CE 395 Special Topics in Machine Learning

Assoc. Prof. Dr. Yuriy Mishchenko

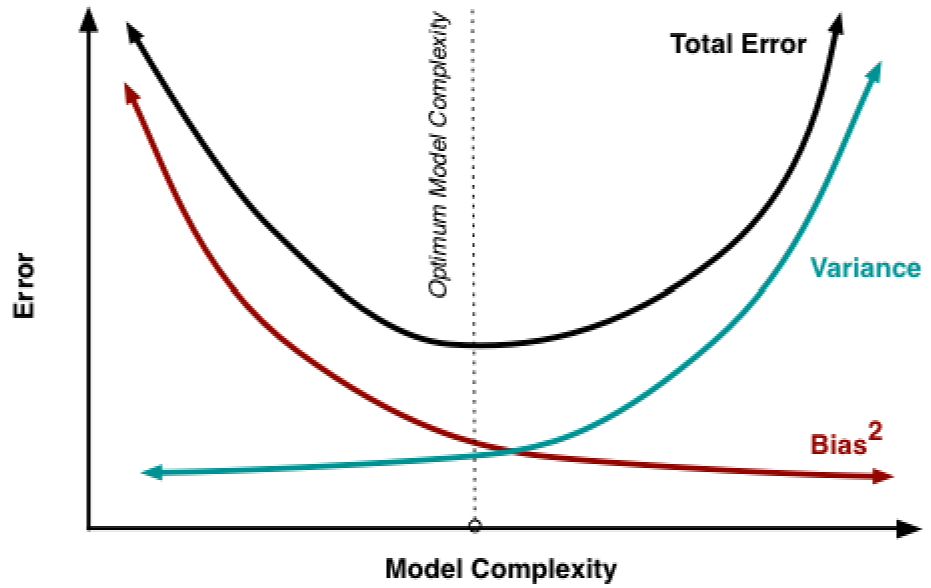
Fall 2017

STATISTICAL LEARNING

Bias-Variance Tradeoff and learning performance assessment

Question: How can we select best model complexity?

Fact: TPE is not an accurate predictor of EPE.



$$\begin{aligned} EPE_\lambda(x) &= E_T[(Y - \hat{f}_\lambda(x))^2 \mid X = x] \\ &= \sigma^2 + \mathrm{Bias}(\hat{f}_\lambda(x))^2 + \mathrm{var}(\hat{f}_\lambda(x)) \end{aligned}$$

Optimism bias

- **E**xpected **P**rediction **E**rror or generalization error – average over all inputs and outputs:

$$EPE = \sum_{(X,Y)} L(y, f_{\hat{\theta}}(x)) P(x, y) = E[L(y, f_{\hat{\theta}}(x))]$$

- **T**raining **P**rediction **E**rror – average error over fixed subset of inputs and outputs used for training:

$$TPE = \sum_{(x_i, y_i) \in T} L(y_i, f_{\hat{\theta}}(x_i)) = E[L(y, f_{\hat{\theta}}(x)) \mid T]$$

Optimism bias

TPE is always below than EPE:

$$TPE \quad < \ EPE$$

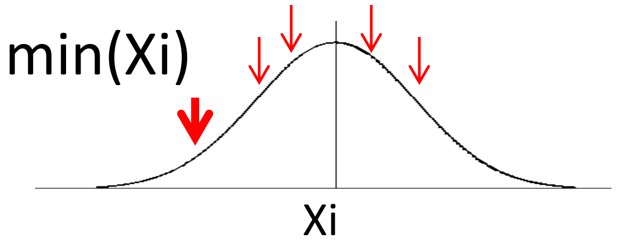
opt=EPE-TPE is called the optimism bias.

Optimism bias

Simple example:

E[max(Xi)]>E[Xi]

E[min(Xi)]<Ei[Xi]



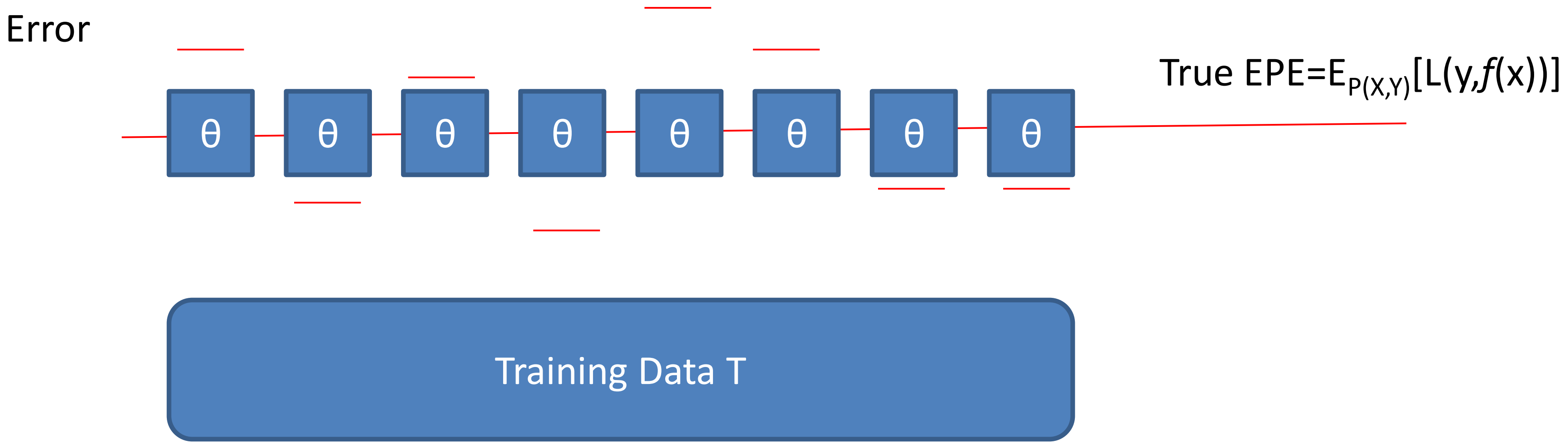
Example optimism bias

Let’s say we predict 2-side coin’s toss by randomly guessing with probability 50/5; the true EPE=50%:

- Now we make n=100 people do the same with N=25 coin tosses, and selected one person that has the highest correct score.
- It turns out that that person guessed about 75% of all the tosses correct! Has that person a super-human ability ???
- Actually, no. One can calculate using probabilities that the correct percentage of the best person in a set of 100 random experimenters, that is $\max_{100}$\{Guess-Accuracy\}, will be nearly 2.5 STD higher than the true percentage of 50% almost all of the time. (Similarly, the lowest score will be 2.5STD less than 50%, but we didn’t select for that!)
- For N=25 coins that evaluates to just about 50%+2.5*10% = 75%! So, that person was nothing special after all – she was just the lucky one that time, and we selected her because that’s what we did.
- Should we try to test that person on a new N=25 coin experiment, chances are **overwhelming** that she wouldn’t be able to guess more than 50% tosses by any meaningful margin.

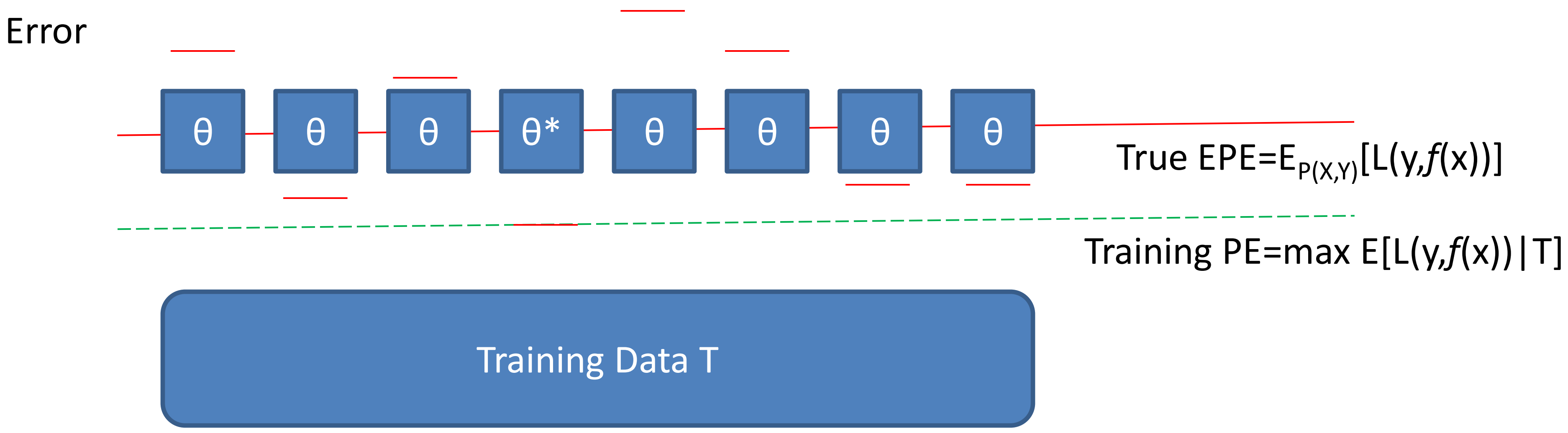
Learning performance assessment

In ML settings, we can view the accuracy of models f_θ as a random variable in relation to random realizations of the training dataset T:



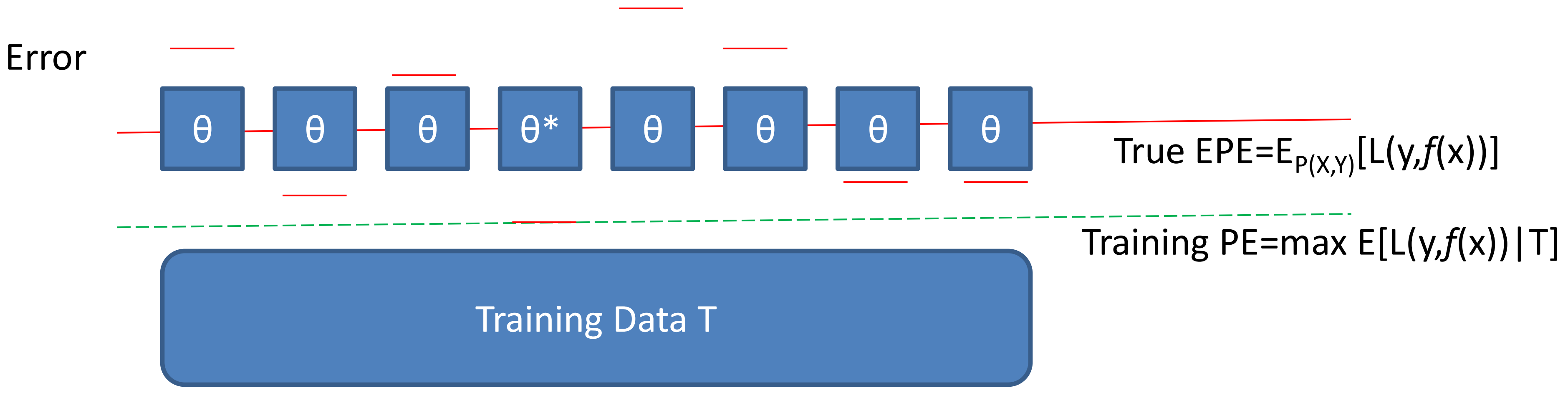
Learning performance assessment

Learning is formulated as maximizing the performance of the models on the training data, essentially – min over all training errors:


Error
θ
θ
θ
θ*
θ
θ
θ
θ
True EPE=$E_{P(X,Y)}[L(y,f(x))]$
Training PE=max $E[L(y,f(x))|T]$
Training Data T

