5. Generalization and regularization

Outline

Model selection

- two models, A and B, are learned on the same training dataset.
- model A has an error of 0.1 on the training set and model B has an error of 1.0.
- which model is better?

Model selection

- two models, A and B, are learned on the same training dataset.
- model A has an error of 0.1 on the training set and model B has an error of 1.0.
- which model is better?
- answer: unknown.
- training error is not a good metric for comparing and selecting models

Test error

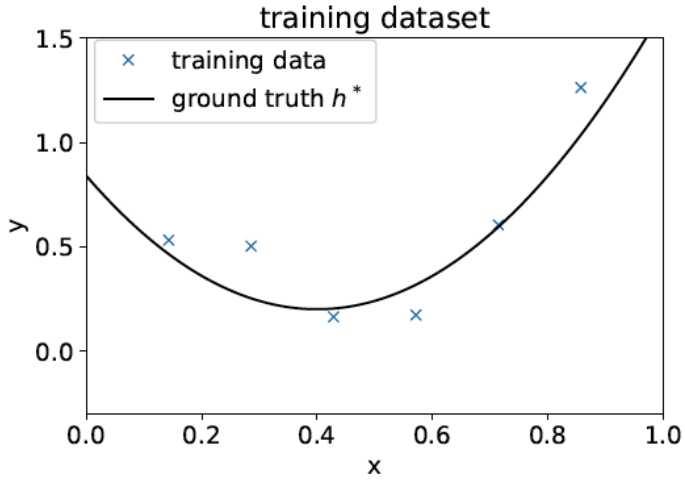
- to compare models, we need to evaluate them on a test set
- the error on the test set is called the **test error**
- measures whether the model generalizes to well to unseen data
- the ultimate goal of machine learning is to minimize the test error, not the training error.
- minimizing the training error is merely an approach towards the goal.
- reducing the training error does not necessarily always reduce the test error
- can be decomposed into three components: bias, variance, and irreducible error

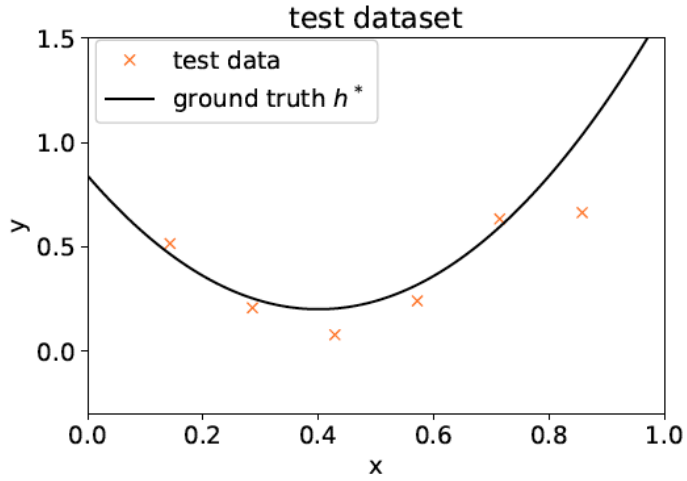
Underfits and overfits

- underfits
  - when both the training and test errors are high
  - cannot make accurate predictions on the training set
  - model being too simple to capture the underlying structure of the data
- overfits
  - when the training error is much lower than the test error
  - make accurate predictions on the training set but not on the test set
  - model being too flexible and captures noise in the training data
- both are related to the bias-variance decomposition of the test error

Bias-variance tradeoff

- an example of regression from `https://cs229.stanford.edu/main_notes.pdf`
- the ground true: $y^{(i)} = h^*(x^{(i)}) + \xi^{(i)}$
- h^* is a quadratic function and $\xi^{(i)} \sim N(0, \sigma^2)$ is the noise.

