Statistical Machine Learning

Pier Francesco Palamara
Department of Statistics
University of Oxford

Slide credits and other course material can be found at:
http://www.stats.ox.ac.uk/~palamara/SML19_BDI.html

Last time: Loss function and risk

- How good is the prediction? We can use a **loss function** $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ to formalize the quality of the prediction.
- Typical loss functions for regression:
  - **Squared loss**
  $$L(Y, f(X)) = (f(X) - Y)^2 \,.$$
  - **Absolute loss**
  $$L(Y, f(X)) = |f(X) - Y| \,.$$

Risk

For a given loss function L, the **risk** R of a learned function f is given by the expected loss

$$R(f) = \mathbb{E}_{P_{XY}} \left[L(Y, f(X)) \right],$$

where the expectation is with respect to the true (unknown) joint distribution of (X, Y).

- The risk is unknown, but we can compute the **empirical risk**:

$$R_N(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i)).$$

Hypothesis space and Empirical Risk Minimization

- Hypothesis space $\mathcal{H}$ is the space of functions f under consideration.
- **Inductive bias**: necessary assumptions on "plausible" hypotheses
- Find best function in the space of hypothesis $\mathcal{H}$ minimizing the risk:

$$f_\star = \operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{X,Y}[L(Y, f(X))]$$

- **Empirical Risk Minimization** (ERM): minimize the empirical risk instead, since we typically do not know $P_{X,Y}$.

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i))$$

- How complex should we allow functions f to be? If hypothesis space $\mathcal{H}$ is "too large", ERM will overfit. Function

$$\hat{f}(x) = \begin{cases} y_i & \text{if } x = x_i, \\ 0 & \text{otherwise} \end{cases}$$

  will have zero empirical risk, but is useless for generalization, since it has simply "memorized" the dataset.

Linear regression: Solution in matrix form

Compact expression

$$R_N(\boldsymbol{\theta}) = ||\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}||_2^2 = \left\{\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} - 2\left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}\right)^{\mathrm{T}}\boldsymbol{\theta}\right\} + \text{const}$$

Gradients of Linear and Quadratic Functions

- $\nabla_{\boldsymbol{x}} \boldsymbol{b}^\top \boldsymbol{x} = \boldsymbol{b}$
- $\nabla_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$ (symmetric $\boldsymbol{A}$)

Normal equation

$$\nabla_{\boldsymbol{\theta}} R_N(\boldsymbol{\theta}) \propto \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = 0$$

This leads to the linear regression solution[1]

$$\boldsymbol{\theta} = \left(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$$

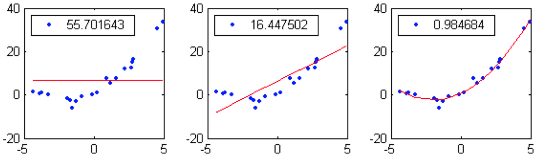
[1] Also see PRML book, Section 3.1.2 for a geometric interpretation.

Nonlinear basis functions

Can we learn non-linear functions? We can use a nonlinear mapping

$$\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^D \to \boldsymbol{z} \in \mathbb{R}^M$$

For instance, we could use polynomials of increasing order, $\phi_k(\boldsymbol{x}_i) = \boldsymbol{x}_i^k$



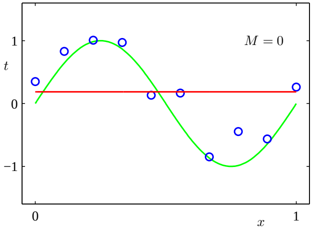
P = 0 P = 1 P = 2

The linear regression solution has a new design matrix

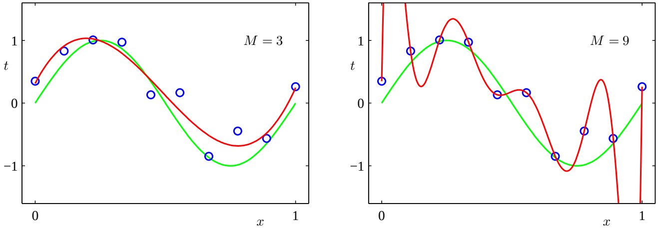
$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1)^{\mathrm{T}} \\ \boldsymbol{\phi}(\boldsymbol{x}_2)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_N)^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{N \times M}, \quad \boldsymbol{\theta}^{\mathsf{LMS}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y}$$

