 1, \ldots, B$ can proceed independently of the others.
- Bagging is a relatively easy way to improve a existing method. It simply needs adding
  1. a loop that selects the bootstrap sample and sends it to the learning machine and
  2. a back end that does the aggregation.
- Note however that if the original learning machine has an interpretable structure (e.g. classification tree) this is lost for the sake of increased accuracy.


PHENOMENON
PROBLEM FORMULATION
EXPERIMENTAL DESIGN
RAW DATA
PREPROCESSING
DATA
MODEL GENERATION
PARAMETRIC IDENTIFICATION
MODEL VALIDATION
MODEL SELECTION
MODEL