Estimator Theory explained *via* Regression

F. Noé[1]

Deep Learning Classes, FU Berlin 2018

Regression

- **Supervised learning** problem
- **Training set**: N points $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in \mathbb{R}^n$, $\mathbf{y}_i \in \mathbb{R}$, $i = 1, \ldots N$.
- **Aim**: *approximate* the equation $f : \mathbb{R}^n \mapsto \mathbb{R}$ underlying the data with a **model** $\hat{y}(\mathbf{x}; \mathbf{w})$ using **parameters** $\mathbf{w} \in \mathbb{R}^n$:

$$\hat{y}_i = \hat{y}(\mathbf{x}_i; \mathbf{w}) \approx f(\mathbf{x}_i)$$

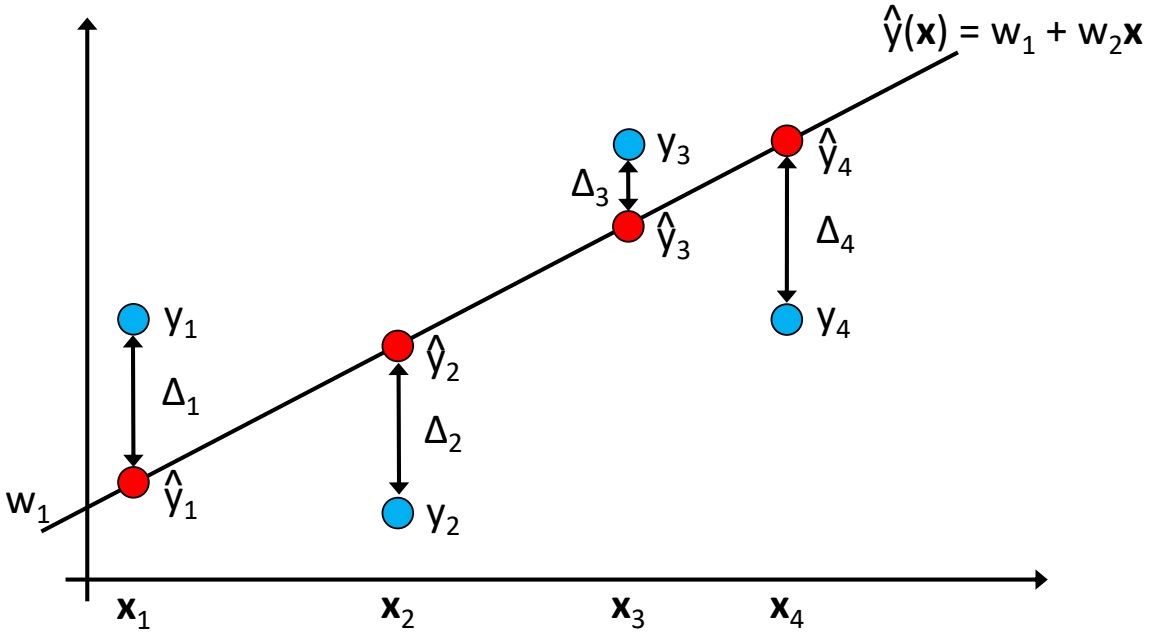
- **Learning problem**: Find $\mathbf{w}$ such that we minimize the residuals $\boldsymbol{\Delta}$:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\boldsymbol{\Delta}\|$$

defined by

$$\Delta_i = y_i - \hat{y}(\mathbf{x}_i; \mathbf{w}).$$

Regression



Example: Fit data with linear function $\hat{y}(\mathbf{x}) = w_1 + w_2\mathbf{x}$ by finding the parameters $\mathbf{w} = (w_1, w_2)$.

We define $\|\mathbf{\Delta}\|$ using the p-norm of $\mathbf{\Delta} \in \mathbb{R}^n$:

$$\|\mathbf{\Delta}\|_p = \left(\sum_{i=1}^{n} |\Delta_i|^p \right)^{1/p},$$

Choice of p determines the type of regression problem:

p	Learning Problem	Name
1	$\min_{\mathbf{w}} \sum_{i=1}^{n} \lvert\Delta_i\rvert$	L^1 (linear) optimization
2	$\min_{\mathbf{w}} \sum_{i=1}^{n} \Delta_i^2$	Least squares (Gauss $\sim$1800)
∞	$\min_{\mathbf{w}} \max_i \Delta_i$	Tschebyscheff regression

- Most supervised machine learning problems can be formulated as the problem to minimize a **cost** or **loss** function $C(\mathbf{X}, \mathbf{Y}, \theta)$.
  - $\mathbf{X} \in \mathbb{R}^{N \times n}$ is the matrix of N input data points or features. A feature has n dimensions.
  - $\mathbf{Y} \in \mathbb{R}^{N \times l}$ are the labels. A label has l dimensions.
  - The pair $(\mathbf{X}, \mathbf{Y}) = (\mathbf{x}_i, \mathbf{y}_i)_{i=1,\dots,N}$ forms the **training data**.
- Here we call $\theta = \mathbf{w}$. We consider univariate function regression, i.e. the labels can be written as a vector $\mathbf{y} \in \mathbb{R}^N$.
  We choose the **mean squared error** as loss function:

$$C(\mathbf{X}, \mathbf{y}, \mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}(\mathbf{x}_i; \mathbf{w}))^2$$

  Note that the mean squared error and the 2-norm $\sqrt{C(\mathbf{X}, \mathbf{y}, \mathbf{w})} = \|\mathbf{y} - \hat{\mathbf{y}}\|_2$ have the same minimum.
- **Learning problem**: seek $\mathbf{w}$ such that the loss function is minimal:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} [y_i - \hat{y}(x_i; \mathbf{w})]^2$$

  The prefactor N^{-1} has no influence on $\hat{\mathbf{w}}$.

