Overfitting and Underfitting

Functional Programming and Intelligent Algorithms

Que Tran

Høgskolen i Ålesund

13th March 2017

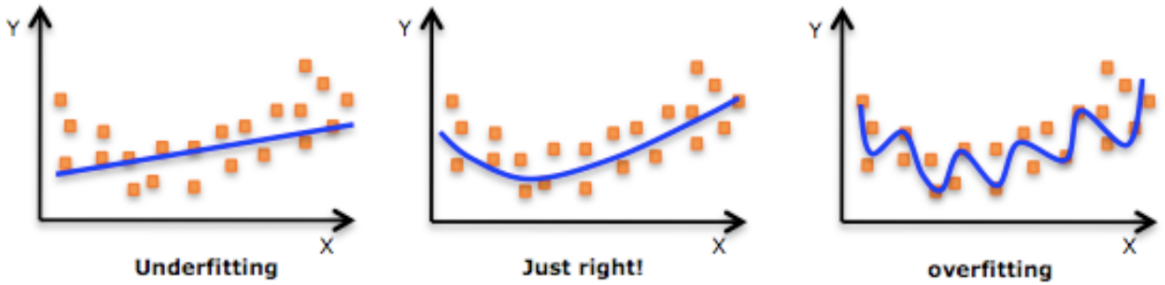
Underfitting and overfitting

— *Underfitting*: model cannot capture the underlying trend of the data.

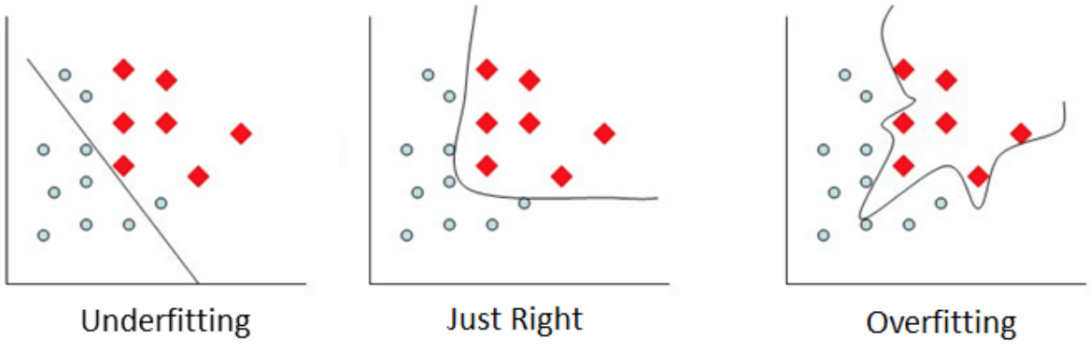
— *Overfitting*: model captures the noise of the data.