Regression with nonlinear basis functions

Fitting samples from a sine function: **underrfitting** as $f(x)$ is too simple

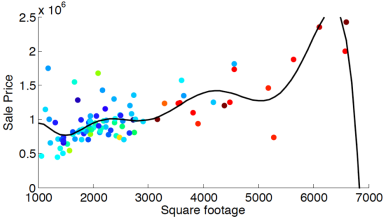


Better fit for higher order, but overfitting as $f(x)$ is too flexible



Overfitting can be quite disastrous

Fitting the housing price data with $M = 7$



Note that the price would go to zero (or negative) if you buy bigger ones!
This is called poor generalization/overfitting.

Validation and Cross-Validation

Generalization

- Generalization ability: what is the out-of-sample (testing) error of the learner f?
- Two important factors determining generalization ability:
  - Model complexity
  - Training data size
- We learn f by minimizing some variant of empirical risk $R_N^{\text{emp}}(f)$- what can we say about the true risk $R(f)$?

Empirical vs True Risk

- In general,

$$R(f) = R_N^{\text{emp}}(f) + \text{overfit penalty.}$$

- Overfit penalty depends on the complexity of the model (also see Vapnik-Chervonenkis, or **VC theory**).
- We will look at two strategies to tune a model's complexity:
  - (Cross-)Validation, where we estimate $R(f)$ to calibrate the model
  - Regularization, where we try to approximate a model's overfit penalty

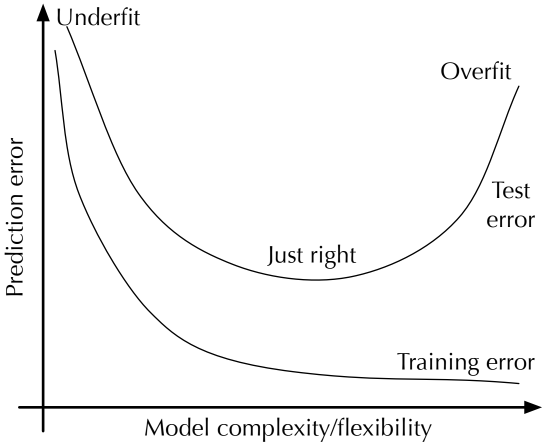
- **testing error** can be obtained by setting aside some of the data.
  - testing error $\neq$ training error.
  - For any example not used in training:

$$\mathbb{E}\left[L\left(y_{\text{test}}, f(x_{\text{test}})\right)\right] = R(f).$$

  - But for examples used in training:

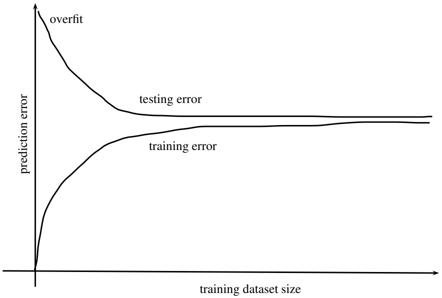
$$\mathbb{E}\left[L\left(y_{\text{train}}, f(x_{\text{train}})\right)\right] \neq R(f).$$

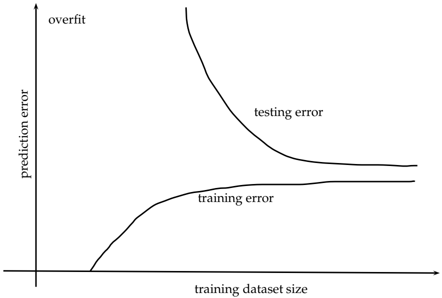
Learning Curves



Fixed dataset size, varying model complexity.