Equivalent statistical interpretation (Gauss): Assume that observations y_i are produced from $\hat{y}(\mathbf{x}_i; \mathbf{w})$ with an additive measurement error that is *independent, identically distributed (iid)* from a *Normal distribution*.

$$y_i = \hat{y}(\mathbf{x}_i; \mathbf{w}) + \Delta_i \qquad \Delta_i \sim \mathcal{N}(0, 1) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\Delta_i^2/2}$$

We seek the **maximum likelihood estimator** $\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} L(\mathbf{X}, \mathbf{Y}, \mathbf{w})$ with

$$L(\mathbf{X}, \mathbf{Y}, \mathbf{w}) = \mathbb{P}\left[y_1 = \hat{y}(\mathbf{x}_1), ..., y_N = \hat{y}(\mathbf{x}_N) \mid \mathbf{w}\right] = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}(y_i - \hat{y}(\mathbf{x}_i; \mathbf{w}))^2}$$

This is equivalent with:

$$\begin{aligned}
\arg\max L(\mathbf{X}, \mathbf{Y}, \mathbf{w}) &= \arg\max \log L(\mathbf{X}, \mathbf{Y}, \mathbf{w}) \\
&= \arg\max \sum_{i=1}^{N} -\frac{1}{2} \left(y_i - \hat{y}(\mathbf{x}_i; \mathbf{w})\right)^2 - \log\left(\sqrt{2\pi}\right) \\
&= \arg\min \sum_{i=1}^{N} \left(y_i - \hat{y}(\mathbf{x}_i; \mathbf{w})\right)^2
\end{aligned}$$

Linear Least Squares Regression

- In a linear regression problem, the model has the form:

$$\hat{y}(\mathbf{x}_i; \mathbf{w}) = \mathbf{x}_i^\top \mathbf{w} = \sum_{j=1}^{n} x_{ij} w_j$$

- Matrix notation: $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$. Linear least squares (LLS) regression problem:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

- **Featurization**: Starting from some initial data $\mathbf{r}_i \in \mathbb{R}^d$, we define n **basis functions** or **feature functions** $\phi_j : \mathbb{R}^d \to \mathbb{R}$. Thus every datapoint is featurized:

$$\mathbf{r}_i \to \mathbf{x}_i = (\phi_1(\mathbf{r}_i), ..., \phi_n(\mathbf{r}_i))^\top .$$

- Our linear regression model then has the form

$$\hat{y}(\mathbf{r}_i; \mathbf{w}) = w_1 \phi_1(\mathbf{r}_i) + ... + w_n \phi_n(\mathbf{r}_i)$$

  with can be nonlinear in $\mathbf{r}$.
- **Example**: with $\phi = \{1, r, r^2\}$ and $\mathbf{w} = (w_1, w_2, w_3)$ we can fit a quadratic function $f(r) = w_1 + w_2 r + w_3 r^2$.

The vector $\mathbf{w} \in \mathbb{R}^n$ is the solution to $\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2$ exactly if it fulfills the normal equations

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

The linear regression problem has a unique solution exactly if the rank of $\mathbf{X}$ is maximal, i.e. $\mathrm{rk}(\mathbf{X}) = n$.
Direct inversion (numerically unstable and inefficient):

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^+ \mathbf{y}.$$

where $\mathbf{X}^+$ is the Moore-Penrose pseudoinverse of $\mathbf{X}$.
Defining the covariance matrices

$$\mathbf{C}_{XX} = \mathbf{X}^\top \mathbf{X}$$
$$\mathbf{C}_{XY} = \mathbf{X}^\top \mathbf{y}$$

(here $\mathbf{C}_{XY} \in \mathbb{R}^{n \times 1}$, but it is a matrix if we have multiple regression targets) the formal solution can be written as:

$$\mathbf{w} = \mathbf{C}_{XX}^{-1} \mathbf{C}_{XY}.$$

Validation and hyperparameter selection

- **Validation**: LLS solution gives us the **in-sample training error**:

$$E_{\text{in}} = C(\mathbf{X}^{\text{train}}, \mathbf{y}^{\text{train}}, \hat{\mathbf{w}}) = \left\|\mathbf{y}^{\text{train}} - \mathbf{X}^{\text{train}}\hat{\mathbf{w}}\right\|_2,$$

  but we would like to validate how good the learnt model **predicts** an independent data set, i.e. the out-of-sample **validation or test error** E_{out}:

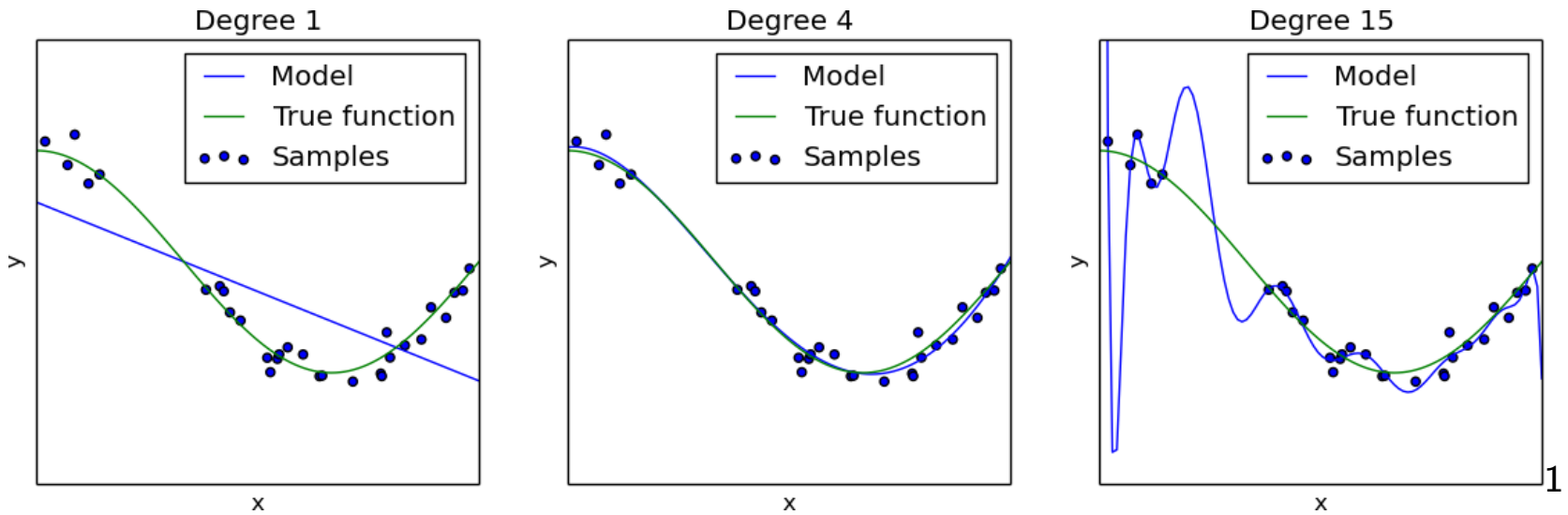
$$E_{\text{out}} = C(\mathbf{X}^{\text{val}}, \mathbf{y}^{\text{val}}, \hat{\mathbf{w}}) = \left\|\mathbf{y}^{\text{val}} - \mathbf{X}^{\text{val}}\hat{\mathbf{w}}\right\|_2,$$

- **Hyperparameter selection**: Hyperparameters cannot not be obtained from the learning algorithm (here LLS). For example, the number of type of feature functions ϕ .
- **Example**: The type of function ϕ used for training cannot be determined by minimizing the training error. For example, the model

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{N} w_i 1_{\mathbf{x}_i}(\mathbf{x}) \quad \text{with} \quad 1_{\mathbf{x}_i}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} = \mathbf{x}_i \\ 0 & \mathbf{x} \neq \mathbf{x}_i \end{cases}$$

  has zero training error, but predicts $f(\mathbf{x}) = 0$ for every point $\mathbf{x}$ not in the training set.