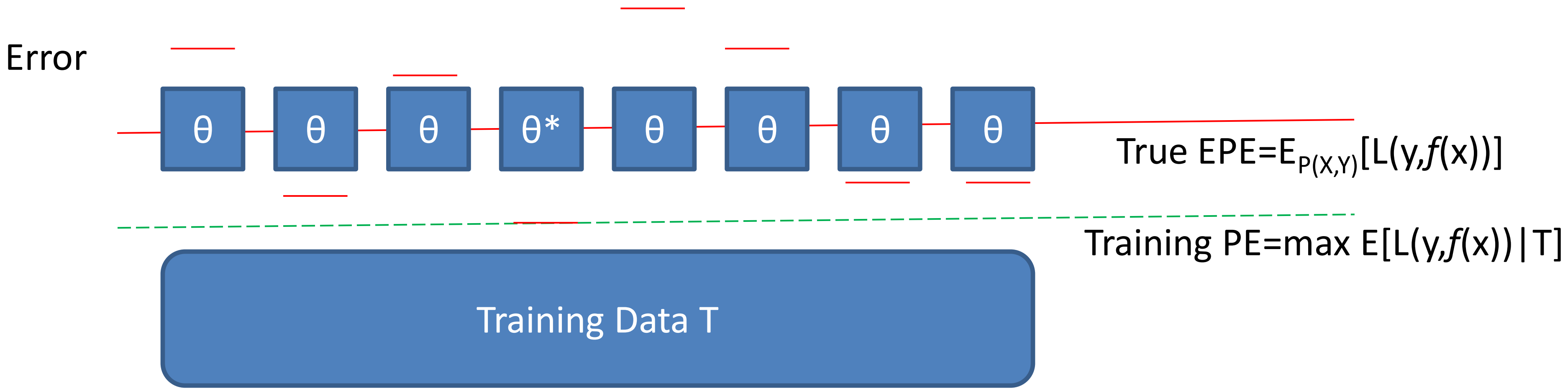
Learning performance assessment

The error of such a model on training data **TPE** is always **lower** than the **true EPE**


Error
θ
θ
θ
θ*
θ
θ
θ
θ
True EPE=$E_{P(X,Y)}[L(y,f(x))]$
Training PE=max $E[L(y,f(x))|T]$
Training Data T

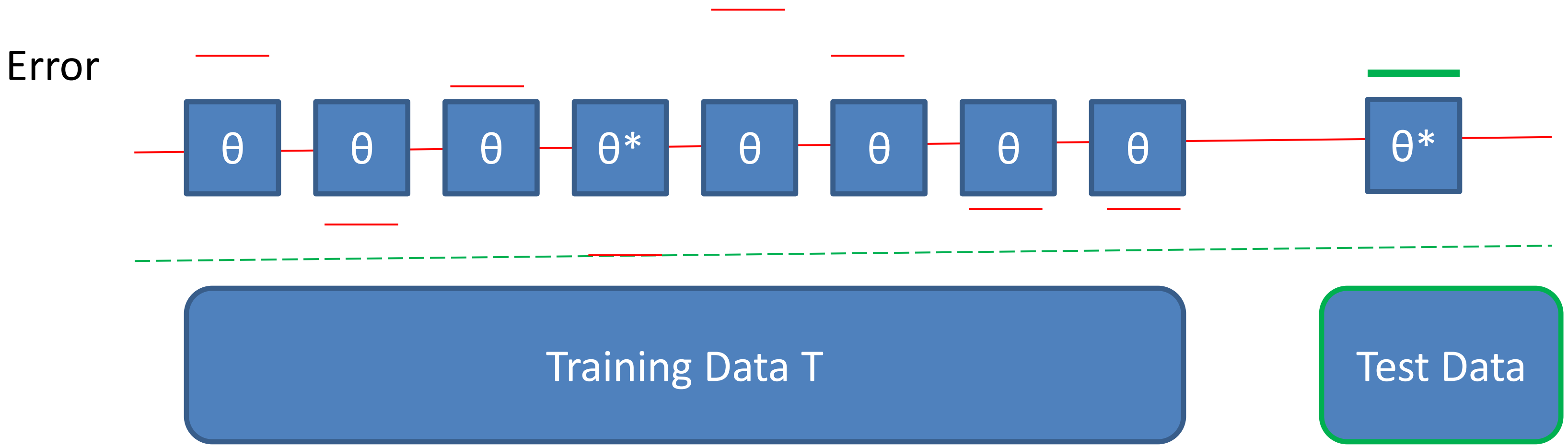
Learning performance assessment

Natural question: How to estimate the true EPE?


Error
θ
θ
θ
θ*
θ
θ
θ
θ
True EPE=$E_{P(X,Y)}[L(y,f(x))]$
Training PE=max $E[L(y,f(x))|T]$
Training Data T

Learning performance assessment

The answer is possible once one understands the **optimism bias**: by using data from P(X,Y) that is *new*, *original*, or *independent*



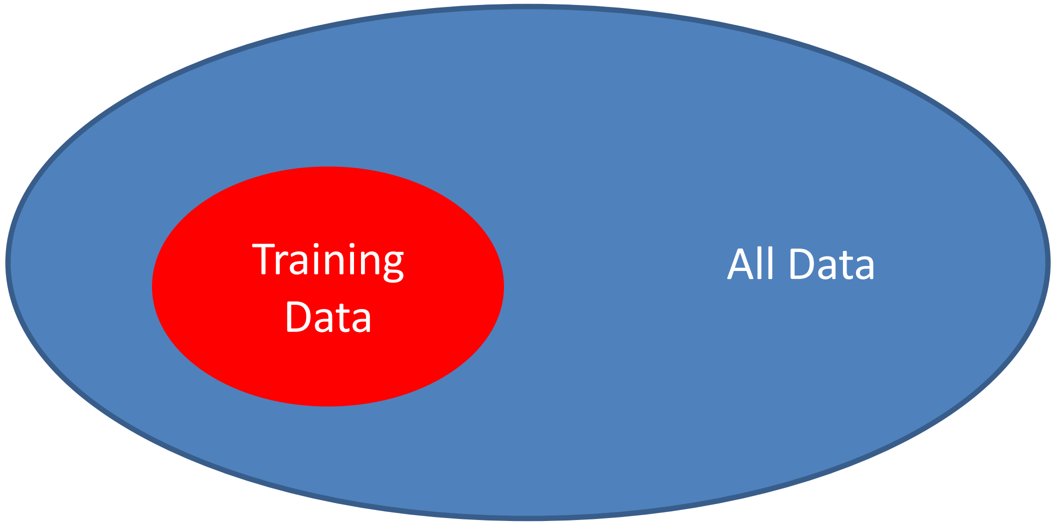
Learning performance assessment

The specific implementation of that idea is called **cross-validation**:

- Cross-validation consists in using independent or new datasets – called the test datasets – to estimate EPE of an already trained ML model.
- The need for cross-validation lies directly in the phenomenon of optimism bias, that is, the error of trained model evaluated on the training data is not an indicator of the model’s true prediction error.

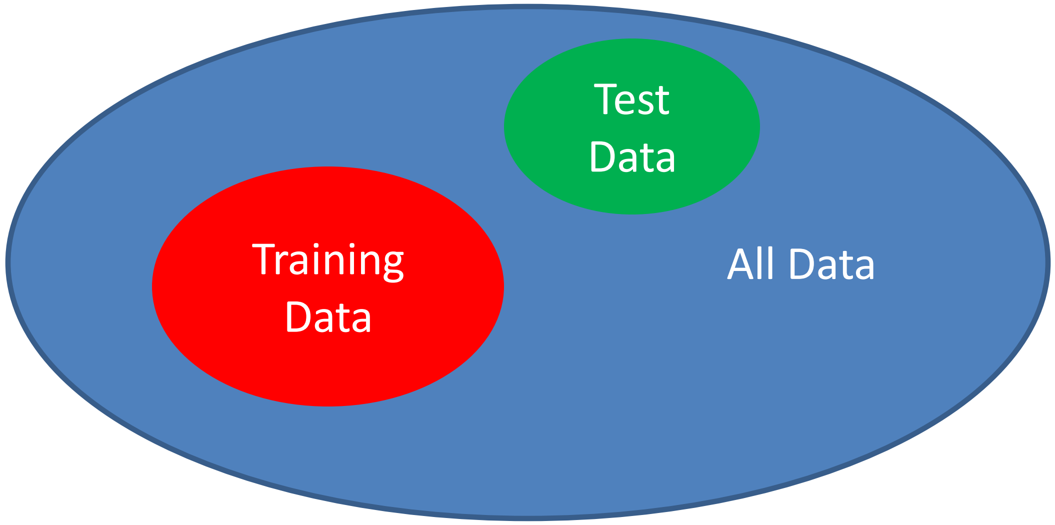
Learning performance assessment

Model optimized on a subset of data will naturally tend to show a better performance on that subset than on the rest of the data:



Learning performance assessment

The solution to that problem is to test the model on an independent sample of true data:



Learning performance assessment

In formulas:

$$T = \{(x_i, y_i) \sim P(X, Y)\}$$ Training set T

$$\theta^*(T) : \max \ E_{(x,y)\sim T}[L(y, f_\theta(x))]$$ Best model on T

Optimism bias in TPE

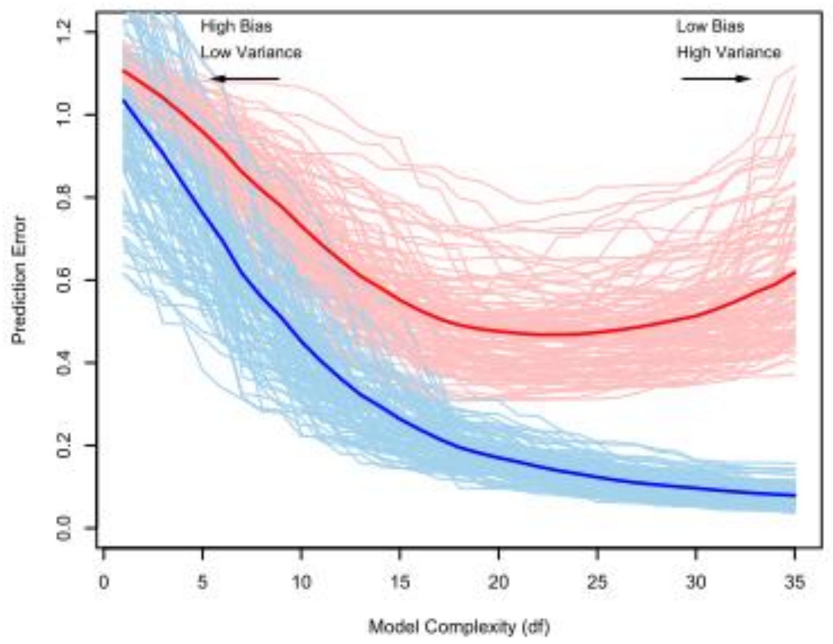
$$E_{(x,y)\sim T}[L(y, f_{\theta^*(T)}(x))] \leq E_{(x,y)\sim P(x,y)}[L(y, f_{\theta^*(T)}(x))]$$

Estimating EPE by using sample of data independent of θ*

$$EPE = E_{(x,y)\sim P(x,y)}[L(y, f_{\theta^*(T)}(x))] \approx \frac{1}{n_{test}} \sum_{(x_i, y_i)\in Test} L(y_i, f_{\theta^*}(x_i))$$

High-bias and high-variance conditions

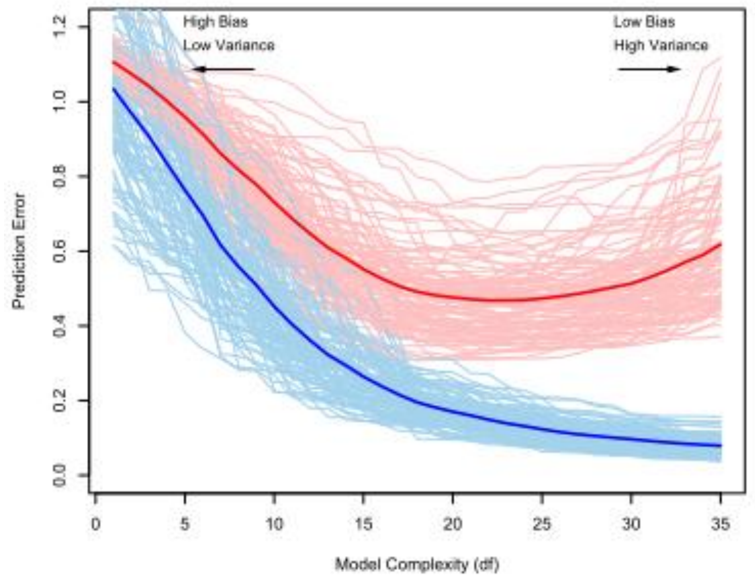
Degree of freedom or **dof** loosely quantifies how flexible a ML class of models can be in fitting data, that is how much such a model can fit random data and how much optimism bias therefore it will suffer. dof roughly equals the number of free parameters in a model.