Learning Curves

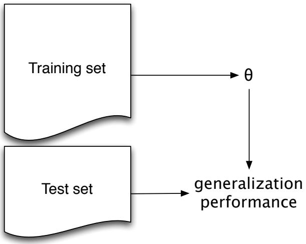




Fixed model complexity, varying dataset size.
Two models: one simple, one complex. Which is which?

Optimizing Tuning Parameters

- How can we optimize generalization ability, via optimizing choice of tuning parameters, model size, and learning parameters?
- Suppose we have split data into training/test set.
- Test set can be used to determine generalization ability, and used to choose best setting of tuning parameters/model size/learning parameters with best generalization.
- Once these tuning parameters are chosen, still important to determine generalization ability, but cannot use performance on test set to gauge this anymore!



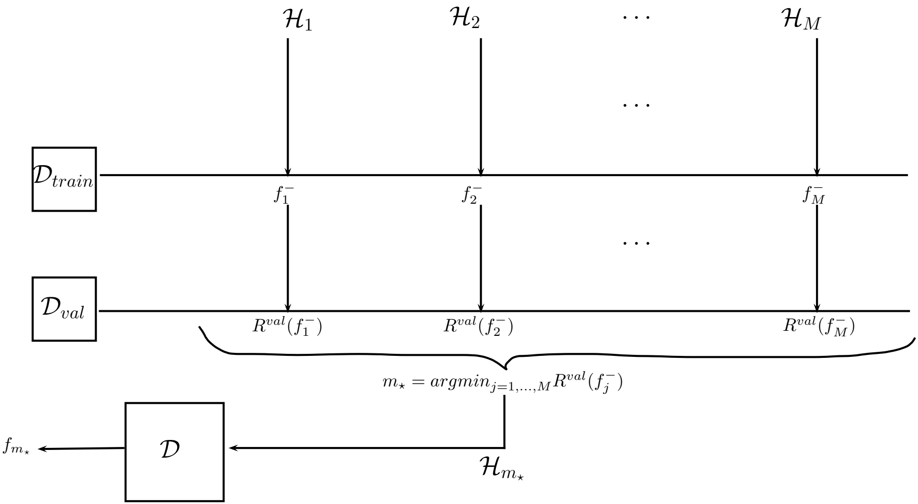
Validation error

- Idea: split data into 3 sets: training set, test set, and **validation set**.
- **Out-of-sample average loss**. For a dataset $\{\tilde{x}_i, \tilde{y}_i\}_{i=1}^{v}$ unseen in training

$$R^{\mathsf{val}}(f) = \frac{1}{v}\sum_{i=1}^{v} L\left(\tilde{y}_i, f(\tilde{x}_i)\right)$$

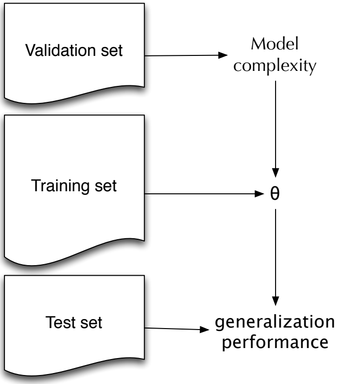
- $\mathbb{E}\left[R^{\mathsf{val}}(f)\right] = R(f)$, $\mathsf{Var}\left[R^{\mathsf{val}}(f)\right] \asymp \frac{1}{v}$, i.e. $R^{\mathsf{val}}(f) = R(f) \pm \mathcal{O}\left(\frac{1}{\sqrt{v}}\right)$
- Just like testing error so far.
- It becomes validation error only once it is used to **change our learning**.

Validation


$\mathcal{H}_1$
$\mathcal{H}_2$
$\cdots$
$\mathcal{H}_M$
$\mathcal{D}_{train}$
f_1^-
f_2^-
f_M^-
$\mathcal{D}_{val}$
$R^{val}(f_1^-)$
$R^{val}(f_2^-)$
$R^{val}(f_M^-)$
$m_\star = argmin_{j=1,\ldots,M} R^{val}(f_j^-)$
$\mathcal{D}$
$f_{m_\star}$
$\mathcal{H}_{m_\star}$