Underfitting vs. Overfitting



[1]

[1] From http://scikit-learn.org

Validation

- **Data-based validation** is an effective way to solve the hyperparameter selection problem:
- Divide dataset into
  - **training set** $(\mathbf{X}^{\text{train}}, \mathbf{y}^{\text{train}})$
  - **validation set** $(\mathbf{X}^{\text{val}}, \mathbf{y}^{\text{val}})$.
- Learn parameters using the training set:

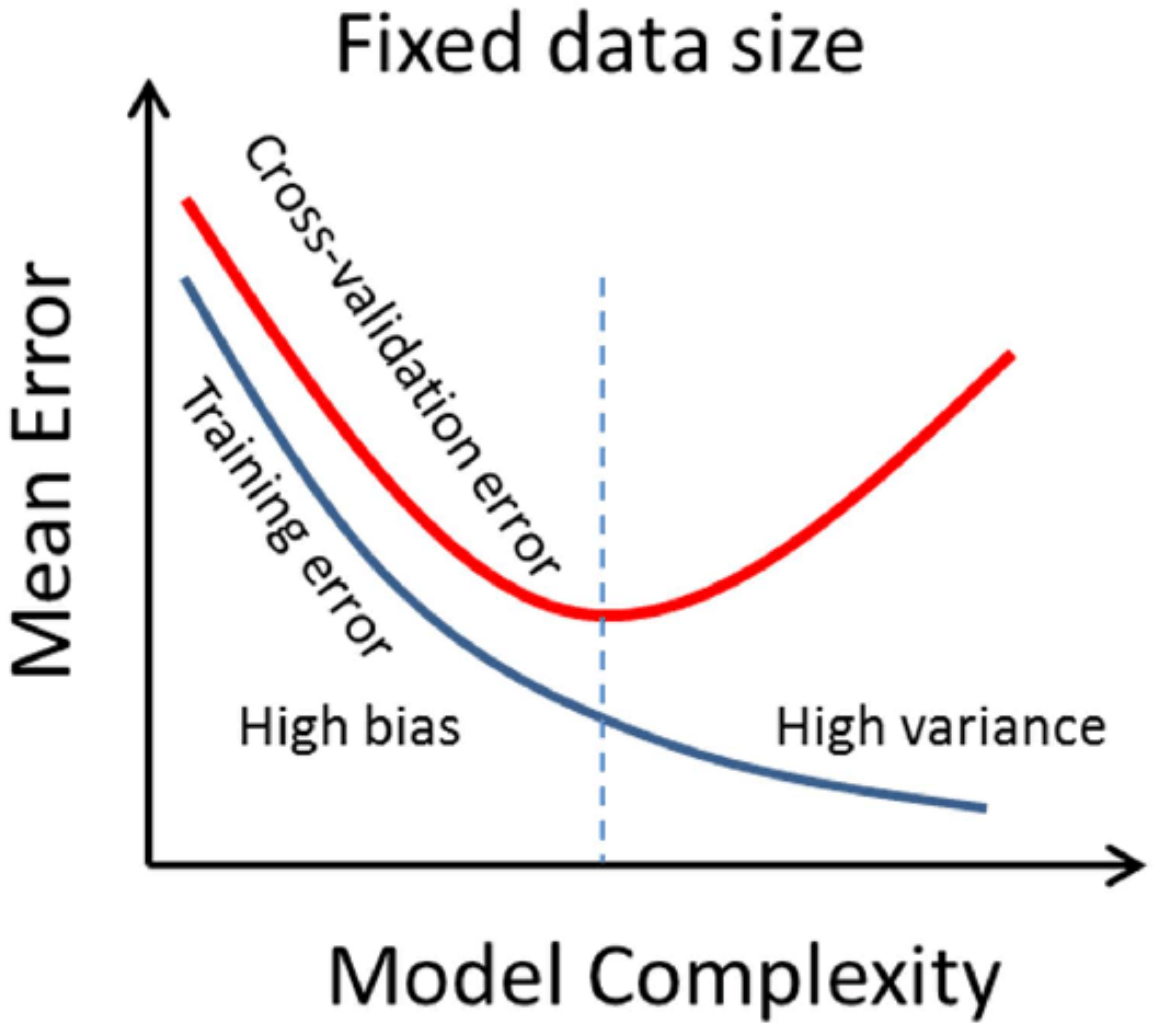
$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left\| \mathbf{y}^{\text{train}} - \mathbf{X}^{\text{train}} \mathbf{w} \right\|_2$$

  The resulting residual $E_{\text{in}} = \left\| \mathbf{y}^{\text{train}} - \mathbf{X}^{\text{train}} \hat{\mathbf{w}} \right\|_2$ is the **training error** or **training loss**.
- The error of the learnt model in predicting data not used for the training,

$$E_{\text{out}} = \left\| \mathbf{y}^{\text{val}} - \mathbf{X}^{\text{val}} \hat{\mathbf{w}} \right\|_2$$

  is called the **validation** or **error/loss**. It provides a metric to validate how well the model generalizes to new data
- **Choose hyperparameters** by minimizing the validation error. (careful: selecting hyperparameters *and* computing E_{out} requires a third *test set* or a more advanced validation routine).