High-bias and high-variance conditions

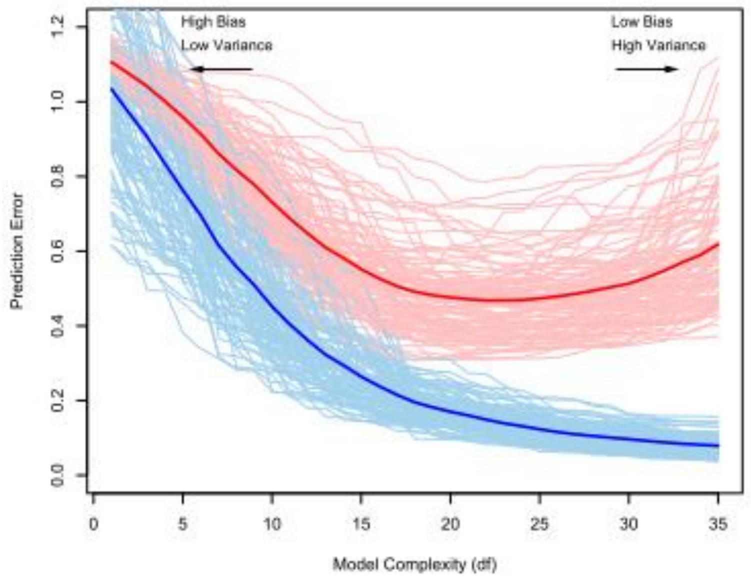
The learning situation to the left of that diagram is known as **high bias condition**

- High bias means that model with low dof (low complexity) is insufficient to capture real data – think about fitting linear model to nonlinear data; either training and test errors are high and close in that setting



High-bias and high-variance conditions

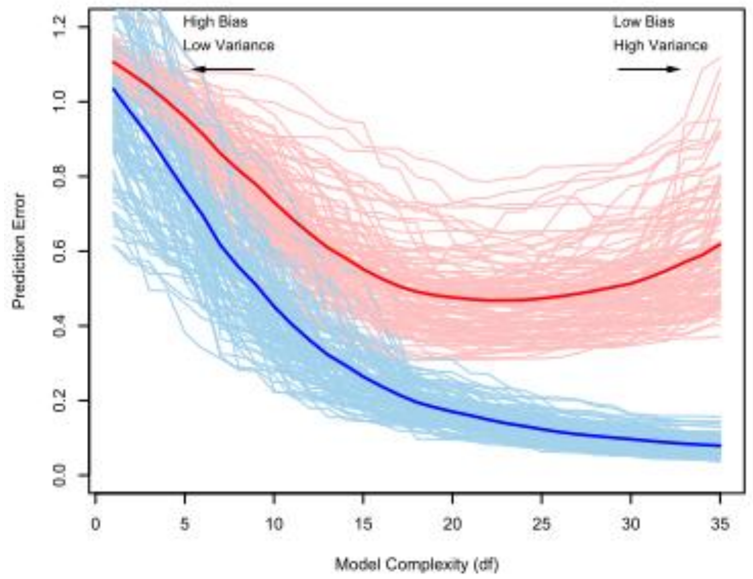
The method to improve in that situation is to increase the model’s complexity to such that is capable to capture the real data. For example, include new nonlinear terms into a linear basis expansion, decrease penalty parameter λ, etc.



High-bias and high-variance conditions

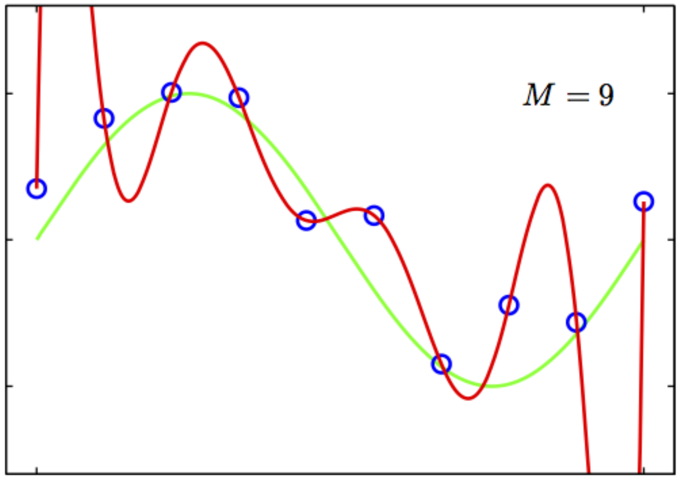
The learning situation to the left is known as **high variance condition**

- High variance means that the model is too flexible and fits the random features of training data rather than the true function *f*(x), thus training error is practically zero but test error is very high



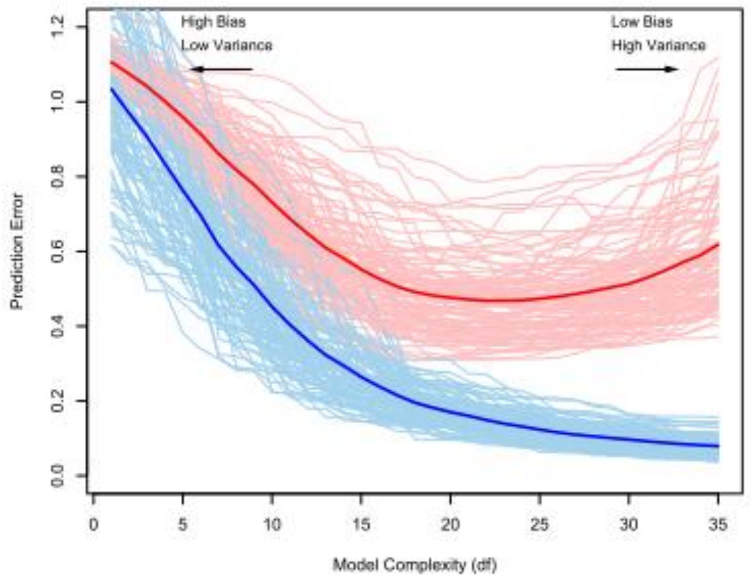
High-bias and high-variance conditions

Another name for the high-variance regime is **overfitting** – the model fits noise in the data rather than the relationship itself. Overfit models are said to **have bad generalization ability** or to **not generalize well**, meaning that they perform very badly whenever they see new data, not seen before. The models are said to **memorize the training data instead of learning**.



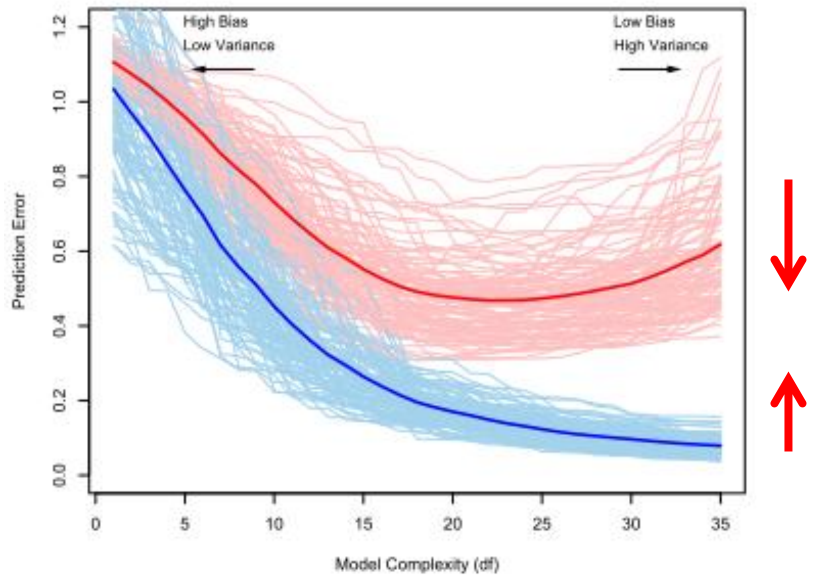
High-bias and high-variance conditions

The methods to improve in high-variance regime is to try models with lower complexity – for example by removing some of the terms in basis expansions or increasing the penalty λ. Another option is to increase the size of the training dataset, making E[L(x,*f*(x))|T] approximate closer EPE



High-bias and high-variance conditions

Indeed, presently industry-standard for training high-dof ML models (such as deep neural networks) is to overcome overfitting with data – as more data is given to model, the gap between training and test errors narrows.



Bias-variance tradeoff in KNN

We will use the simple KNN algorithm to illuminate the bias-variance tradeoff problem explicitly.

Bias-variance tradeoff in KNN

We assume additive error data model with zero-mean noise Eε=0 for data generation:

$$Y = f(X) + \varepsilon$$

and that a fixed set of inputs x_i is selected at the beginning, leaving all the variability in the training data T to purely randomness in outputs y_i, due to ε.

Bias-variance tradeoff in KNN

KNN predictions are going to be:

$$\hat{y}_K(x_0) = \frac{1}{K}\sum_{l=1}^{K} y_{(l)}$$

$$= \frac{1}{K}\sum_{l=1}^{K} f(x_{(l)}) + \frac{1}{K}\sum_{l=1}^{K} \varepsilon_{(l)}$$

$$\mathrm{E}[\hat{y}_K(x_0)] = \frac{1}{K}\sum_{l=1}^{K} f(x_{(l)}) + \frac{1}{K}\sum_{l=1}^{K} E[\varepsilon_{(l)}] = \frac{1}{K}\sum_{l=1}^{K} f(x_{(l)})$$

$$\mathrm{var}(\hat{y}_K(x_0)) = \mathrm{var}\left(\frac{1}{K}\sum_{l=1}^{K} \varepsilon_{(l)}\right) = \frac{\sum_{l=1}^{K} \mathrm{var}(\varepsilon_{(l)})}{K^2} = \frac{K\sigma_\varepsilon^2}{K^2} = \frac{\sigma_\varepsilon^2}{K}$$

Bias-variance tradeoff in KNN

EPE evaluated in that setting (as opposed to when x_i are also random) is the so called **in-sample error**:

$$
\begin{aligned}
EPE_K(x_0) &= E_T[(Y - \hat{f}_K(x_0))^2 \mid X = x_0] \\
&= \sigma^2 + [f(x_0) - \frac{1}{K}\sum_{l=1}^{K} f(x_{(l)})]^2 + \frac{\sigma^2}{K} \\
&= \sigma^2 + [\text{Bias}(\hat{f}_K(x_0))^2 + \text{Var}(\hat{f}_K(x_0))]
\end{aligned}
$$

Bias-variance tradeoff in KNN

Observe the 3 terms in the in-sample error:

- σ^2 - uncontrolled variation of y at fixed x_0 – **this is the irreducible error**
- Error in the average prediction due to particular choice of the training points x_i – **this is the bias**
- Error in the individual prediction due to random realization of outputs y_i at x_i – **this is the variance**

$$
\begin{aligned}
EPE_K(x_0) &= E_T[(Y - \hat{f}_K(x_0))^2 \mid X = x_0] \\
&= \sigma^2 + [\text{Bias}(\hat{f}_K(x_0))^2 + \text{var}(\hat{f}_K(x_0))] \\
&= \sigma^2 + [f(x_0) - \frac{1}{K}\sum_{l=1}^{K} f(x_{(l)})]^2 + \frac{\sigma^2}{K}
\end{aligned}
$$