Validation

- For each combination of tuning parameters γ:
  - Train our model on the training set, fit parameters $\theta = \theta(\gamma)$, obtaining decision function $f_{\theta(\gamma)}$.
  - Evaluate $R^{\text{val}}\left(f_{\theta(\gamma)}\right)$ average loss on a validation set.
- Pick γ^* with best performance on validation set.
- Using γ^*, train on both training and validation set to obtain the optimal θ^*.
- $R^{\text{val}}(f_{\theta(\gamma^*)})$ is now a biased estimate of $R(f_{\theta(\gamma^*)})$ and can be overly optimistic!
- Evaluate model with γ^*, θ^* on test set, reporting generalization performance.



Bias introduced by validation

- **Example**: Selecting between two equally bad classifiers f_1 and f_2:

$$R(f_1) = R(f_2) = 0.5.$$

- Assume that we have independent unbiased estimators $\mathsf{R}_1 = R^{\text{val}}(f_1)$, $\mathsf{R}_2 = R^{\text{val}}(f_2)$, both uniform on $[0, 1]$
- Learning rule $f_\star$ chosen to minimize R^{val} is either f_1 or f_2, so also equally bad.
- But $\mathbb{E}\min(\mathsf{R}_1, \mathsf{R}_2) = \frac{1}{3}$ (since $\mathbb{E}\min(\{U_{[0,1]}\}_{i=1}^{n}) = (n+1)^{-1}$), so in terms of validation error it may appear that we are getting an improvement!

Validation error and Generalization

How contaminated are different parts of data in terms of being able to tell us something about generalization ability?

- Training data: fully contaminated, used in learning - $R^{\text{emp}}(f)$ is usually far from $R(f)$ (unless the model is too simple for the amount of data).
- Validation data: partly contaminated, used in model selection / meta-learning - $R^{\text{val}}(f)$ is biased, but still useful, provided that:
  - we have a large enough validation set size v
  - we do not use it to select from a large number M of models
  - Typically,

$$R(f) \leq R^{\text{val}}(f) + \underbrace{\mathcal{O}\left(\sqrt{\frac{\log M}{v}}\right)}_{\text{overfit penalty of the meta-model}}$$

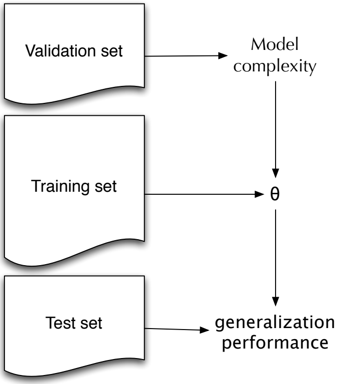
- Test data: clean, not used for any part of learning.

Size of validation set?

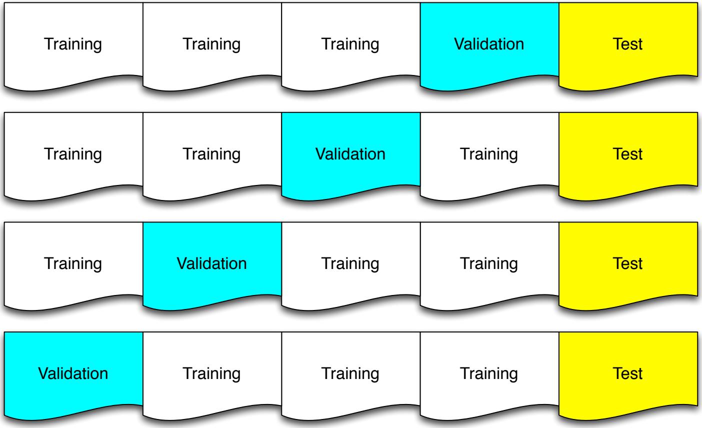
- In practice, there is just one dataset! If v is used for computing validation error, then only $n - v$ used for training.
  - Small v : $R^{\text{val}}(f^-)$ is a bad estimate of $R(f^-)$
  - Large v : $R^{\text{val}}(f^-)$ is a reliable estimate of a much worse risk (f^- learned on much less data than f)!
- We are using:

$$R(f) \underset{\text{(need small } v)}{\approx} R(f^-) \underset{\text{(need large } v)}{\approx} R^{\text{val}}(f^-)$$

- Wasteful to split into 3 subsets.
- Different approach: **cross-validation**.