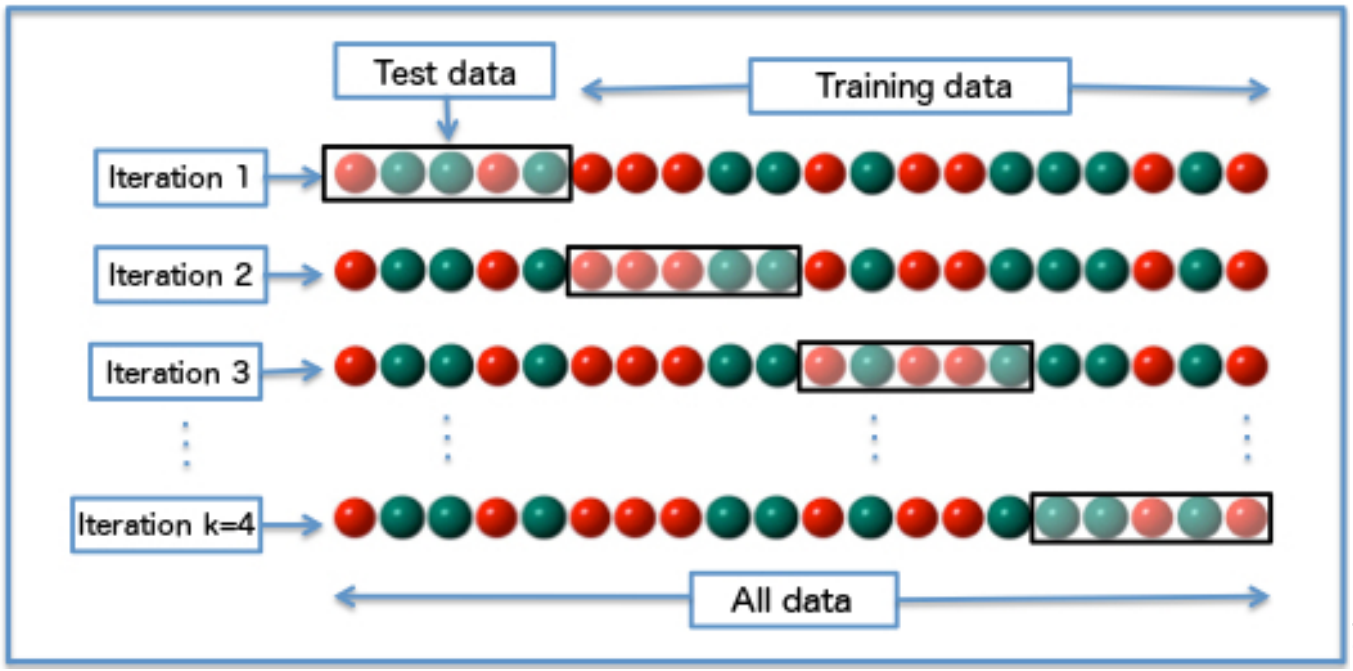
Underfitting vs. Overfitting



Cross-validation

- Pathological division where rare events / outliers are included only in training or validation set can lead to undesirable behavior.
- Methods to "shuffle" training and test data to reduce the bias from the data splitting.
- **Cross-validation** is a simple and widely used approach:
  1. Split the data into k nonoverlapping folds $(\mathbf{X}^i, \mathbf{y}^i)$. The complementary sets are $(\mathbf{X}^{-i}, \mathbf{y}^{-i})$.
  2. For each fold i:
     1. Train learning algorithm on training data:
     $$\hat{\mathbf{w}}^i = \arg\min_{\mathbf{w}} \left\| \mathbf{y}^{-i} - \mathbf{X}^{-i}\mathbf{w} \right\|_2$$
     2. Compute validation error:
     $$E^i_{\text{out}} = \left\| \mathbf{y}^i - \mathbf{X}^i \hat{\mathbf{w}}^i \right\|_2$$
  3. Cross-validation error is then given by:
  $$E_{\text{out}} = \frac{1}{k} \sum_{i=1}^{k} E^k_{\text{out}}.$$

Cross-Validation



[1]

[1] From https://en.wikipedia.org/wiki/Cross-validation_(statistics)

We now explicitly distinguish between the true function f that is sampled by a given set of observations $(\mathbf{x}_i, y_i)$:

$$y_i = f(\mathbf{x}_i) + \Delta_i \qquad \Delta_i \sim \mathcal{N}(0, 1)$$

and the estimator $\hat{\mathbf{y}}(\mathbf{x}; \mathbf{w})$.

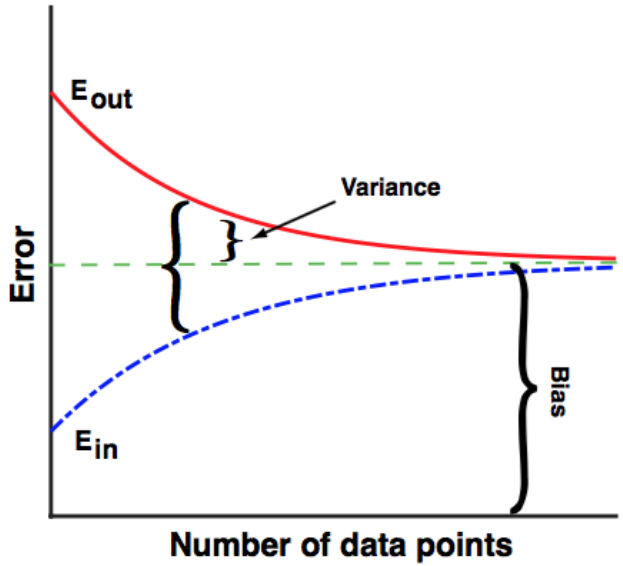
Learning problem

Learn function $f(\mathbf{x})$ by selecting function $\hat{y}(\mathbf{x})$ from a hypothesis set $\mathscr{H}$, which (in some sense) performs a best approximation $\hat{y} \approx f$.

Prediction problem

How can the learning problem be meaningfully defined if $f(x)$ can, in principle, take any value on *unobserved* inputs?

Answer: a meaningful definition of learning is that the fitted model will perform approximately as well in predicting unseen data as it did in approximating training data ($E_{\text{in}} \approx E_{\text{out}}$).