Illustration of underfitting and overfitting

Regression



Classification

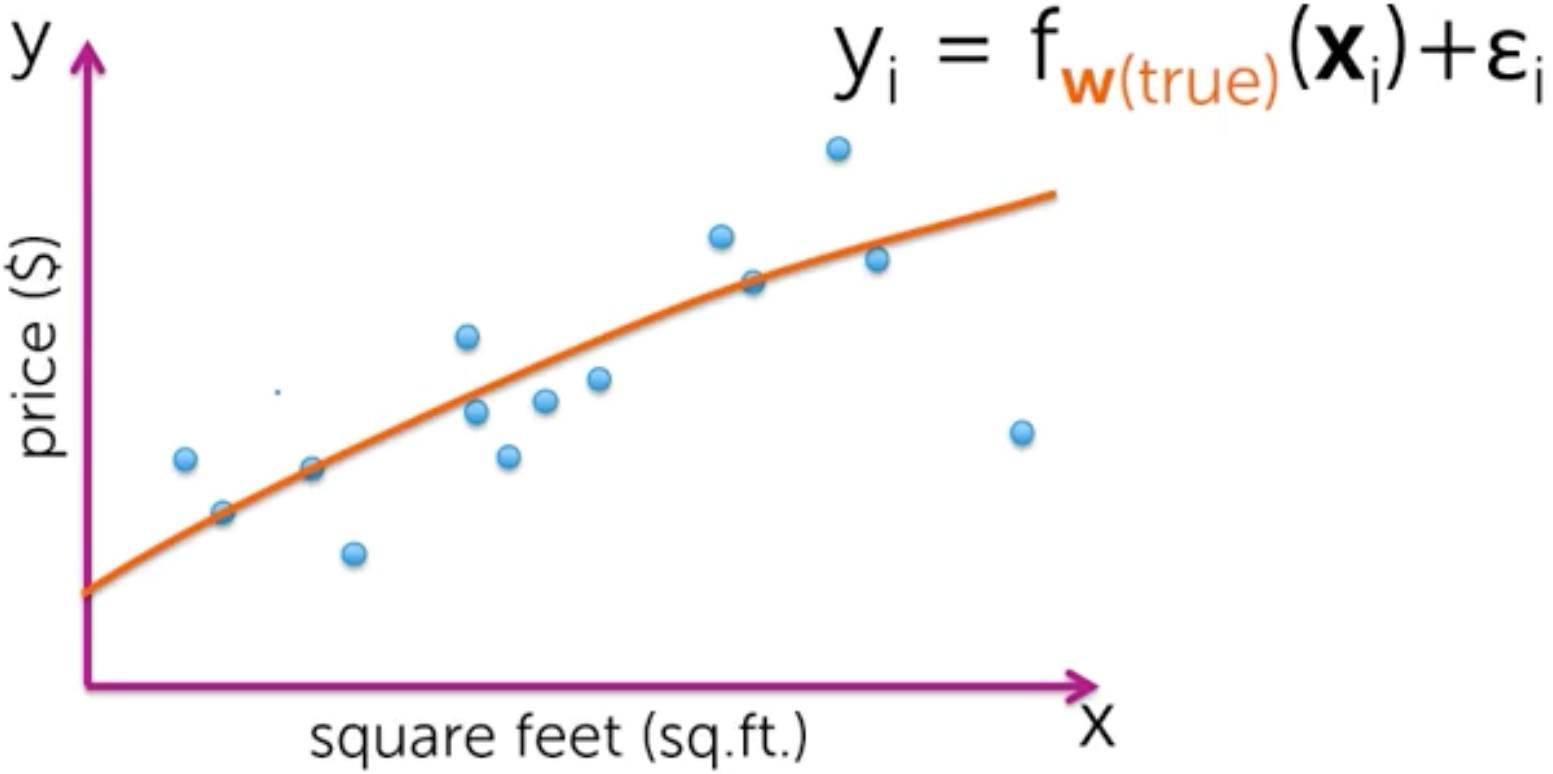


3 sources of error

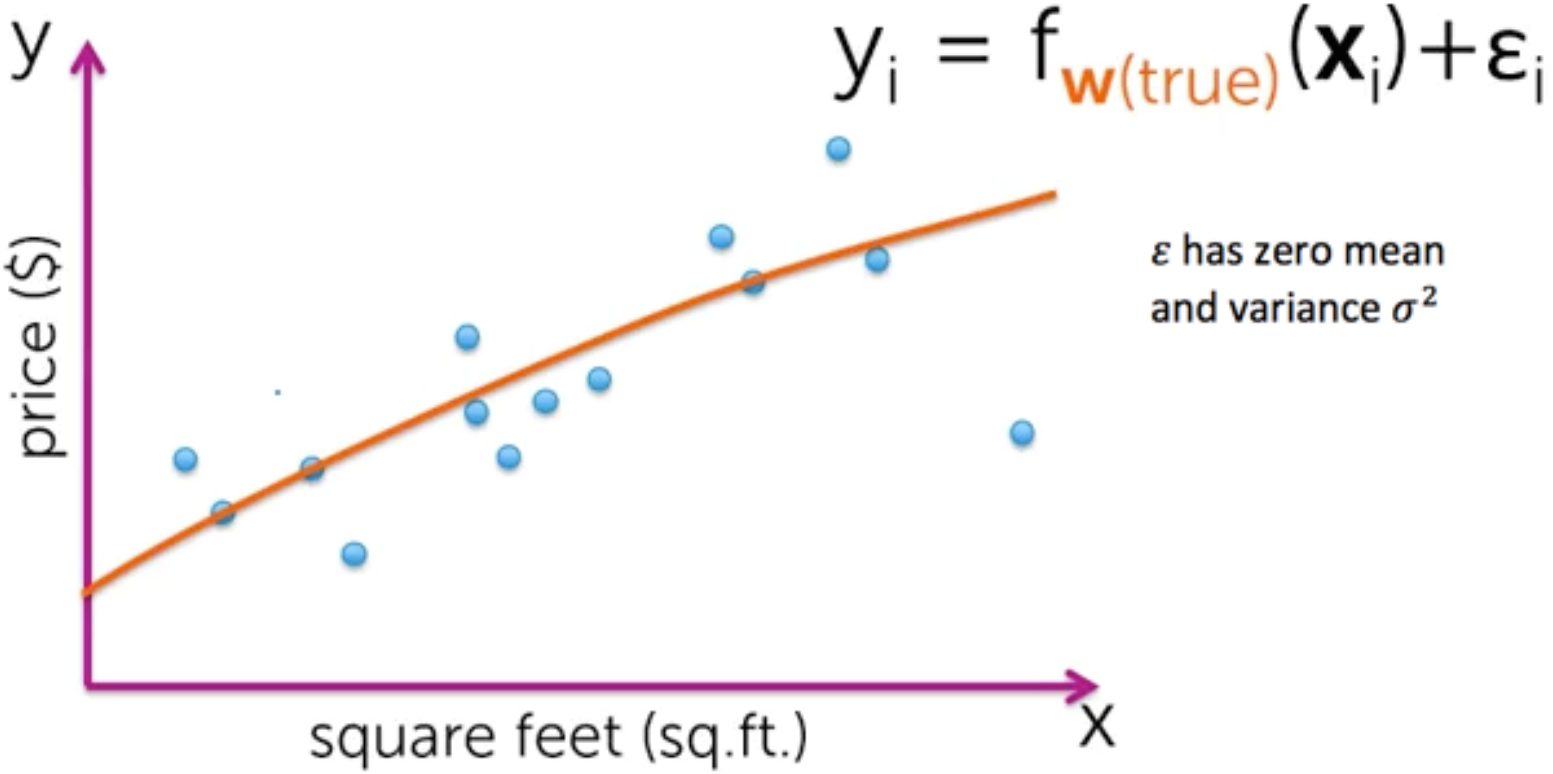
There are 3 sources of error:

- Noise
- Bias
- Variance

Noise



Noise


y
price ($)
square feet (sq.ft.)
x
$y_i = f_{w(true)}(x_i) + \varepsilon_i$
ε has zero mean
and variance σ^2

Bias-variance tradeoff

- Bias: is a measure of how far off the model estimated values are from the true values. It is the error from erroneous assumptions in the learning algorithm.
- Variance: refers to the change in parameter estimates across different data sets. It is the error from sensitivity to small fluctuations in the training set.