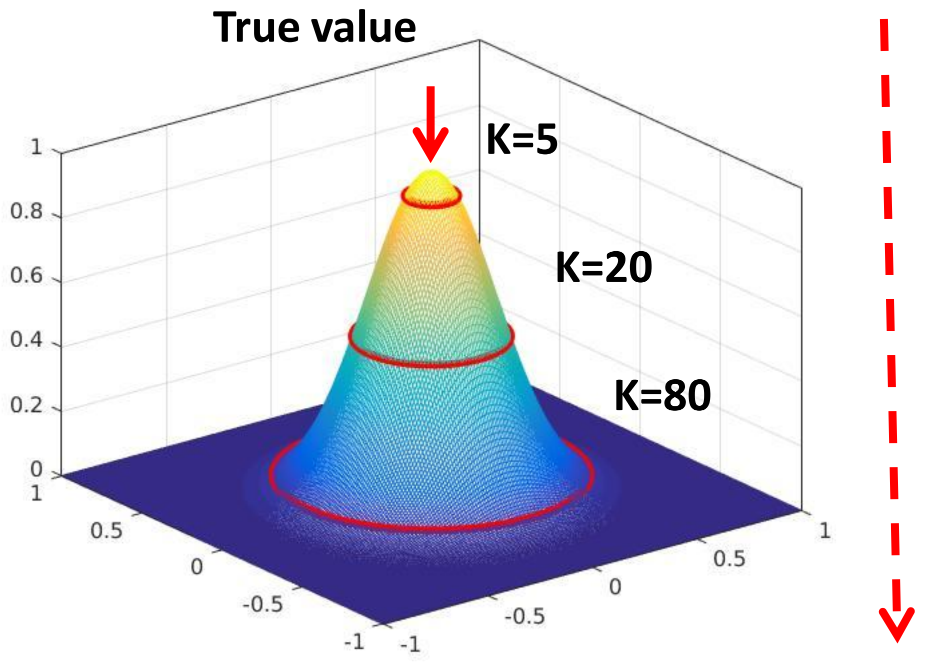
Bias-variance tradeoff in KNN

Only the last two terms are affected by K:

- The **variance** decreases ~1/K as greater number of points gets averaged to yield *f*^(x_0)
- The **Bias** increases as more *f*($x_{(l)}$) gets averaged due to each new *f*($x_{(l)}$) moving further away from *f*(x_0)

$$\begin{aligned} EPE_K(x_0) &= E_T[(Y - \hat{f}_K(x_0))^2 \mid X = x_0] \\ &= \sigma^2 + [\mathrm{Bias}(\hat{f}_K(x_0))^2 + \mathrm{var}(\hat{f}_K(x_0))] \\ &= \sigma^2 + [f(x_0) - \frac{1}{K}\sum_{l=1}^{K} f(x_{(l)})]^2 + \frac{\sigma^2}{K} \end{aligned}$$

Bias-variance tradeoff in KNN

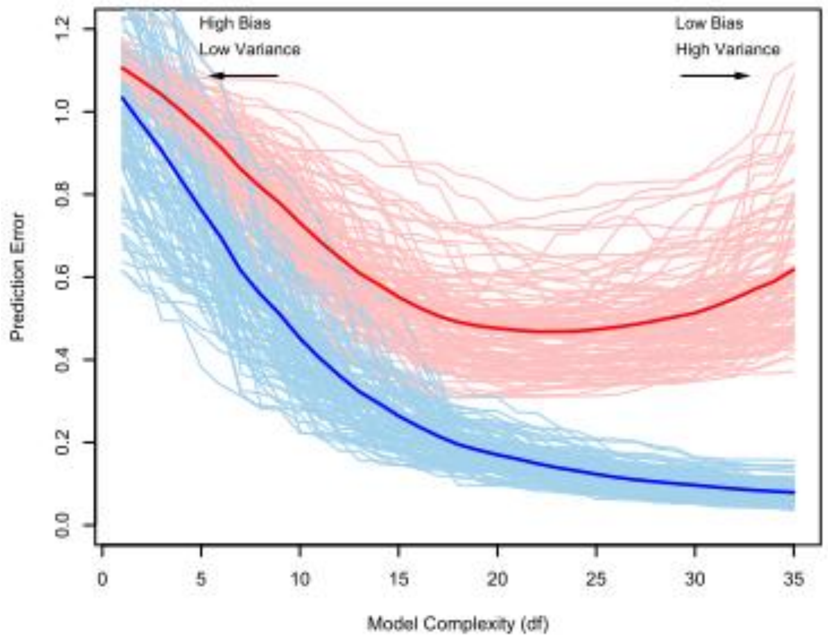


Statistical variation of averages decreases as more points get averaged over

The average's bias increases because included points move further away from x0 in the average

Bias-variance tradeoff in KNN

If we plot **the total error** vs **the complexity parameter here**, which for KNN is **dof=n/K –** the typical behavior of the generalization error is observed: as K increases, the error decreases thanks to gains in variance σ^2/K, but then EPE begins to rise again due to increasing deviations of the averages $f(x_{(l)})$ from $f(x_0)$



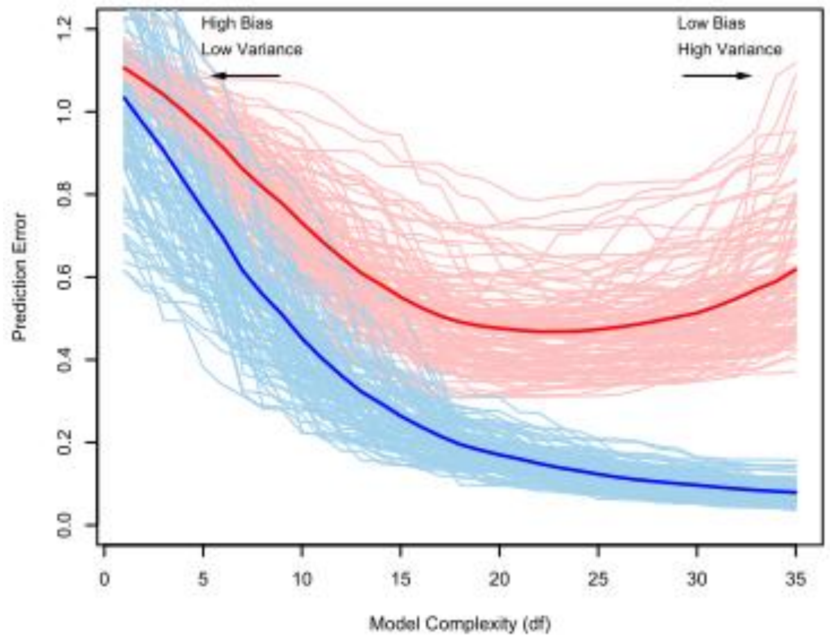
dof=n/K

Bias-variance tradeoff in KNN

The contributions of each term:

$$EPE_K(x_0) = E_T[(Y - \hat{f}_K(x_0))^2 \mid X = x_0]$$

$$= \sigma^2 + [f(x_0) - \frac{dof}{n}\sum_{l=1}^{K} f(x_{(l)})]^2 + \frac{\sigma^2}{n} dof$$

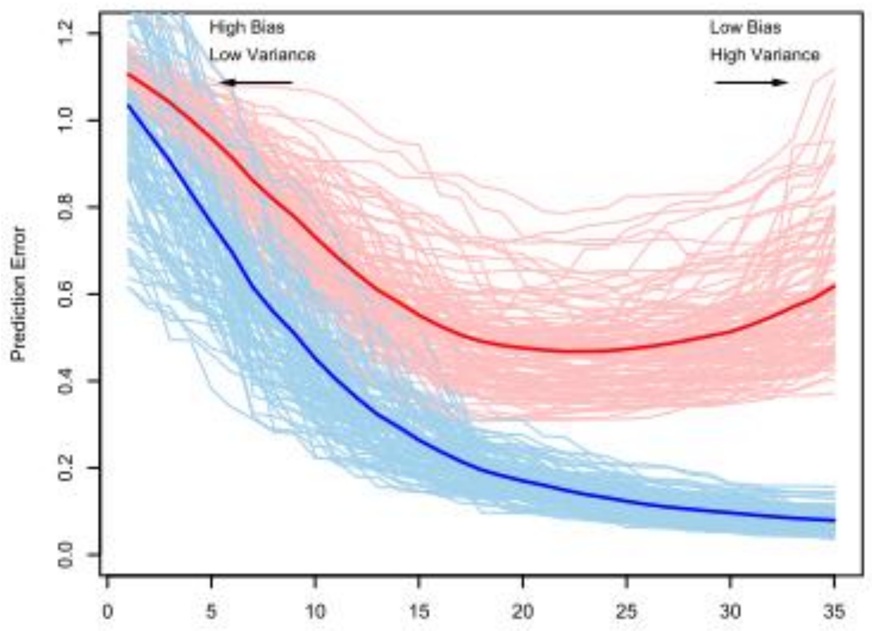


dof=n/K

Bias-variance tradeoff in KNN

In this example, we try to estimate $f(0)=1$ in $y=\exp(-8x^2)$, discussed in the last class, by using KNN with K=1..100, n=100 x_i, repeating the above for N=100 random runs:

- The red and blue lines are the errors on test and training datasets.
- Thick red line is the average test error and therefore an estimate of the true EPE.
- Thick blue line is the average test error E[err], affected by the optimism bias.
- Optimism bias is linearly increasing with dof here.



dof=n/K

Bias-variance tradeoff in linear model

2nd example – linear model

$$\{x_i, y_i\} \rightarrow \hat{y} = x^T \theta_p$$

assume generative additive error linear model with p parameters θ

$$\{x_i, y_i\} \leftarrow y = x^T \theta_p + \varepsilon$$

Bias-variance tradeoff in linear model

We know the solution for linear model:

$$\min RSS(\theta) = \min |y - \hat{y}|^2$$

explicitly:

$$\hat{\theta} = (X^T X)^{-1} X^T y,$$

$$\hat{y} = X\hat{\theta} = X(X^T X)^{-1} X^T y = Hy$$

Bias-variance tradeoff in linear model

We can use that solution to find theoretically EPE & TPE:

$$TPE = \frac{1}{n} | y - \hat{y} |^2 = \frac{1}{n} | (I - H) y |^2 = (1 - \frac{p}{n}) \sigma^2$$

$$EPE = E | y(x) - \hat{y}(x) |^2$$

$$= \sigma_\varepsilon^2 + E_x \left[f(x) - x^T \theta^* \right]^2 + \frac{p}{n} \sigma_\varepsilon^2$$

n is the number of data points, p is the number of weights θ, σ_ε^2 is the variance of noise ε

Bias-variance tradeoff in linear model

In linear model:

- Training error **TPE=$(1-p/n)\sigma_\varepsilon^2$** is always **smaller** than the noise variance σ_ε^2. In fact, if **$p=n$, TPE=0**, which means that linear model with $n=p$ parameters exactly fits every point in the data (quite naturally so since the number of equations in $y_i=x_i^T\theta$ then equals the number of unknown θ, so all equations can be fit perfectly).
- Nonetheless, true error **EPE=$(1+p/n)\sigma_\varepsilon^2$** is always **greater** than σ_ε^2! If **$p=n$, EPE=2 σ_ε^2**, meaning that generalization error is worst for the largest (and on training data the best) linear model !!!