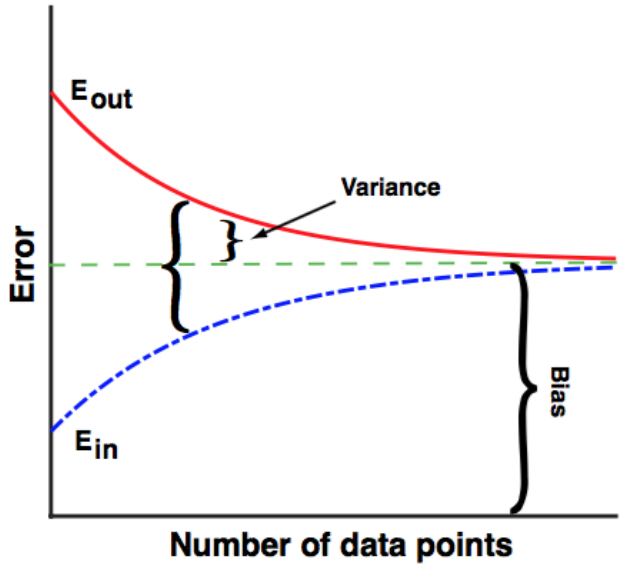
[1]

We assume that the true function f is sufficiently complicated so that we cannot learn it exactly, i.e.

$$\sum_{i=1}^{N} \left(f(\mathbf{x}_i) - \hat{\mathbf{y}}(\mathbf{x}_i; \mathbf{w})\right)^2 > 0 \ \forall \mathbf{w}$$

Even in the limit $N \to \infty$ we maintain an **asymptotic error** $E_{\text{in}} = E_{\text{out}}$, called the **model bias** $\to$ property of the function class $\mathscr{H}$.

[1] From Mehta et al, arXiv:1803.08823v1

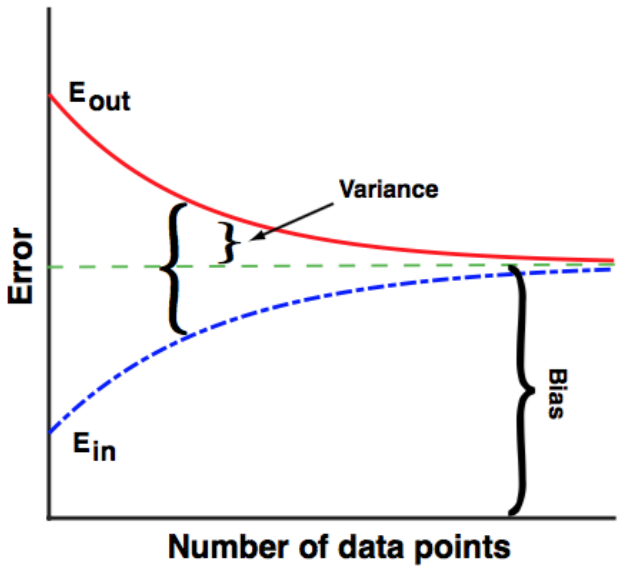


[1]

Typical behavior:

- After quick initial drop (not shown), E_{in} increases with N towards the model bias.
- E_{out} decreases with increasing N as more cases are observed and thus covered by the model.
- The **generalization gap** $E_{\text{out}} - E_{\text{in}}$ (due to overfitting) decreases with increasing N.

[1] From Mehta et al, arXiv:1803.08823v1

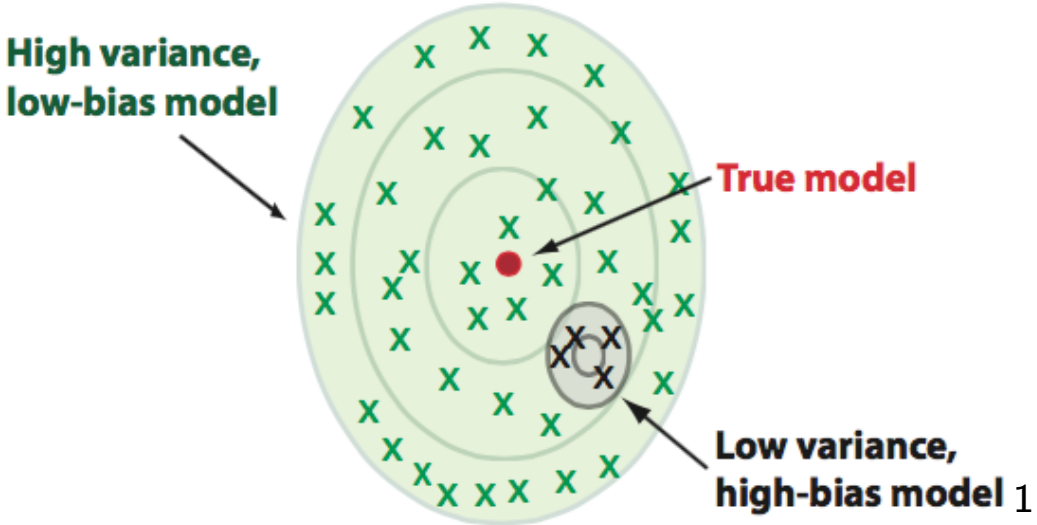


[1]

Insights:

- It is not sufficient to minimize E_{in}, as E_{out} may be large. $\rightarrow$ *regularization*.
- As the true bias is not practically available, one minimizes E_{out}.

[1] From Mehta et al, arXiv:1803.08823v1



[1]

- **Bias-variance tradeoff**: The more/less expressive the model, the larger/smaller the fluctuations, respectively.
- To minimize E_{out}, it is sometimes better to use a more-biased model with small variance than a less-biased model with large variance.
- Asymptotically, i.e. with increasing size of the training set, complex models will perform better than simpler models as they have reduced bias.
- Optimal model selection depends on the amount of training data.

[1] From Mehta et al, arXiv:1803.08823v1

Bias-Variance decomposition

Task: for a given estimator, e.g. LLS, model the behavior of the out-of-sample MSE without knowing the true function f:

$$E_{\text{out}} = C(\mathbf{X}^{\text{val}}, \mathbf{y}^{\text{val}}, \hat{\mathbf{y}}^{\text{val}}) = \left\| \mathbf{y}^{\text{val}} - \hat{\mathbf{y}}^{\text{val}} \right\|_2^2,$$

where $\mathbf{X}^{\text{val}} = (\mathbf{x}_1^{\text{val}}, \ldots, \mathbf{x}_N^{\text{val}})^\top$ are the features of the validation set, $\mathbf{y}^{\text{val}}$ are the corresponding observations and $\hat{\mathbf{y}}^{\text{val}}$ are the predictions of the estimator.