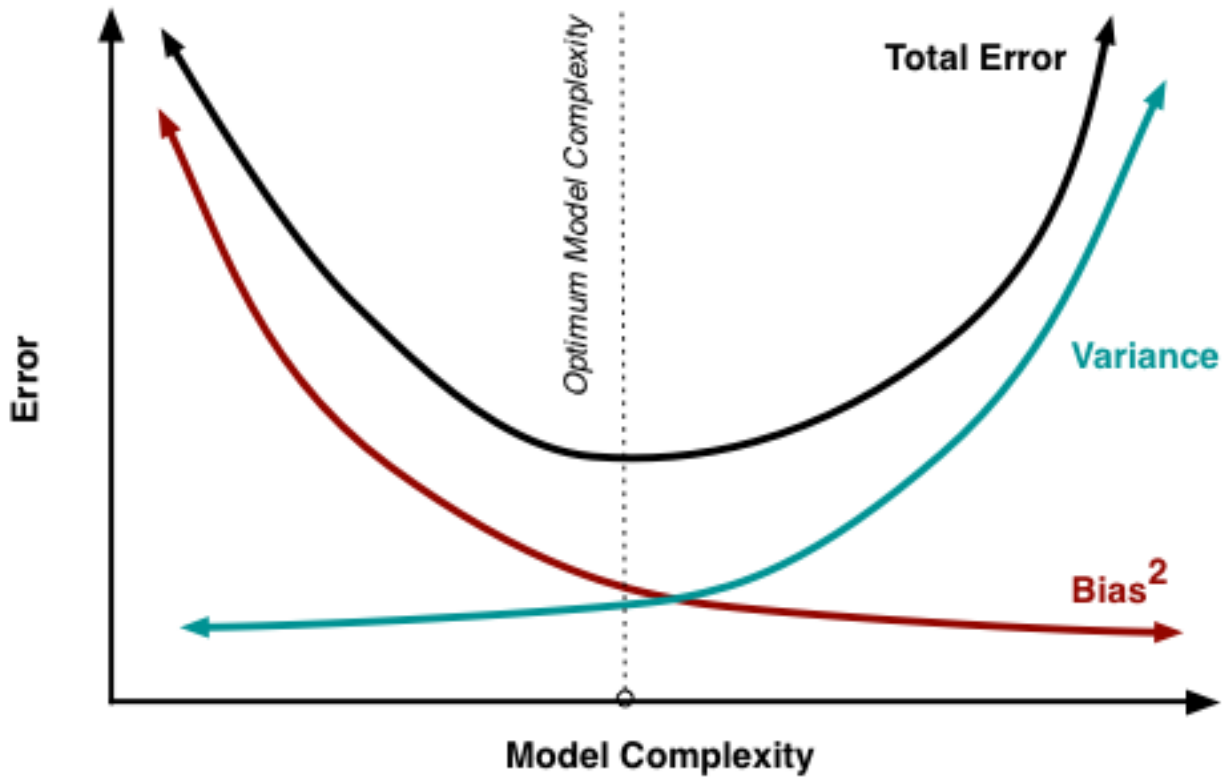
Bias-variance tradeoff

— Bias-variance decomposition of squared error:
Target function: $y = f(x) + \epsilon$ where the noise, ϵ, has zero mean and variance σ^2.

$$
\begin{aligned}
&\textit{Expected\_prediction\_error} \\
&= E\left[(y - \hat{f}(x))^2\right] \\
&= E\left[\hat{f}(x) - f(x)\right]^2 + \left(E\left[\hat{f}(x)^2\right] - E\left[\hat{f}(x)\right]^2\right) + \sigma_e^2 \\
&= \textit{Bias}^2 + \textit{Variance} + \textit{IrreducibleError}
\end{aligned}
$$

— Prove?

Bias-variance tradeoff



Underfitting and overfitting

- *Underfitting*: model cannot capture the underlying trend of the data, **low variance** but **high bias**.
- *Overfitting*: model captures the noise of the data, **high variance** but **low bias**.

Underfitting and overfitting in NN

Performance on ...	Underfitting	Overfitting
Training set	Bad	Good
Testing set	Bad	Bad

Underfitting	Overfitting
Too few epochs	Too many epochs
Too few nodes	Too little training data