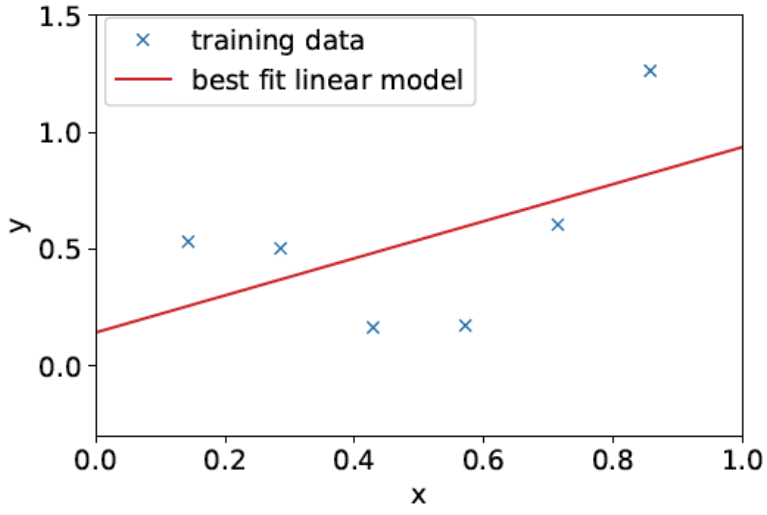


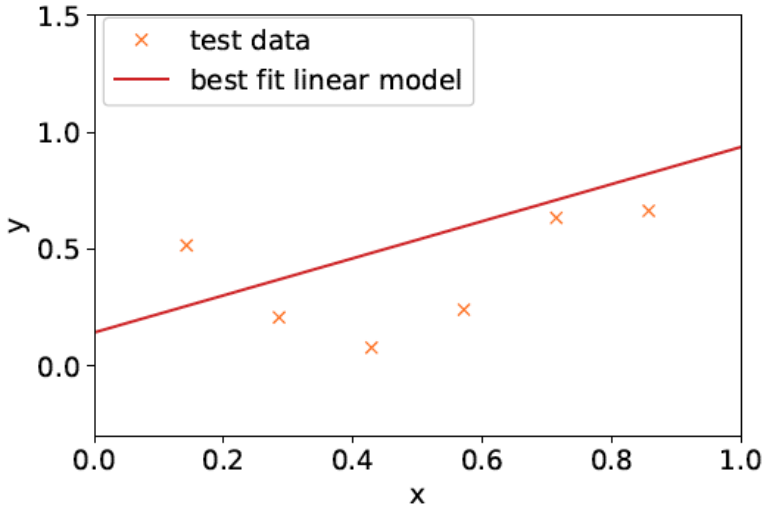


- goal: learn a model $h(x)$ to approximate h^* using training data

Underfits

- fit a linear model with limited noisy data

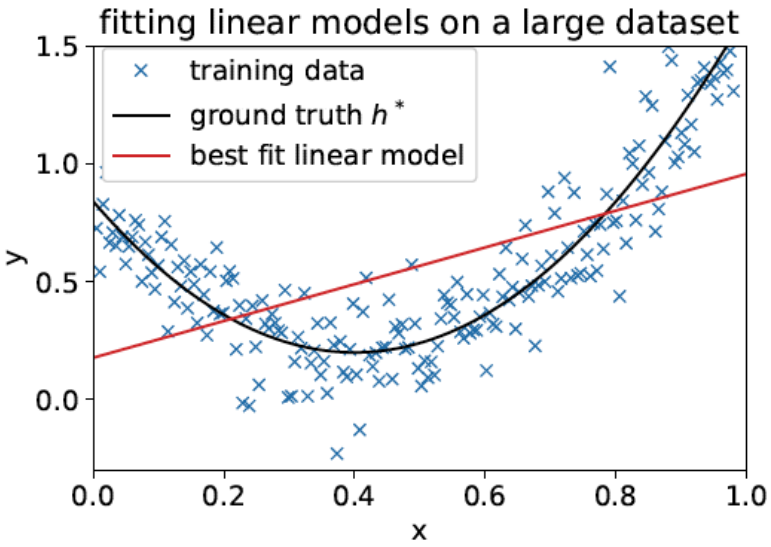


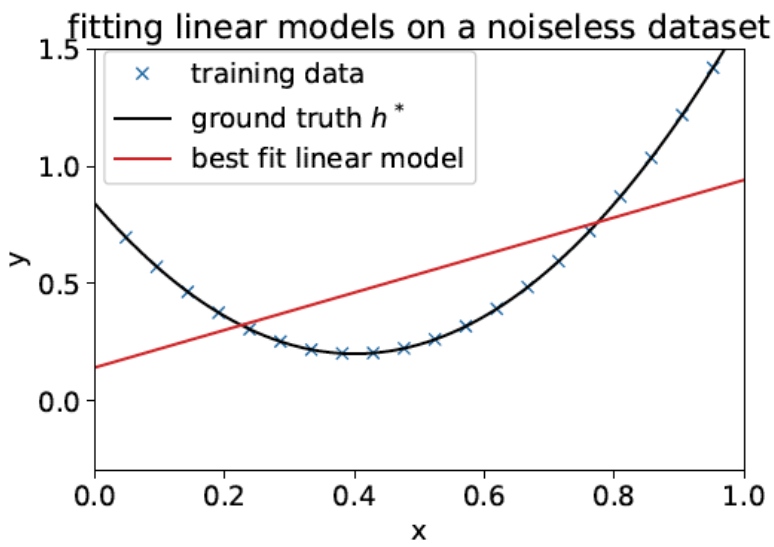


- both training and test errors are large

Underfits

- fit a linear model with more or noiseless data

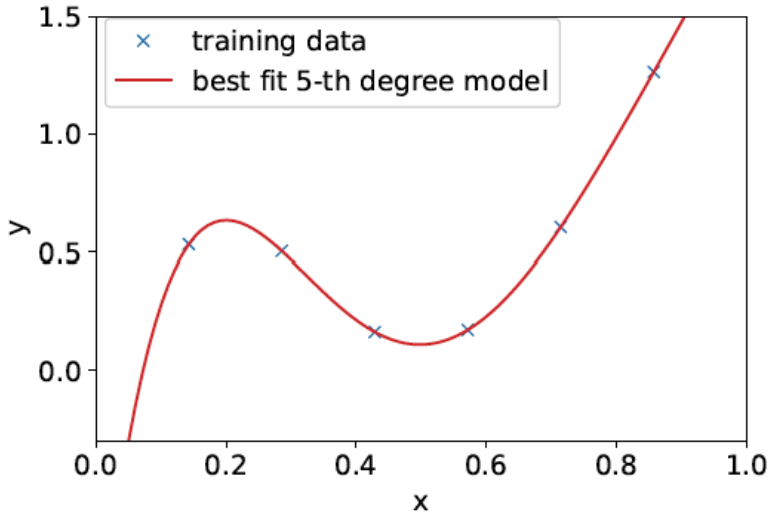


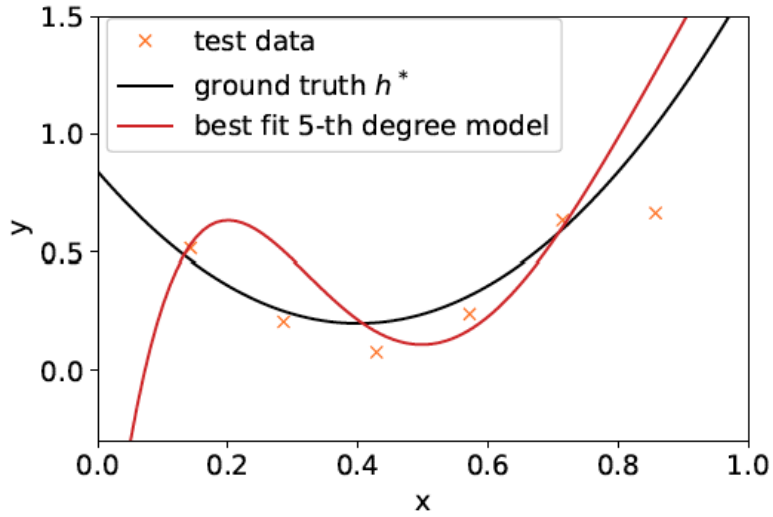