Idea: Compute **expected** E_{out} of given estimator $\hat{y}$ on all data $y = f(\mathbf{x}) + \varepsilon$ drawn from the true model $f(\mathbf{x})$ with following approach:

1. Fix observation points $\mathbf{x}_i$
2. Repeat:
   1. Run experiment, observe training data $(\mathbf{X}, \mathbf{y}^{\text{train}}) = (\mathbf{x}_i, y_i^{\text{train}})_{i=1,\ldots N}$ and train the estimator $\hat{y}(\mathbf{x})$.
   2. Repeat experiment, observe validation data $(\mathbf{X}, \mathbf{y}^{\text{val}}) = (\mathbf{x}_i, y_i^{\text{val}})_{i=1,\ldots N}$.
3. Compute expectation $\mathbb{E}$ over observations $\mathbf{y}^{\text{train}}$ and $\mathbf{y}^{\text{val}}$ by averaging over noise realizations.

$$\mathbb{E}\left[C(\mathbf{X}, \mathbf{y}^{\text{val}}, \hat{\mathbf{y}}^{\text{val}})\right] = \mathbb{E}\left[\sum_{i=1}^{N} \left(y_i^{\text{val}}(\mathbf{x}_i) - \hat{y}(\mathbf{x}_i)\right)^2\right]$$

Bias-Variance decomposition

$$\mathbb{E}\left[\sum_{i=1}^{N}\left(y_i^{\text{val}}(\mathbf{x}_i)-\hat{y}(\mathbf{x}_i)\right)^2\right]=\mathbb{E}\left[\sum_{i=1}^{N}\left(y_i^{\text{val}}(\mathbf{x}_i)-f(\mathbf{x}_i)+f(\mathbf{x}_i)-\hat{y}(\mathbf{x}_i)\right)^2\right]$$

$$=\sum_{i=1}^{N}\underbrace{\mathbb{E}_{\mathbf{y}^{\text{val}}}\left[\left(y_i^{\text{val}}(\mathbf{x}_i)-f(\mathbf{x}_i)\right)^2\right]}_{=\mathbb{E}_{\mathbf{y}^{\text{val}}}\left[\varepsilon^2\right]=\sigma_\varepsilon^2}$$

$$+\sum_{i=1}^{N}\mathbb{E}_{\mathbf{y}^{\text{train}}}\left[\left(f(\mathbf{x}_i)-\hat{y}(\mathbf{x}_i)\right)^2\right]$$

$$+2\sum_{i=1}^{N}\underbrace{\mathbb{E}_{\mathbf{y}^{\text{val}}}\left[y_i-f(\mathbf{x}_i)\right]}_{=\mathbb{E}_{\mathbf{X}^{\text{val}}}[\varepsilon]=0}\mathbb{E}_{\mathbf{y}^{\text{train}}}\left[f(\mathbf{x}_i)-\hat{y}(\mathbf{x}_i)\right]$$

$$=\sum_{i=1}^{N}\underbrace{\sigma_\varepsilon^2}_{\text{Noise}}+\underbrace{\mathbb{E}_{\mathbf{y}^{\text{train}}}\left[\left(f(\mathbf{x}_i)-\hat{y}(\mathbf{x}_i)\right)^2\right]}_{\text{Estimator error}}$$

Bias-Variance decomposition

Decomposition of the estimator error:

$$\mathbb{E}\left[(f(\mathbf{x}_i)-\hat{y}_{\mathbf{X}}(\mathbf{x}_i))^2\right] = \mathbb{E}\left[\left(\underbrace{f(\mathbf{x}_i)-\mathbb{E}\left[\hat{y}(\mathbf{x}_i)\right]}_{a}+\underbrace{\mathbb{E}\left[\hat{y}(\mathbf{x}_i)\right]-\hat{y}(\mathbf{x}_i)}_{b}\right)^2\right]$$
$$= \mathbb{E}\left[a^2\right]+\mathbb{E}\left[b^2\right]-2\mathbb{E}\left[ab\right]$$

The mixed term disappears (abbreviating $\bar{y}_i = \mathbb{E}\left[\hat{y}(\mathbf{x}_i)\right]$):

$$\mathbb{E}\left[ab\right] = \mathbb{E}\left[(f(\mathbf{x}_i)-\bar{y}_i)(\bar{y}_i-\hat{y}(\mathbf{x}_i))\right]$$
$$= \mathbb{E}\left[f(\mathbf{x}_i)\bar{y}_i\right]-f(\mathbf{x}_i)\mathbb{E}\left[\hat{y}(\mathbf{x}_i)\right]-\bar{y}_i^2+\bar{y}_i\mathbb{E}\left[\hat{y}(\mathbf{x}_i)\right]=0$$

The square **bias** $\mathbb{E}\left[a^2\right]$ is the asymptotic estimation error in the infinite data limit from the true value:

$$\mathbb{E}\left[a^2\right] = \mathrm{Bias}^2 = \sum_{i=1}^{N}\left(f(\mathbf{x}_i)-\mathbb{E}\left[\hat{y}(\mathbf{x}_i)\right]\right)^2$$

The **variance** $\mathbb{E}\left[b^2\right]$ is the estimator's fluctuation due to finite-sample effects:

$$\mathbb{E}\left[b^2\right] = \mathrm{Var} = \sum_{i=1}^{N}\mathbb{E}\left[\left(\hat{y}(\mathbf{x}_i)-\mathbb{E}\left[\hat{y}(\mathbf{x}_i)\right]\right)^2\right]$$

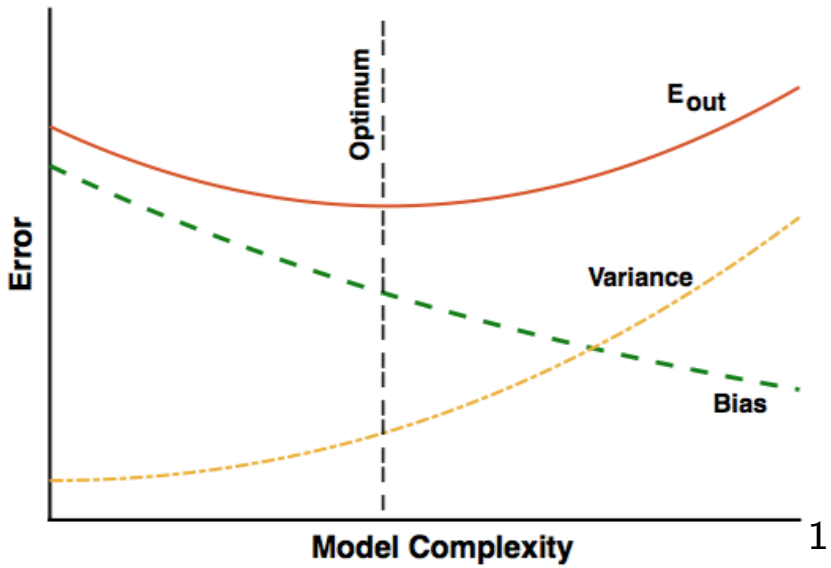
Bias-Variance decomposition

Combining these expressions, we see that the expected out-of-sample error, i.e. the **expected loss** of our model can be decomposed as:

$$E_{\text{out}} = \mathbb{E}\left[\sum_{i=1}^{N}\left(y_i^{\text{val}}(\mathbf{x}_i) - \hat{y}(\mathbf{x}_i)\right)^2\right] = \text{Bias}^2 + \text{Var} + \text{Noise}.$$