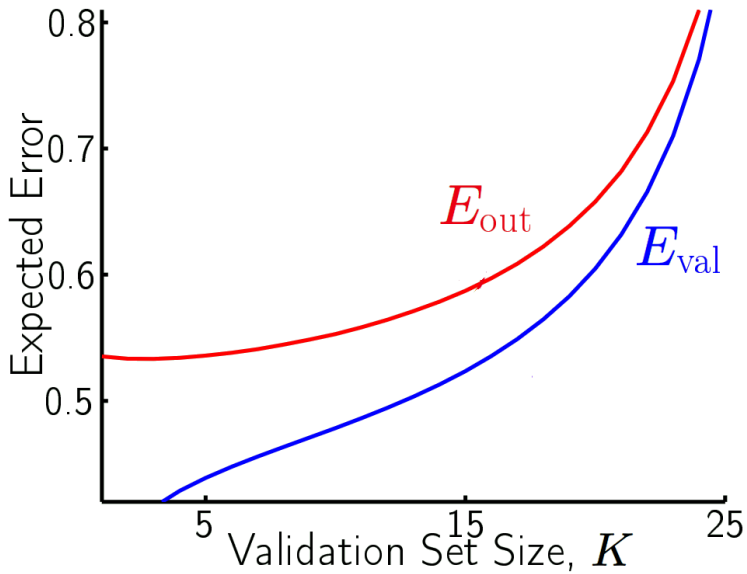
Avoid underfitting/overfitting

- Retraining neural networks
- Early stopping

Retraining neural networks

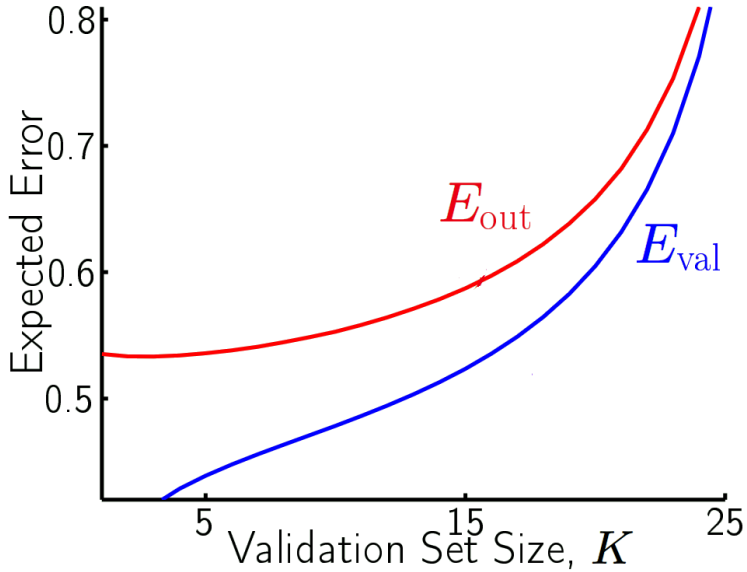
— Split the data (training set, validation set, testing set)
— Train with different configurations and parameters
— Test each configuration on the validation set
— Choose the design based on tests with the validation set

Validation set size

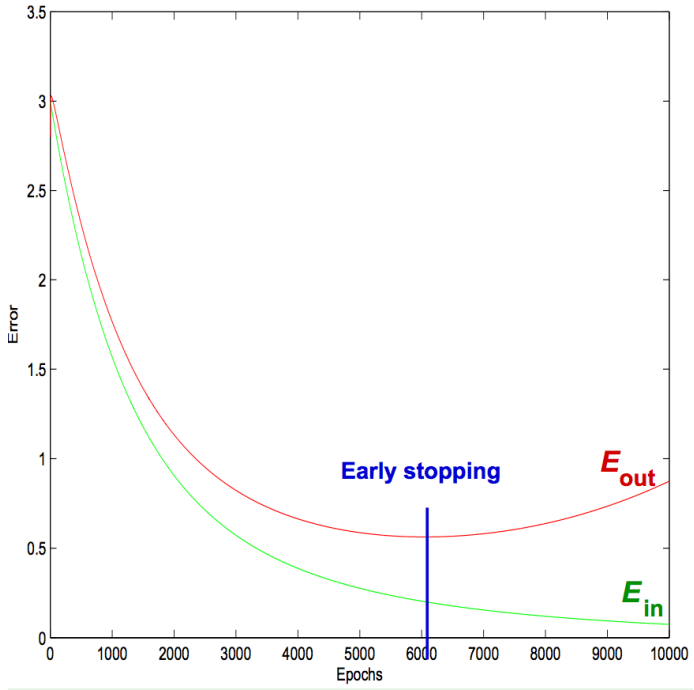


Validation set size

Rule of thumb:
$K = \frac{N}{5}$
K: validation set size
N: Data set size



Early stopping



- Split the training data into a training set and a validation set.
- Train only on the training set and evaluate on the validation set once in a while, e.g. after every fifth epoch.
- Stop training when the error on the validation set is higher than it was the last time(s) it was checked.
- Use the weights the network had in that previous step as the result of the training run.