Bias-variance tradeoff in linear model

- The difference EPE-TPE is the optimism bias, in linear model we can obtain it theoretically:

$$opt = EPE - TPE = 2\frac{dof}{n}\sigma_{\varepsilon}^{2}$$

- Increasing dof leads invariably to decrease of the training error but increase of the true prediction error (on all data)

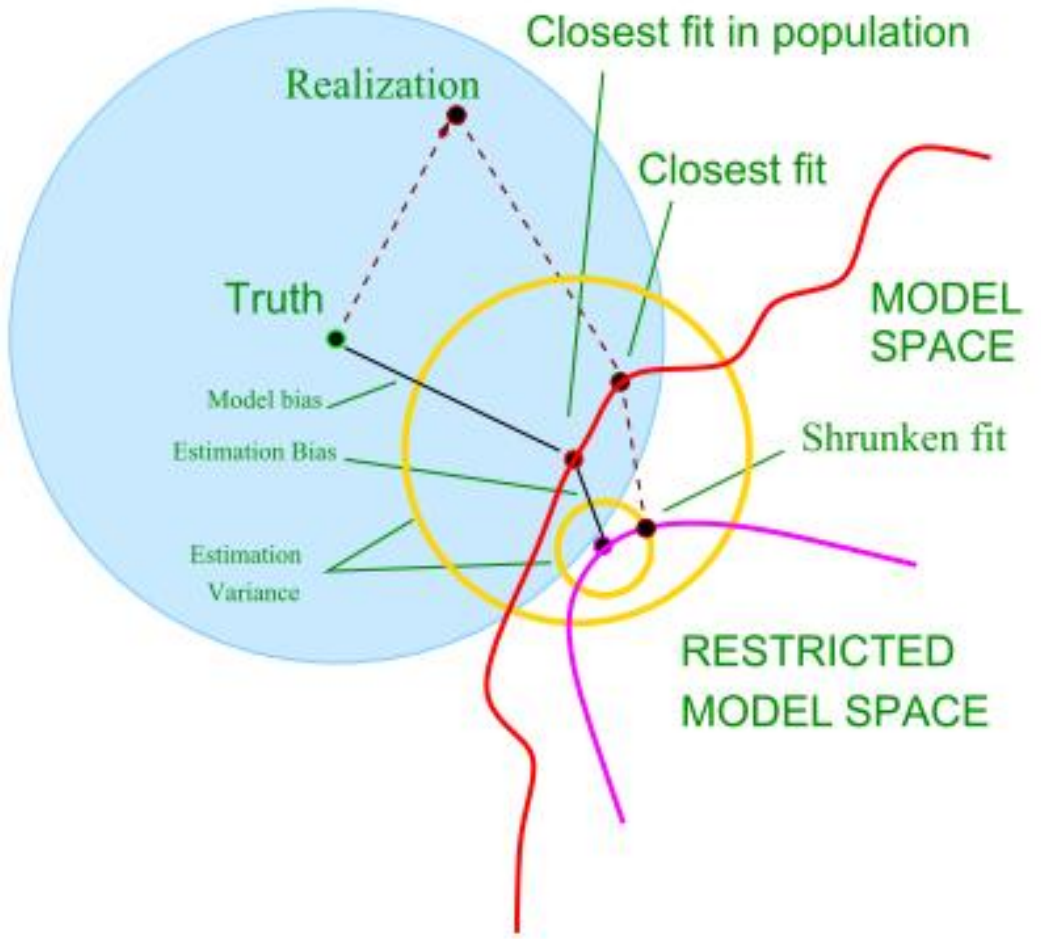
Learning performance assessment and bias-variance tradeoff

Review:

$$
\begin{aligned}
EPE(x_0) &= E[(Y - \hat{f}(x_0))^2 \mid X = x_0] \\
&= \sigma_\varepsilon^2 + (f(x_0) - E\hat{f}(x_0))^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\
&= \sigma_\varepsilon^2 + \text{Bias}(\hat{f}(x_0))^2 + \text{Var}(\hat{f}(x_0)) \\
&= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}
\end{aligned}
$$

This breakdown is the **Bias-Variance Decomposition**. Generally, as the **dof** of a model increases, **Bias** decreases (i.e. *f*^ is more flexible) but **Variance** increases (i.e. *f*^ is more flexible), this results in initially improving then worsening **EPE** in relation to **TPE**.

Schematic of the behavior of bias and variance in ML methods



Cross-validation methods

Important definitions for further discussion:

- **Loss function** (square or absolute error are popular options)

$$L(Y, f(X)) = \begin{cases} (Y - f(X))^2 \\ |Y - f(X)| \end{cases}$$

- **Expected Prediction Error** or **Generalization Error**

$$EPE = E_{P(X,Y)}[L(Y, f(X))]$$

Cross-validation methods

- **Test Error**

$$Err = \frac{1}{m}\sum_{i=1}^{m} L(y_i^{(test)}, f(x_i^{(test)}))$$

Note

$$EPE = E[Err]$$

- **Training Error**

$$\overline{err} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))$$

Cross-validation methods

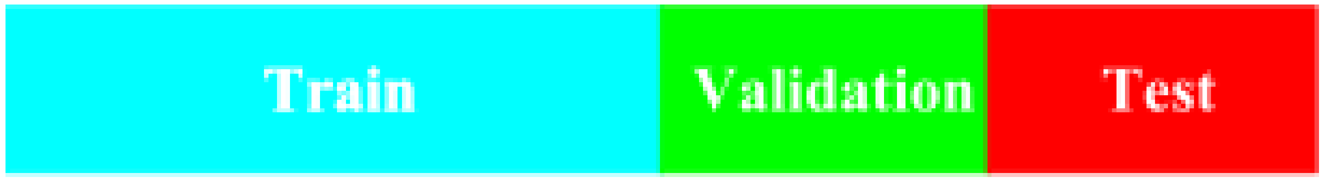
Two separate goals:

- **Model selection** – estimate performance of different models on data in order to choose the best one
- **Model assessment** – having a model, estimate its prediction error on data

Cross-validation methods

How to satisfy both goals:

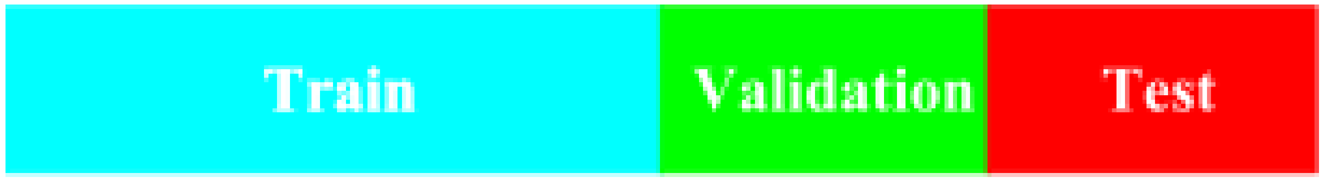
- Divide dataset into training and test parts – (a popular split is **training**, **validation**, and **test sets** as 50%-25%-25%)



Cross-validation methods

- **Training set** is used to fit and select models
- **Validation set** is used to estimate the true prediction error and to choose complexity, such as by choosing K for KNN or p for liner model
- **Test set** is used to estimate the final model's generalization error or true Err or EPE

Ideally, all these sets must be independent from each other – no overlapping points !



Cross-validation methods

The cross-validation scheme motivated statistically and primarily discussed above is called **hold-out cross validation.**

In hold-out cross-validation, a certain part of dataset is put aside and used only as an “independent” sample of data to test final models’ Err. Held-out data is never used in training of models, it should be as close to pristine “new” data as possible.

Cross-validation methods

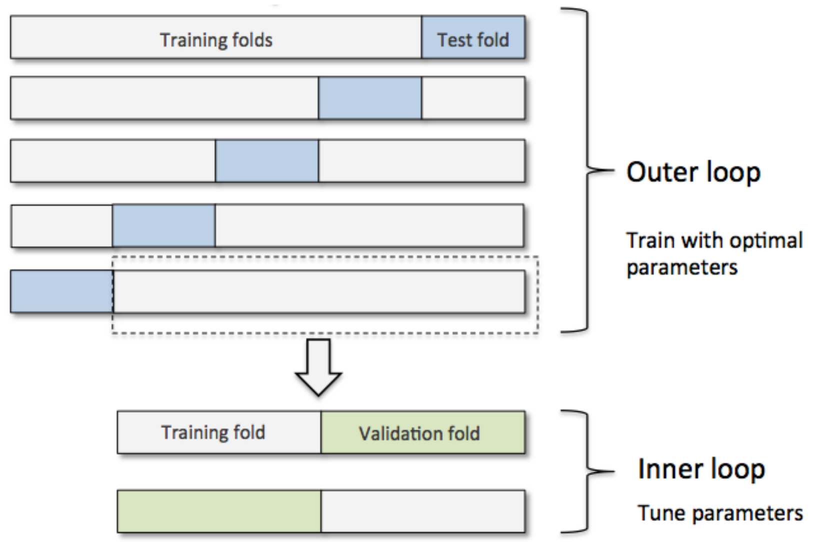
While hold-out cross-validation is statistically sound, it has several problems:

- A certain set of data is reserved and never used for training: if dataset is small to begin with, this sounds like a waste of possibly critical data
- EPE can be estimated only once – on the data segment held out; we can have no idea how variable or how far that estimate is from the real EPE=E[Err] (single value of r.v. to judge the mean!)

Cross-validation methods

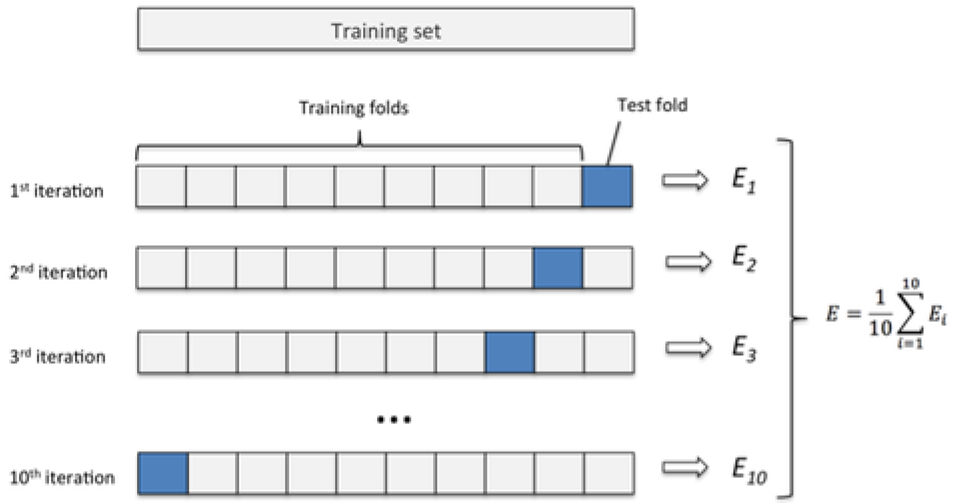
Alternative to hold-out cross-validation is **K-fold cross-validation.**

In K-fold cross-validation, the data is split into K parts, and then sequentially for K rounds one part of the data is held out while the remaining K-1 parts of data are used to train models. The held-out data is used to estimate Err in K-different ways.



Cross-validation methods

K-fold cross validation gives both the estimate of EPE as average test Err over K-folds, as well as an idea of how variable such estimate is – by using the variance of the folds' test Err – and how far from the truth our estimate of EPE can be.



Cross-validation methods

The drawbacks of K-fold cross validation are as follows:

- Model needs to be retrained K-times for each fold; many ML models are very computationally intense; unless fold-cross-validation can be done analytically this may prove unbelievably expensive
- Training and tests in different folds are not fully independent from each other, allowing for optimism bias to remain in EPE estimates

Cross-validation methods

The extreme form of K-fold cross validation where a single data point is held out is **leave-one-out cross validation**

- Leave-one-out cross-validation is folds cross-validation with K=n, n is the number of all data points available
- For each round, one point is held out while n-1 points are used to train model
- Generalization performance is estimated by applying the trained model to the one point left out:

$$Err \approx \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}^{-i}(x_i))$$

Cross-validation methods

- Obviously, due to the need to train n models in leave-one-out cross validation, this type of cross-validation can only be done if the errors can be estimated in great part analytically
- This is the case, for example, with the linear model, where one can calculate the leave-one-out EPE analytically from the results of single fit:

$$Err \approx \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{1 + H_{ii}}, \quad H \; is \; \hat{y} = Hy$$

Model complexity criteria

Let’s have a second look at the optimism bias and how it can be estimated to define what is called the “complexity criteria”, which can be used to make educated guess at the best values of the model complexity parameters such as K or p, or regularizing penalties λ.

Model complexity criteria

First of all recall the analytical optimism bias formula we obtained for linear model

$$opt = Err - \overline{err} = 2\frac{p}{n}\sigma_{\varepsilon}^{2}$$

Another way to record the same relationship, considering that y^=Hy, in linear model, where H is the hat matrix, is

$$opt = Err - \overline{err} = \frac{2}{n}\sum_{i=1}^{n}\operatorname{cov}(\hat{y}_i, y_i)$$

where the sum is of the covariances of estimates y_i^ as those vary in response to random variation in training data y_i.