- The **optimal model** minimizes the expected loss by **balancing bias and variance**.
- A model is **underfitting** the data if bias is too high.
- A model is **overfitting** the data if variance is too high.
- Since data is often limited, a simple model with a finite bias (i.e. an asymptotic error) may be preferable to a complex model with a high variance.
- Optimal choice depends on the amount of data available. The more data, the more complex models are optimal.

- Model complexity is a property of the function class $\mathscr{H}$. For example, model complexity increases with the number of free parameters (e.g. higher-order polynomials are more complex than the linear model).
- Behavior for fixed N:



[1]

- E_{out} is typically minimal for intermediate complexity. Low-complexity involves a large training error, overly complex models involve large prediction errors due to overfitting.

[1] From Mehta et al, arXiv:1803.08823v1

We assume a linear model $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$, and use the least squares regression estimator:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Then the bias of a single sample point $\mathbf{x}_i$ is:

$$\begin{aligned} \text{Bias} &= f(\mathbf{x}_i) - \mathbb{E}\left[\hat{y}(\mathbf{x}_i)\right] \\ &= \mathbf{x}_i^\top \mathbf{w} - \mathbb{E}\left[\mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}\right] \end{aligned}$$

Using the observation model in matrix form, $\mathbf{y} = \mathbf{X}\mathbf{w} + \varepsilon$, and exploiting $\mathbb{E}_{\mathbf{X}}[\varepsilon] = 0$ shows:

$$\begin{aligned} \text{Bias} &= \mathbf{x}_i^\top \mathbf{w} - \mathbb{E}\left[\mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{X}\mathbf{w} + \varepsilon)\right] \\ &= \mathbf{x}_i^\top \mathbf{w} - \mathbb{E}\left[\mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{X}\mathbf{w} + \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \varepsilon)\right] \\ &= \mathbf{x}_i^\top \mathbf{w} - \mathbb{E}\left[\mathbf{x}_i^\top \mathbf{w} + \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \varepsilon\right] \\ &= \mathbf{x}_i^\top \mathbf{w} - \mathbf{x}_i^\top \mathbf{w} - \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbb{E}[\varepsilon] \\ &= 0 \end{aligned}$$

Thus the least squares estimator $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$ is unbiased!

Then the variance of a single sample point $\mathbf{x}_i$ is:

$$\text{Var} = \mathbb{E}\left[(\hat{y}(\mathbf{x}_i) - \mathbb{E}[\hat{y}(\mathbf{x}_i)])^2\right]$$

Using again the linear model $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ with the least squares regression estimator:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

and use the unbiased estimator result $\mathbb{E}_{\mathbf{X}}[\hat{y}_{\mathbf{X}}(\mathbf{x}_i)] = \mathbf{x}_i^\top \mathbf{w}$, we have:

$$\text{Var} = \mathbb{E}\left[\left(\mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y} - \mathbf{x}_i^\top \mathbf{w}\right)^2\right]$$

Using the observation model in matrix form, $\mathbf{y} = \mathbf{X}\mathbf{w} + \varepsilon$:

$$\begin{aligned}
\text{Var} &= \mathbb{E}\left[\left(\mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top (\mathbf{X}\mathbf{w} + \varepsilon) - \mathbf{x}_i^\top \mathbf{w}\right)^2\right] \\
&= \mathbb{E}\left[\left(\mathbf{x}_i^\top \mathbf{w} + \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \varepsilon - \mathbf{x}_i^\top \mathbf{w}\right)^2\right] \\
&= \mathbb{E}\left[\left(\mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \varepsilon\right)^2\right]
\end{aligned}$$

Using $a^2 = aa^\top$ if a is a scalar:

$$\begin{aligned}
\text{Var} &= \mathbb{E}\left[\left(\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\varepsilon\right)\left(\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\varepsilon\right)^\top\right] \\
&= \mathbb{E}\left[\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\varepsilon\varepsilon^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i\right] \\
&= \mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}\left[\varepsilon\varepsilon^\top\right]\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i
\end{aligned}$$

where we have exploitet that $\mathbf{X}^\top\mathbf{X}$ is a symmetric matrix.
The only component in this expression that depends on the data averaging is $\varepsilon\varepsilon^\top$. Writing $\mathbb{E}\left[\varepsilon\varepsilon^\top\right] = \sigma_\varepsilon^2 I$, where I is the identity matrix and cancelling terms gives:

$$\text{Var} = \sigma_\varepsilon^2\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i$$

We use a **principal component decomposition** of the data

$$\frac{1}{N}\mathbf{X}^\top\mathbf{X} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top$$

where the eigenvectors

$$\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_n]$$

are the principal vectors (normalized as $\mathbf{u}_i^\top\mathbf{u}_i = 1$) and the eigenvalues

$$\mathbf{\Sigma}^2 = \mathrm{diag}(\sigma_{X1}^2, ..., \sigma_{Xn}^2)$$

are the variances of the data along the principal vectors.
Then the estimator variance is:

$$\begin{aligned}\mathrm{Var} &= \frac{\sigma_\varepsilon^2}{N}\mathbf{x}_i^\top\mathbf{U}\mathbf{\Sigma}^{-2}\mathbf{U}^\top\mathbf{x}_i \\ &= \frac{\sigma_\varepsilon^2}{N}\sum_{k=1}^{n}\frac{\mathbf{x}_i^\top\mathbf{u}_k}{\sigma_{Xk}^2}\end{aligned}$$