Cross-Validation



Cross-Validation

- Basic approach:
  - Split training set into T folds.
  - For each γ and each $t = 1, \ldots, T$:
    - Use fold t as validation set and the rest to train the model parameters $\theta_t(\gamma)$, obtaining trained learner $f^-_{t,\gamma}$.

$$R^{\text{val}}_t(f^-_{t,\gamma}) = \frac{1}{|\text{Fold}(t)|} \sum_{i \in \text{Fold}(t)} L(y_i, f^-_{t,\gamma}(x_i))$$

  - Choose γ^* which minimizes validation error averaged over folds

$$\frac{1}{T} \sum_{t=1}^{T} R^{\text{val}}_t(f^-_{t,\gamma})$$

  - Train model with tuning parameter γ^* on all training set to obtain f_{γ^*}.
  - Report generalization performance on test set.
- **Leave-One-Out (LOO)** cross validation: one data item per fold, i.e., $T = n$.

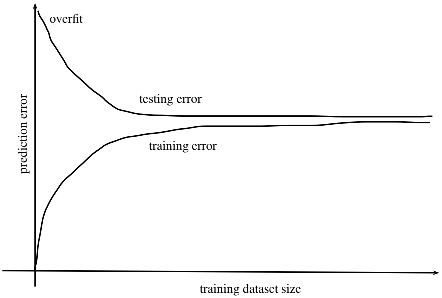
Cross-validation can be computationally expensive ($T\times$ increase in complexity).

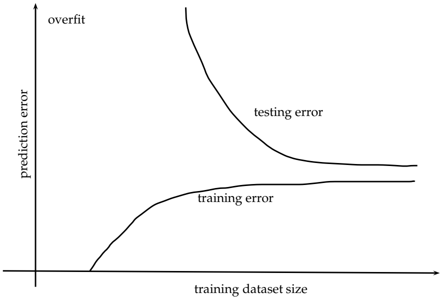
Leave-One-Out Cross-Validation

Leave-one-out (LOO) cross validation: one data item per fold, i.e., $T = n$.

- Since only one data item not used in training, $R(f^-_{t,\gamma})$ are all very close to $R(f_\gamma)$ (small v benefit).
- Thus,
  $$\frac{1}{n}\sum_{t=1}^{n} R_t^{\mathsf{val}}(f^-_{t,\gamma}) = \frac{1}{n}\sum_{t=1}^{n} L(y_t, f^-_{t,\gamma}(x_t))$$
  has a small variance (large v benefit).
- All examples for validation and all examples for training.
- summands are no longer independent

Learning Curves





Fixed model complexity, varying dataset size.
Two models: one simple, one complex. Which is which?

Bias-Variance Tradeoff

- Where does the prediction error come from?
- Example: Squared loss in regression: $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \mathbb{R}$,

$$L(Y, f(X)) = (Y - f(X))^2$$

- Optimal f is the conditional mean:

$$f_*(x) = \mathbb{E}\left[Y|X = x\right]$$

- Follows from $R(f) = \mathbb{E}_X \mathbb{E}\left[\left(Y - f\left(X\right)\right)^2 \middle| X\right]$ and

$$\begin{aligned}
&\mathbb{E}\left[\left(Y - f\left(X\right)\right)^2 \middle| X = x\right] \\
=\ &\mathbb{E}\left[Y^2\middle| X = x\right] - 2f\left(x\right)\mathbb{E}\left[Y\middle| X = x\right] + f\left(x\right)^2 \\
=\ &\text{Var}\left[Y|X = x\right] + \left(\mathbb{E}\left[Y\middle| X = x\right] - f(x)\right)^2.
\end{aligned}$$