Model complexity criteria

This, in fact, is a very general relationship applicable in much wider contexts (see Elements of Statistical Learning Hastie, Tibshirani, Friedman and Advanced) and called in-sample error estimate formula:

$$Err = \overline{err} + \frac{2}{n}\sum_{i=1}^{n} \mathrm{cov}(\hat{y}_i, y_i)$$

Model complexity criteria

The quantity $\sum$cov(y^,y) is thus called the **degrees of freedom** of a ML algorithm, so that

$$\sum_{i=1}^{n} \operatorname{cov}(\hat{y}_i, y_i) = dof \cdot \sigma_{\varepsilon}^{2}$$

and equivalently

$$dof = \frac{\sum_{i=1}^{n} \operatorname{cov}(\hat{y}_i, y_i)}{\sigma_{\varepsilon}^{2}}$$

Model complexity criteria

For linear model, this gives exactly p:

$$dof = \frac{\sum_{i=1}^{n} \operatorname{cov}(\hat{y}_i, y_i)}{\sigma_\varepsilon^2} = p \qquad (*)$$

For other methods, d≠p and (*) defines the **effective degree of freedom** of ML method

Model complexity criteria

For example, for KNN algorithm with n training points and K neighbor-local averages, one can calculate the e*ffective* dof explicitly

$$dof_{KNN} = \frac{\sum_{i=1}^{n} \mathrm{cov}(\hat{y}_i, y_i)}{\sigma_{\varepsilon}^{2}} = \frac{n}{K}$$

Model complexity criteria

Using effective dof, the optimism bias becomes

$$Err = \overline{err} + 2\frac{dof}{n}\sigma_{\varepsilon}^{2}$$

where “dof” is the number of degrees of freedom of ML method. This relationship is known as **Akaike Information Criterion** or **AIC**.

Akaike Information Criterion

- AIC is an estimate of a ML method's real prediction error with the account of the optimism bias

$$AIC = \overline{err}(\theta) + 2\frac{dof}{n}\hat{\sigma}_{\varepsilon}^{2}$$

- AIC can be used to decide on the number of effective dof needed in a ML method to provide the best *generalization* performance, which is what we care about, corresponding to minimizing AIC(dof).

Akaike Information Criterion

- AIC is one of criteria for selecting models in ML
  - Model is described via its effective dof; typically dof is simply taken to equal to the total number of free parameters in the model, dof=p, although it can be lower than that too, but not higher.
  - In relation to that, AIC estimates the amount of the optimism bias suffered by such model’s TPE and therefore the true EPE, with the optimism bias.
  - One is offered a method for choosing dof that can achieve best **predictive performance** of a ML model.

Bayesian Information Criterion

An approach alternative to AIC is the so called **Bayesian Information Criterion** or **BIC**. BIC approaches the problem of selecting a model from Bayesian inference perspective (see Advanced in probability).

Bayesian Information Criterion

BIC evaluates likelihood of a class of data models f_λ marginal over the p parameters $\theta_{1..p}$

$$\log P(X \mid p) = \log \int d\theta_p P(X \mid \theta_p, p) P(\theta_p \mid p)$$

Bayesian Information Criterion

Whereas this integral can be approximately evaluated in relation to the dimensionality p, one obtains

$$\log P(X \mid p) = \log \int d\theta_p P(X \mid \theta_p, p) P(\theta_p \mid p)$$

$$= \text{loglik} - \frac{p}{2} \log n = -\frac{BIC}{2}$$

where BIC is defined as

$$BIC = -2\,\text{loglik} + (\log n) \cdot p$$

VC dimensions

Vapnik-Chervonenkis (VC) dimension is another complexity measure used for function estimators in the Vapnik’s theory

VC dimensions

Vapnik's formula (d is the VC dimension of model here):

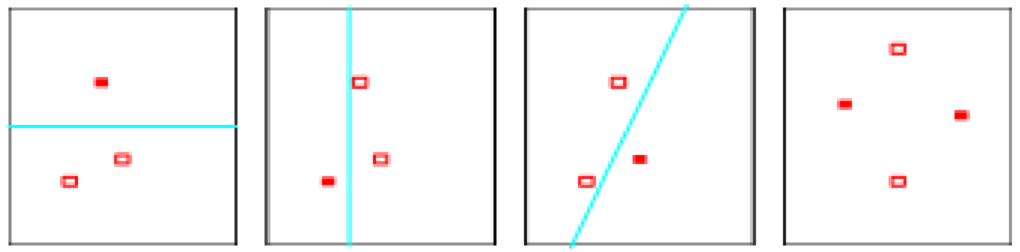
$$\Pr\left\{ Err \leq \overline{err} + \eta_{\delta} \right\} \geq 1 - \delta$$

$$\eta_{\delta} = \sqrt{\frac{d\,[\log(2n/d) + 1] + \log(4/\delta)}{n}}$$

Vapnik's formula states that the probability to observe a deviation of the *Test error* from the T*raining error* at most η_{δ} is bound by δ, defined by the VC dimension of that estimator

VC dimensions

- **VC dimension of a class of models f_θ(X) is defined as the largest number of points in X that can be split in all possible ways by the members of f_θ(X)**
- For example, VC of linear classifier in 2D is 3 (more generally VC of any family of linear classifiers in p dimensions is p+1)



VC dimensions

- Unlike AIC and BIC, which attempt to provide in a sense *an average estimate* of Test error in relation to Train error, VC dimension is a solid probabilistic bound on the difference of the two, which therefore is the main advantage of VC theory
- Compare :
  - “Optimism bias is roughly $2/n \cdot d \cdot \sigma^2$” (in the sense E[opt] as $n \rightarrow \infty$) (AIC, BIC)
  - “Optimism bias is at most η_δ with probability δ” (VC)

VC dimensions

- The main problem of Vapnik's theory is difficulty of computing VC dimensions of practical model classes (essentially impossible)
- Moreover, although VC bound is superior from theoretical viewpoint, it has to be very loose to accommodate all worst-case scenarios
  - (if you were observant enough, you had noticed that while AIC and BIC involve estimate of data's σ^2 in the optimism bias estimate, VC does not do that, and that is at the same time the strength and the weakness of VC method)

VC dimensions

Some VC dimensions for reference:

- Linear classifier/plane in p dimensions, VC=p+1
- Boosting (stagewise linear) classifier with T weak-learners each of VC dimension D – VC=T(D+1)(3log[T(D+1)] +2)
- Neural network with V neurons, E connections, and sigmoid activation functions – $c_1E^2 \leq VC \leq c_2E^2V^2$

Model complexity criteria

In practice:

- **AIC** is known to be a weaker criterion allowing typically for too large models (underestimates optimism bias)
- **BIC** is known to be stronger, resulting in models that are often too small (overestimates optimism bias)
- **VC** is rarely used in practice given that for most practical ML methods VC-dimensions are not known and the bounds self are very loose

Complexity selection using cross-validation

- Currently recommended approach for models is **to use cross-validation**, which is the direct estimate of models' prediction error
- In cross validation, the test error is directly an estimate of Err and moreover theoretically we know that Err=E[Err$_{Test}$]

$$Err = E[Err_{Test}] = E[\frac{1}{n_{test}}\sum_{i=1}^{n_{test}}(y_i - f(x_i))^2]$$

Complexity selection using cross-validation

If possible, the generalization error estimate using K-fold cross-validation should be used ($f^{-k(i)}$ is the fitted function without the fold k):

$$Err \approx CV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}^{-k(i)}(x_i))$$

Complexity selection using cross-validation

Similarly, cross-validation estimate of prediction error using leave-one-out cross-validation can be used if possible (f^{-i} is the fitted function computed without data point i)

$$Err \approx CV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}^{-i}(x_i))$$

Complexity selection using cross-validation

For a special case when for some matrix S

$$\hat{y} = Sy$$

Leave-one-out cross-validation can be done analytically from results of single fit:

$$Err \approx CV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{f}^{-i}(x_i)]^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2$$

Complexity selection using cross-validation

Generalized cross-validation uses that formula to approximately generalize to arbitrary ML methods:

$$Err \approx GCV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(x_i)}{1 - dof/n}\right]^2$$

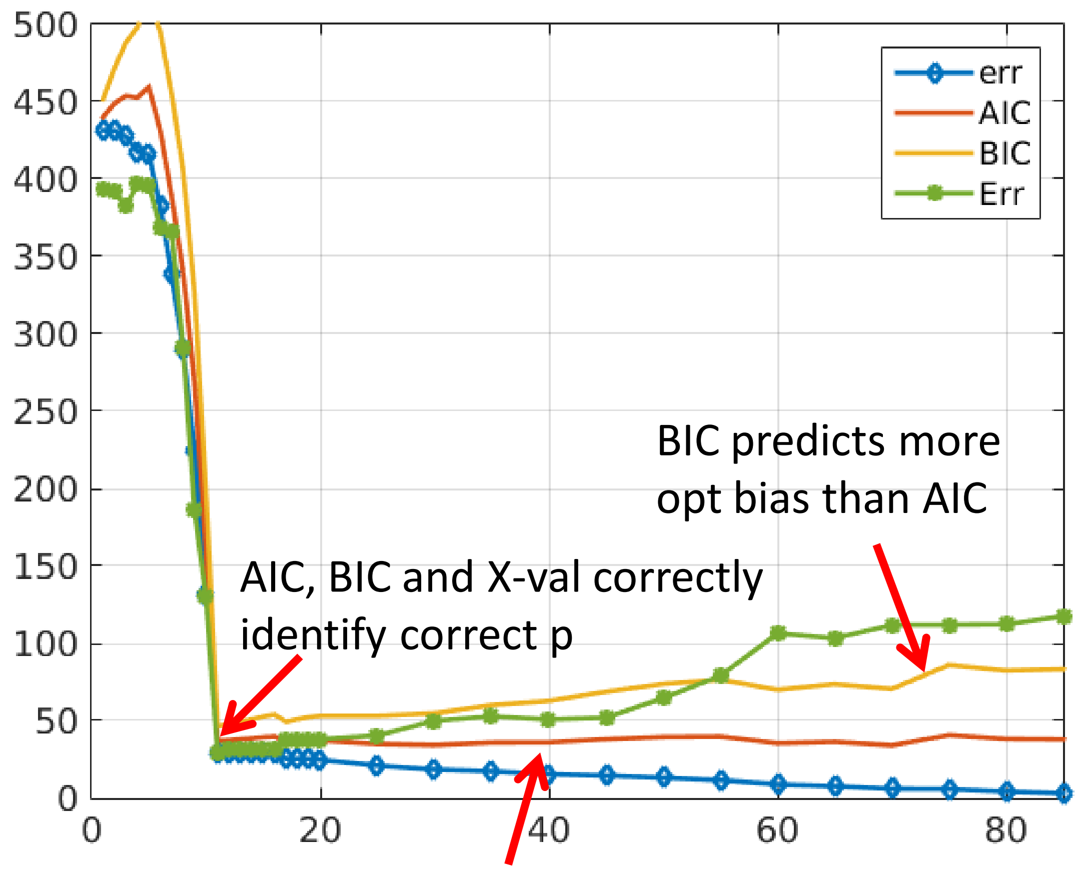
where dof=Tr(S) is the dof of ML method. This may be used as a criterion like AIC or BIC for selection of dof.

Complexity selection using cross-validation

Example - selecting complexity in linear model using AIC/BIC or cross-validation

- 100 examples of $Y=X^T\theta^*$ is generated for given θ^* having 11 nonzero elements (p=11), where $x_{1..p}=x_{1..100}$
- The linear model $Y=X^T\theta$ is fit against $X_{1..p'}$ with p' varying from 1 to 100.
- Training error is obtained from err=E[y^-y]2
- Test error is obtained on 100 examples of $Y=X^T\theta^*$ generated independently
- AIC and BIC are evaluated using (adjusted) noise variance estimate $\sigma\hat{}^2=\text{err}\cdot n/(n-p')$

Complexity selection using cross-validation



AIC underestimates EPE

Complexity selection using cross-validation

One has to be careful about several points for cross-validation to work properly :

- Cross-validation data should be entirely independent of the data used in training **as well as** the manner of training
- This means in particular that the **type of cross-validation** as well as the **cross-validation split** must be chosen before any training of ML models began – one cannot try several cross-validation types and then choose the one that "worked best", that would destroy the independence

Complexity selection using cross-validation

- For a model chosen via max of cross-validated performance, cross-validation error is no longer an accurate predictor of generalization ability – pretty much the same way we had it for the training data error
- In that case, ***a 3rd independent set of data – typically called test data – must be used to evaluate Err.*** Test data cannot be involved in either model training of cross-validation, and should be kept entirely outside of these two processes.