If we assume $\sigma_{Xk}^2 \equiv 1$, then we have:

$$\text{Var} = \frac{\sigma_\varepsilon^2}{N} \sum_{k=1}^{n} \mathbf{x}_i^\top \mathbf{u}_k$$

Now we compute the expected variance, assuming that $\mathbf{x}_i$ is also from $\mathcal{N}(0, I)$. As a result,

$$\mathbb{E}\left[\mathbf{x}_i^\top \mathbf{u}_k\right] = 1$$

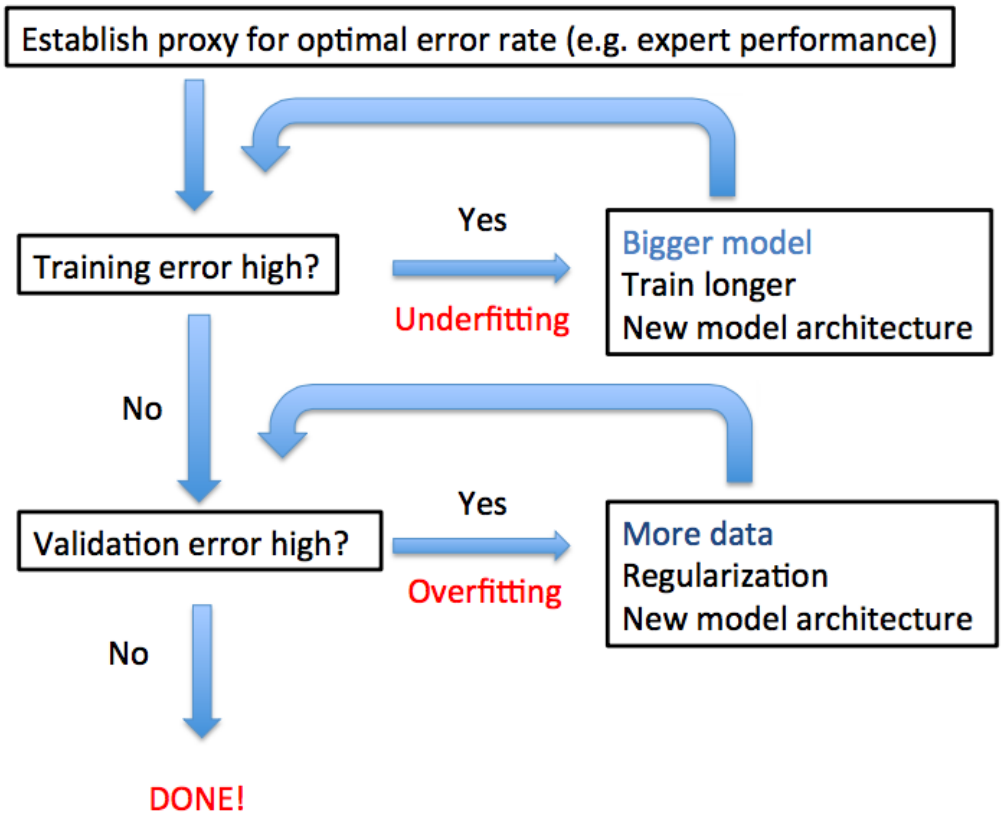
for any $\mathbf{u}_k$ with $\mathbf{u}_k^\top \mathbf{u}_k = 1$, and thus:

$$\mathbb{E}\left[\text{Var}\right] = \mathbb{E}\left[\frac{\sigma_\varepsilon^2}{N} \sum_{k=1}^{n} \mathbf{x}_i^\top \mathbf{u}_k\right] = \sigma_\varepsilon^2 \frac{n}{N}$$

- **Gauss-Markov Theorem**: The ordinary least squares estimator $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ has the **minimum variance** among all unbiased linear estimators. It is thus called the best linear unbiased estimator (BLUE).
- This does not mean this estimator will have the minimum expected loss E_{out} - a biased estimator may have lower E_{out} for finite training data.

Practical workflow

For complex estimators (e.g. neural networks), exhaustive hyperparameter search is unfeasible.
Typical approach:



Regularization

Regularized LLS: add penalty term on $\mathbf{w}$ with suitable norm:

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\| .$$

Purpose:

- **Statistical**: reduce expressiveness of model by reducing fluctuations of $\mathbf{w}$. Allows to control the bias-variance tradeoff via λ.
- **Numerical**: regularized solutions often numerically better behaved.
- **Structural**: e.g., induce sparsity in solution.

Regularization method depends on penalty type:

Regularization type	Penalty term	Prior	Solution methods
Tikhonov regularization Ridge regression	$\|\mathbf{w}\|_2^2$	Normal	Closed form
Lasso regression	$\|\mathbf{w}\|_1$	Laplace	Proximal gradient descent
l_0 regularization	$\|\mathbf{w}\|_0$	-	Forward selection, Backward elimination
Elastic nets	$(1-\alpha)\|\mathbf{w}\|_1 +$ $\alpha\|\mathbf{w}\|_2$	-	Proximal gradient descent

L2 (Ridge) Regularization

We would like to work in high-dimensional feature spaces

$$\mathbf{r}_i \to \mathbf{x}_i = (\phi_1(\mathbf{r}_i), ..., \phi_n(\mathbf{r}_i))^\top .$$

However, this leads to danger of overfitting. To avoid overfitting, we penalize the norm of the solution:

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 ,$$

where λ is a hyperparameter.
Taking derivatives and setting them to zero yields the solution:

$$\begin{aligned} \mathbf{w} &= \left(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} \\ &= \tilde{\mathbf{C}}_{XX}^{-1} \mathbf{C}_{XY} \end{aligned}$$

This is equal to the direct solution of the normal equations, only that we use the so-called shrinkage estimator for the covariance matrix:

$$\tilde{\mathbf{C}}_{XX} = \lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X}$$

Sparsity-inducing Regularization

- **L0 regularization**

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_0 ,$$

  Most extreme way to enforce sparsity. Magnitude of the coefficients of $\mathbf{w}$ does not matter, we only want to minimize the number of non-zero entries. This regularization function is not commonly used in practice, as it is very difficult to solve.

- **L1 regularization**, e.g. using the least absolute selection and shrinkage (LASSO) method.

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1 ,$$

- **Elastic net**

$$\min \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \left[(1-\alpha) \|\mathbf{w}\|_1 + \alpha \|\mathbf{w}\|_2^2 \right],$$

  Where α switches between the two extremes $\alpha = 0$ (L1 regularization) and $\alpha = 1$ (Ridge regression).