Bias-Variance Tradeoff

- Optimal risk is the intrinsic conditional variability of Y (noise):

$$R(f_*) = \mathbb{E}_X \left[\text{Var}\left[Y|X\right]\right]$$

- **Excess risk**:

$$\begin{aligned} R(f) - R(f_*) &= \mathbb{E}_X \left[\text{Var}\left[Y|X\right] + (f_*(X) - f(X))^2 - \text{Var}\left[Y|X\right]\right] \\ &= \mathbb{E}_X \left[(f(X) - f_*(X))^2\right] \end{aligned}$$

- How does the excess risk behave **on average**?
- Consider a randomly selected dataset $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^n$ and $f^{(\mathcal{D})}$ trained on $\mathcal{D}$ using a model (hypothesis class) $\mathcal{H}$.

$$\begin{aligned} \mathbb{E}_{\mathcal{D}}\left[R(f^{(\mathcal{D})}) - R(f_*)\right] &= \mathbb{E}_{\mathcal{D}}\mathbb{E}_X\left[\left(f^{(\mathcal{D})}(X) - f_*(X)\right)^2\right] \\ &= \mathbb{E}_X\mathbb{E}_{\mathcal{D}}\left[\left(f^{(\mathcal{D})}(X) - f_*(X)\right)^2\right]. \end{aligned}$$

Bias-Variance Tradeoff

- Denote $\bar{f}(x) = \mathbb{E}_{\mathcal{D}} f^{(\mathcal{D})}(x)$ (average decision function over all possible datasets)

$$\mathbb{E}_{\mathcal{D}}\left[\left(f^{(\mathcal{D})}(X) - f_*(X)\right)^2\right] = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(f^{(\mathcal{D})}(X) - \bar{f}(X)\right)^2\right]}_{\text{Var}_X(\mathcal{H},n)} + \underbrace{\left(\bar{f}(X) - f_*(X)\right)^2}_{\text{Bias}^2_X(\mathcal{H},n)}$$

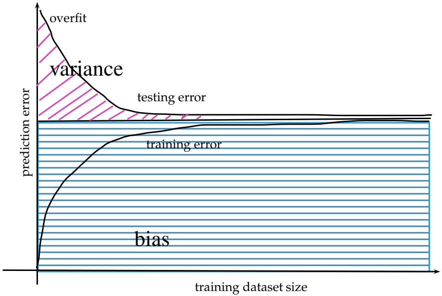
Now,

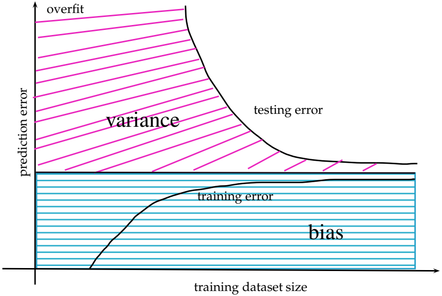
$$\mathbb{E}_{\mathcal{D}} R(f^{(\mathcal{D})}) = R(f_*) + \mathbb{E}_X \text{Var}_X(\mathcal{H}, n) + \mathbb{E}_X Bias^2_X(\mathcal{H}, n)$$

Where does the prediction error come from?

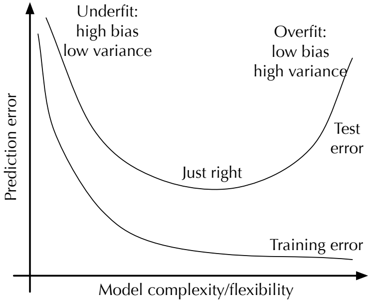
- **Noise**: Intrinsic difficulty of regression problem.
- **Bias**: How far away is the best learner in the model (average learner over all possible datasets) from the optimal one?
- **Variance**: How variable is our learning method if given different datasets?

Learning Curves





Building models to trade bias with variance



- Building a machine learning model involves trading between its bias and variance. We will see many examples in the next lectures:
  - Bias reduction at the expense of a variance increase: building more complex models, e.g. adding nonlinear features and additional parameters, increasing the number of hidden units in neural nets, using decision trees with larger depth, decreasing the **regularization** parameter.
  - Variance reduction at the expense of a bias increase: early stopping, using k-nearest neighbours with larger k, increasing the **regularization** parameter.