Complexity selection using cross-validation

- If any data pre-processing is involved in ML – such as pre-selecting significant features or using feature banks, etc. – such pre-processing should be done entirely inside the training data, without involving or touching the cross-validation data
- *Cross-validation data set should not be touched in any way until the model evaluation*

Complexity selection using cross-validation

Example:

- Imaging a classification problem using a large number of predictors
  - Pre-screening predictors **using all data** in order to find “good” predictors – such that show strong correlation with output – is a common practice
  - One can then use that small set of predictors to construct a strong ML model
  - One then uses cross-validation to estimate generalization error Err

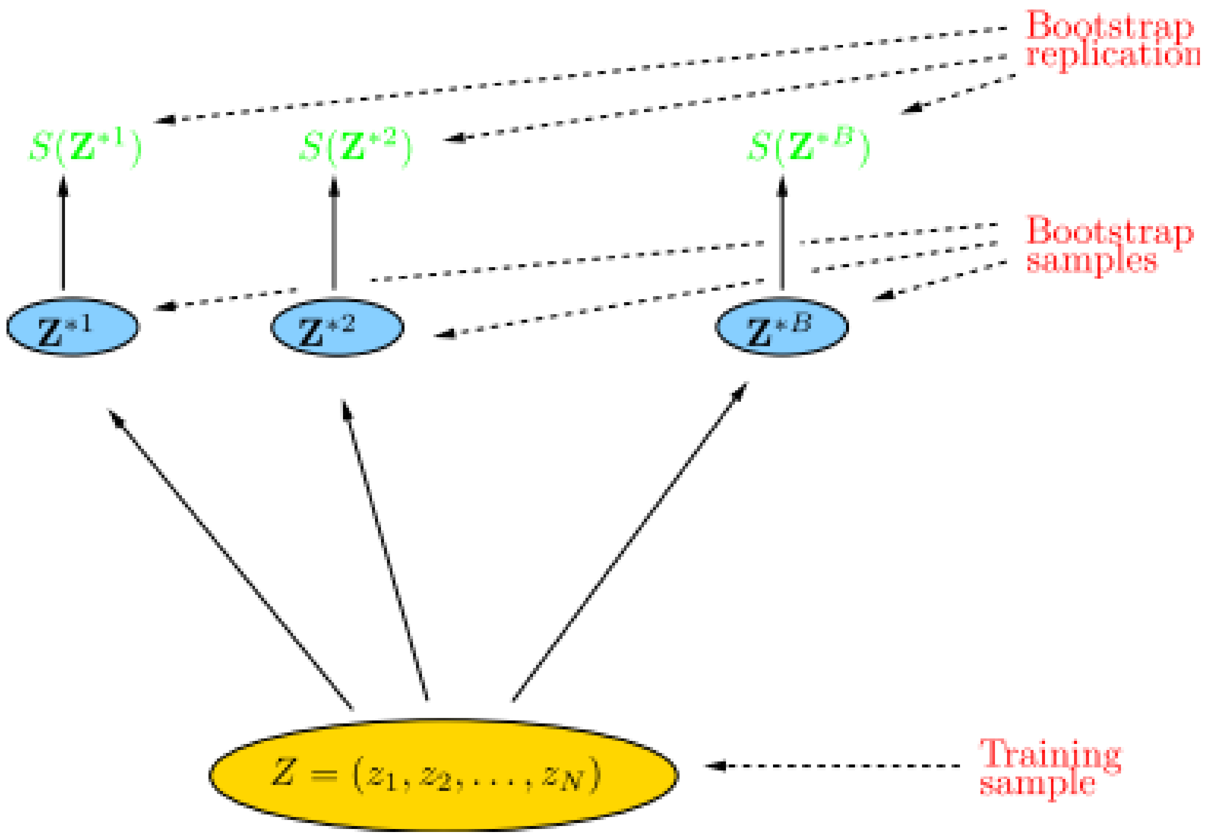
Bootstrap and model averaging methods

- Bootstrap methods depart from the same ideological standpoint as cross-validation
- The key idea of bootstrapping is to use “fake” datasets generated from whatever data we have by re-drawing examples with replacement at random, from the fixed corpus of available data
- Bootstrapping in a sense relies simulating the precise statistical learning setting by spanning random training datasets from single data base

Bootstrap and model averaging methods

- For example, having a data of D=1,000,000 points, one can generate a 1000 random datasets each containing 100,000 examples chosen at random with replacement from D
- One can then use each of those dataset to learn a different ML models *f*^ and then use these 1000 models to construct a new predictor E*f*^

Bootstrap and model averaging methods



Bootstrap and model averaging methods

- Bootstrap allows estimating Err directly from bootstrap replicas (here f$^{(b)}$ means model fitted using bth bootstrap dataset):

$$EPE_{boot} = \frac{1}{B}\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n} L(y_i, \hat{f}^{(b)}(x_i))$$

- Estimate the error in Err estimate by using bootstrap replicas

$$\mathrm{var}(EPE_{boot}) = \mathrm{var}\left(\frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}^{(b)}(x_i))\right)$$

Bootstrap and model averaging methods

- Prediction using bootstrap models are obtained via **model averaging**, which is also accidentally the foundation of one of the most successful currently **random forest** and **decision tree-bagging** methods:

$$E[\hat{f}(x_0)] = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{(b)}(x_0)$$

- Prediction's error or variance of that estimate can be estimated from bootstrap models as well

$$\mathrm{var}[\hat{f}(x_0)] = \mathrm{var}\left(\hat{f}^{(b)}(x_0)\right)$$

QUESTIONS FOR SELF-CONTROL

- Explain why we need cross-validation in machine learning.
- What is a model’s degrees of freedom and how dof relates to model’s performance?
- Let’s say one model has dof=10 and another model has dof=1000. What can be said about the two model’s bias, variance, and total prediction error?
- What is high-bias condition? How one can get a model out of high-bias condition?
- What is high-variance condition? How one can get a model out of high-variance condition?
- Identify the irreducible error, bias, and variance parts in the total error formula for KNN:

$$EPE_{KNN}(x_0) = \sigma^2 + [f(x_0) - \frac{1}{K}\sum_{l=1}^{K} f(x_{(l)})]^2 + \frac{\sigma^2}{K}$$

- Identify the irreducible error, bias, and variance parts in the total error formula for linear model (if any parts are missing, explain why):

$$EPE = \sigma_\varepsilon^2 + \frac{p}{n}\sigma_\varepsilon^2$$

-

- Write down the formulas for TPE, EPE, and optimism bias of a linear model.
- Write the formula for optimism bias in terms of model's dof and noise σ^2
- Explain the purpose of each of the usual cross-validation data partition, namely – training, validation, and test data segments
- Describe hold-out cross-validation.
- Describe K-fold cross-validation
- Describe leave-one-out cross-validation
- Write down the Akaike information criterion for model selection.
- Write down the Bayesian information criterion for model selection.
- Define VC-dimensions.
- Write down the Vapnik's bound on generalization error given VC dimensions.
- What is VC dimensions of a plain in n dimensions.
- Explain how one can use cross-validation for model selection.
- Explain how to use bootstrapping to obtain an average estimate of a model's prediction error
- Explain how to use bootstrapping to estimate the average prediction f^(x) and the variance of that prediction for a model.

- Carefully examine the theoretical formula for the optimism bias: opt=2dof/nσ_ε^2.
  - How does this formula depend on model dof, training sample size n, and σ_ε^2 – the data's intrinsic variability or noise?
  - Use this formula to argue that no matter the situation, the overfitting problem can be resolved by using more and more data during training, as long as dof of ML model will remain bounded from above dof<M.
- Consider in-sample EPE for K=1NN algorithm obtained in class:

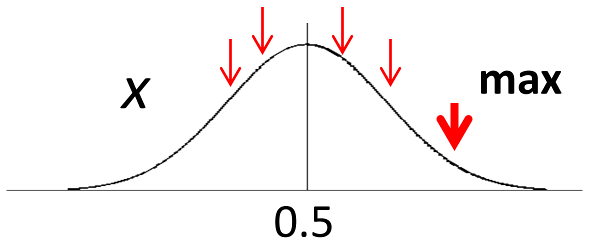
$$EPE_{KNN}(x_0) = \sigma^2 + [f(x_0) - f(x_{(1)})]^2 + \frac{\sigma^2}{K=1}$$

  For K=1, TPE of KNN is trivial, because the nearest neighbor to any *training* point is always that point self, so actually $TPE_{K=1NN}=0$. What is the optimism bias of K=1NN?
- Suppose we are not happy with TPE=0 as an predictions' error estimate for K=1NN. Suggest a minimal modification to the error estimation procedure that can give us a more meaningful estimate of the prediction errors. How does this modification compare to any of the cross-validation approaches we discussed?
- [Advanced] Derive TPE for the KNN algorithm with general K. What is the optimism bias for general K NN algorithm?

ADVANCED

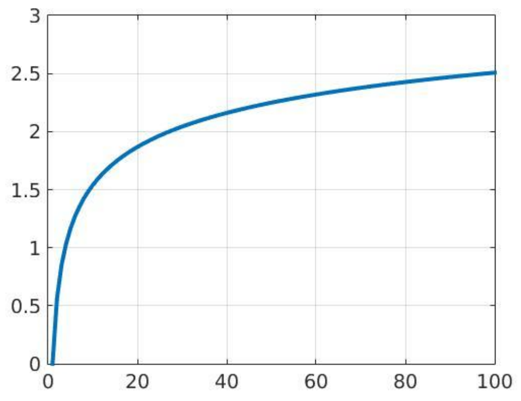
Optimism Bias

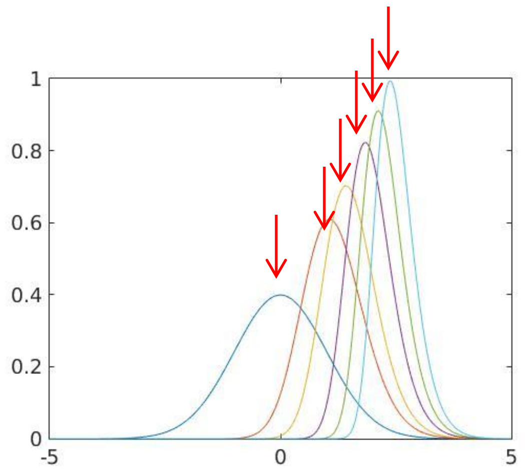
Optimism bias for max of n normal r.v.: $X_n = \max(x_{1..n})$



$$P(X_n) = nP\{x = X_n\}P\{x \le X_n\}^{n-1}$$
$$= nP(x)F(x)^{n-1}$$

Optimism bias in in units of STD(x) vs n



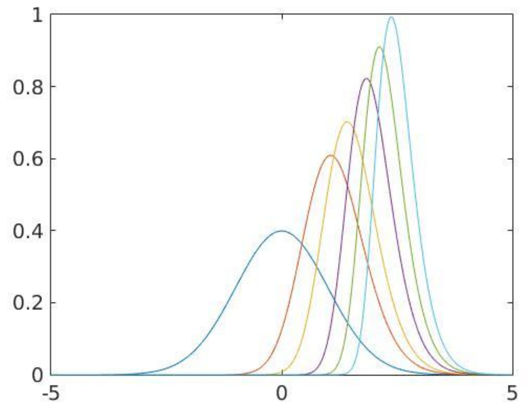


n=5,10,25,50,100

Optimism Bias

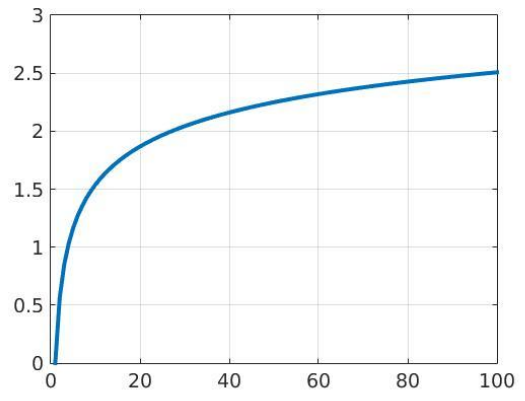
As an example, $E[X_{100}]$ is greater than $E[x]$ by almost 2.5σ

P(X_n)



n=5,10,25,50,100

Optimism bias for max of n Normal r.v. vs n in units of σ



Optimism Bias

- Optimism Bias in linear model

$$\{x_i, y_i\} \rightarrow \ y = x^T \theta$$

$$\min \ |y - X\theta|^2 \Rightarrow \hat{\theta} = (X^T X)^{-1} X^T y,$$

$$\hat{y} = X(X^T X)^{-1} X^T y = Hy$$

Optimism Bias

- The critical relationship is

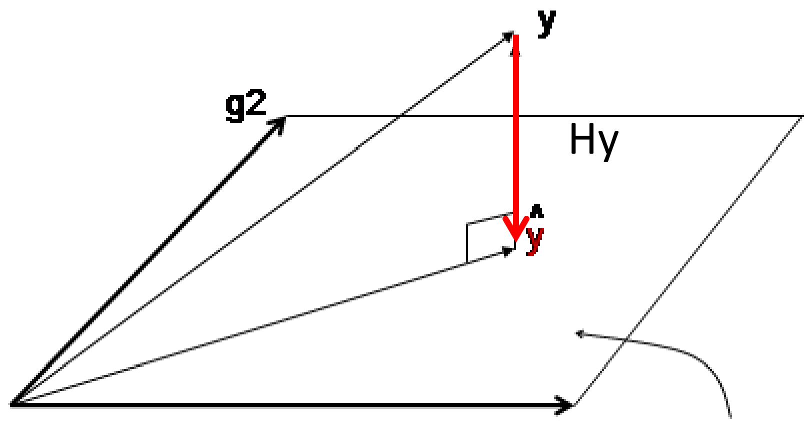
$$\hat{y} = X(X^T X)^{-1} X^T y = Hy$$

Optimism Bias

- One can observe that the hat-matrix H has projection operator property $H^2y=Hy$:

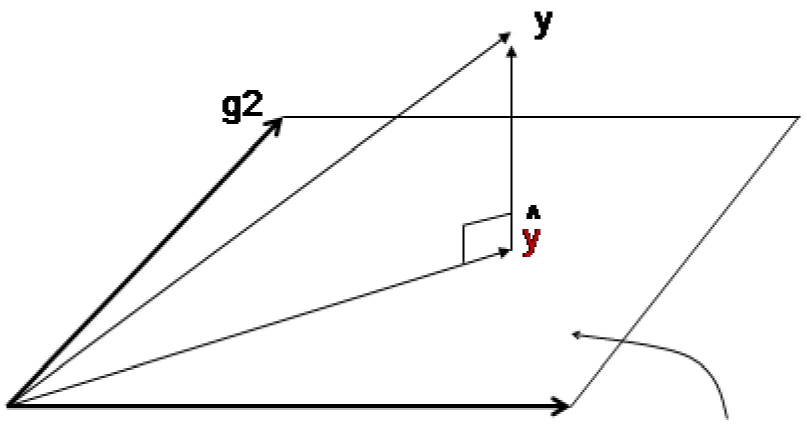
$$H^2 = H \Leftarrow$$

$$(X(X^TX)^{-1}X^T)\cdot(X(X^TX)^{-1}X^T) = X(X^TX)^{-1}X^T$$



Optimism Bias

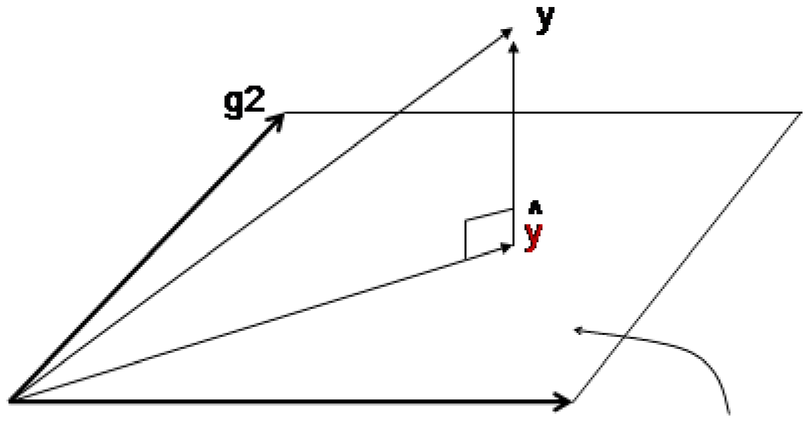
- Projector operators P are such operators that take any vector "vertically" into a linear subspace, so that second time P is applied, nothing happens – P(Px)=Px



Optimism Bias

- The above makes it rather obvious that then there will exists some orthonormal basis for H (specifically the basis of the subspace of H) where H will take the form a part of identity matrix:

$$H = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0 & 0 & \\ & & & \\ & & & \cdots \end{bmatrix}$$



Optimism Bias

- This allows us to establish that Tr[H] – an invariant property of matrices with respect to orthogonal transformations, for hat-matrix H is simply p – the size of the projection subspace, equivalently the dimensionality of solution θ

$$H = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0 & 0 & \\ & & & \dots \end{bmatrix} \Rightarrow \sum H_{ii} = \mathrm{Tr}[H] = p$$

Optimism Bias

$$y = x^T \theta^* + \varepsilon \Rightarrow$$

$$\overline{err} = \frac{1}{n} |(I - H) y|^2 = (1 - \frac{p}{n}) \sigma^2$$

- Since essentially H is a part of identity matrix I of length p, Tr(I-H)=n-p. Note that larger dof get smaller training error and in the limit p=n the training error is err=0 !

$$I - H = \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0 & 1 & \\ & & & \dots \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0 & 0 & \\ & & & \dots \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \\ 0 & 0 & 0 & \\ 0 & 0 & 1 & \\ & & & \dots \end{bmatrix}$$

Optimism Bias

- It is a simple exercise in algebra then to show that the mean training error of linear model is (assuming the data really came from a linear model)

$$\overline{err} = \frac{1}{n} | y - X\theta |^2$$

$$= \frac{1}{n} | y - \hat{y} |^2$$

$$= \frac{1}{n} | (I - H) y |^2$$

$$= \frac{1}{n} Tr[(I - H)[\text{cov}(y_i, y_j)]^{(as\ matrix)}]$$

$$= (1 - \frac{p}{n}) \sigma_\varepsilon^2$$

Optimism Bias

- Is this a reasonable behavior? What is the true prediction error of the fitted linear model?
- Suppose the data came from noisy linear model really:

$$y_i = x_i^T \theta^* + \varepsilon_i$$

- Then one can find with a bit of algebra and probability

$$EPE = E[(y - \hat{y})^2] = (1 + \frac{Tr[H]}{n})\sigma^2 = (1 + \frac{p}{n})\sigma^2$$

Local Methods in High Dimensions

The critical step here is

$$E_T x_0^T (\mathbf{X}^T \mathbf{X})^{-1} x_0 \sigma^2 = \sigma^2 \frac{\mathrm{Tr}\; H}{n} = \sigma^2 \frac{p}{n}$$

$$EPE\;(x_0) = \sigma^2 \frac{p}{n} + \sigma^2$$

Optimism Bias

Then:

- The training set error:

$$\overline{err} = (1 - p/n)\sigma^2$$

- The true prediction error

$$EPE = (1 + p/n)\sigma^2$$

- Observations?

Optimism Bias

- Increasing model dof/complexity leads invariably to reduction in the fit's residual-sum-of-squares but **increases** the actual prediction error !
- The optimism bias is EPE-err = 2p/n σ^2, where p=Tr[H]

$$\overline{err} = (1 - p/n)\sigma^2 \qquad EPE = (1 + p/n)\sigma^2$$

Optimism Bias

- Another way to record the optimism bias is

$$opt = EPE - err = 2\frac{Tr[H]}{n} = \frac{2}{n}\sum_{i=1}^{n} \mathrm{cov}(\hat{y}_i, y_i)$$

Considering

$$\hat{y}_i = Hy_i$$

This, in fact, is very general relationship.

Optimism Bias

- In-sample error estimate formula

$$EPE = err + \frac{2}{n} \sum_{i=1}^{n} \mathrm{cov}(\hat{y}_i, y_i)$$

Optimism Bias

This follows from earlier MSE formula rather straightforwardly if we discard the assumption that y and y^ are independent:

$$
\begin{aligned}
E[\overline{err}] &= E\left[\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|^2\right] = \frac{1}{n}\sum_{i=1}^{n} E\,|y_i - \hat{y}_i|^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} \left\{Ey_i^2 - 2Ey_i\hat{y}_i + E\hat{y}_i^2\right\} \\
&= \frac{1}{n}\Sigma\{[(Ey_i)^2 + \mathrm{var}(y_i)] + [(E\hat{y}_i)^2 + \mathrm{var}(\hat{y}_i)] - 2Ey_i\hat{y}_i\} \\
&= \frac{1}{n}\Sigma\{\mathrm{var}(y_i) + (Ey_i - E\hat{y}_i)^2 + \mathrm{var}(\hat{y}_i) - 2(Ey_i\hat{y}_i - Ey_iE\hat{y}_i)\} \\
&= \frac{1}{n}\Sigma\{\underbrace{\mathrm{var}(y_i) + (Ey_i - E\hat{y}_i)^2 + \mathrm{var}(\hat{y}_i)}_{\text{EPE}} - 2\,\mathrm{cov}(y_i, \hat{y}_i)\} \\
&= EPE - \frac{2}{n}\sum_{i=1}^{n} \mathrm{cov}(y_i, \hat{y}_i)
\end{aligned}
$$

Akaike Information Criterion

- AIC was historically originally derived by Akaike for log-likelihood model-estimation and log-likelihood loss function (see Advanced in Probability), where it actually is

$$AIC = -\frac{2}{n}\,\mathrm{loglik} + 2\frac{d}{n}$$

- For additive noise model with Normal noise, that forms evidently reduces to our result before

$$y = f(x) + \varepsilon \Rightarrow \mathrm{loglik} = -\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{2\sigma_\varepsilon^2} = -\frac{n}{2\sigma_\varepsilon^2}\overline{err}$$

Bayesian Information Criterion

- Assume that we have a set of K candidate model-classes each having parameters θ_k. (K here can be thought as the number of parameters in model of certain form θ_k, or models class with different penalty parameters λ, and anything else).
- Suppose that for each class we have a prior belief about the possible parameters of that models expressed by a prior distribution $P(\theta_K|K)$.
- We want to choose the best model given data.

Bayesian Information Criterion

- By Bayes theorem, the probability of each different class K of models being the generator of data, given collected data, is (the integral is over all different models θ_k in that class)

$$\Pr(K \mid D) \propto \Pr(K)\Pr(D \mid K)$$

$$\propto \Pr(K)\int d\theta_K \Pr(D \mid \theta_K, K)\Pr(\theta_K \mid K)$$

Bayesian Information Criterion

- Whenever prior Pr(K) is uniform, the so called Laplace approximation can be used to simplify and approximately evaluate this integral with respect to the dimensionality of the integrand θ_K (where d_K is the number of parameters in θ_K)

$$\log \Pr(D \mid K) \propto \Pr(D \mid \theta_K, K) - \frac{d_K}{2} \log n + O(1)$$

- The first term is recognized as loglik, the rest directly gives BIC

Bayesian Information Criterion

- In the above, BIC expressed the posterior probability of true model class, as defined by data D, to be in the class of models K.
- Therefore, we can actually write using BIC

$$\Pr(K \mid D) = \frac{e^{-BIC_K/2}}{\sum_k e^{-BIC_k/2}}$$

- the exact relative likelihoods of each class of models in the data