- using more training data does not help reduce either error
- the **bias** of a model is the test error when the model is trained on a very (infinitely) large training set
- models that underfit the data have high bias

Overfits

- fit a 5-th degree polynomial with noisy data

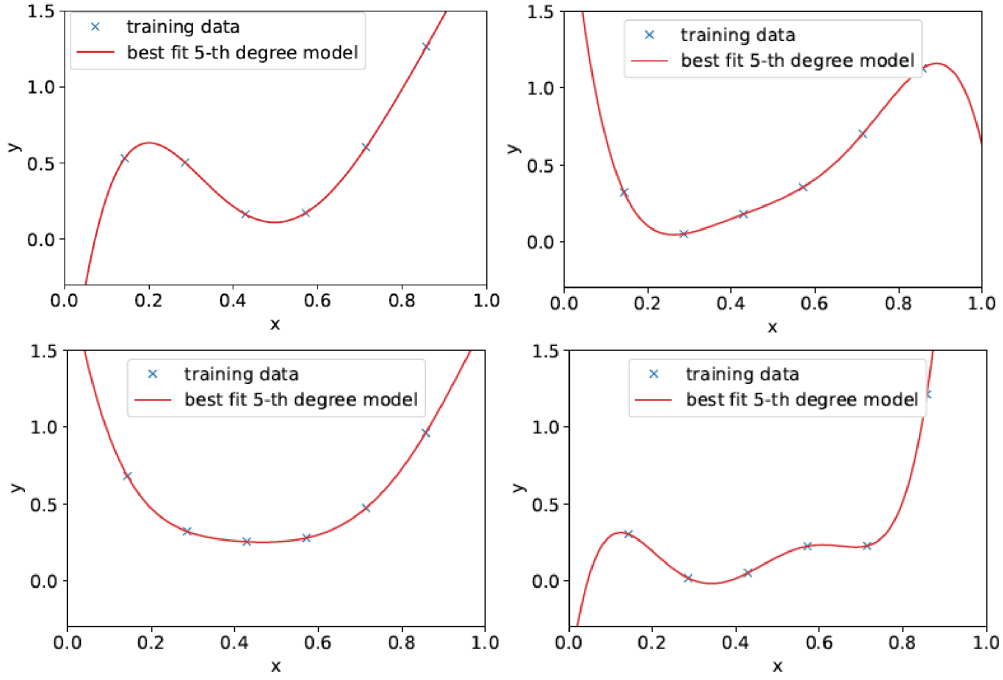




- very small (zero) training error but large test error
- the model is so flexible that it even fits the patterns in training data that is due to noise

Overfits

- fit a 5-th degree polynomial on different training sets

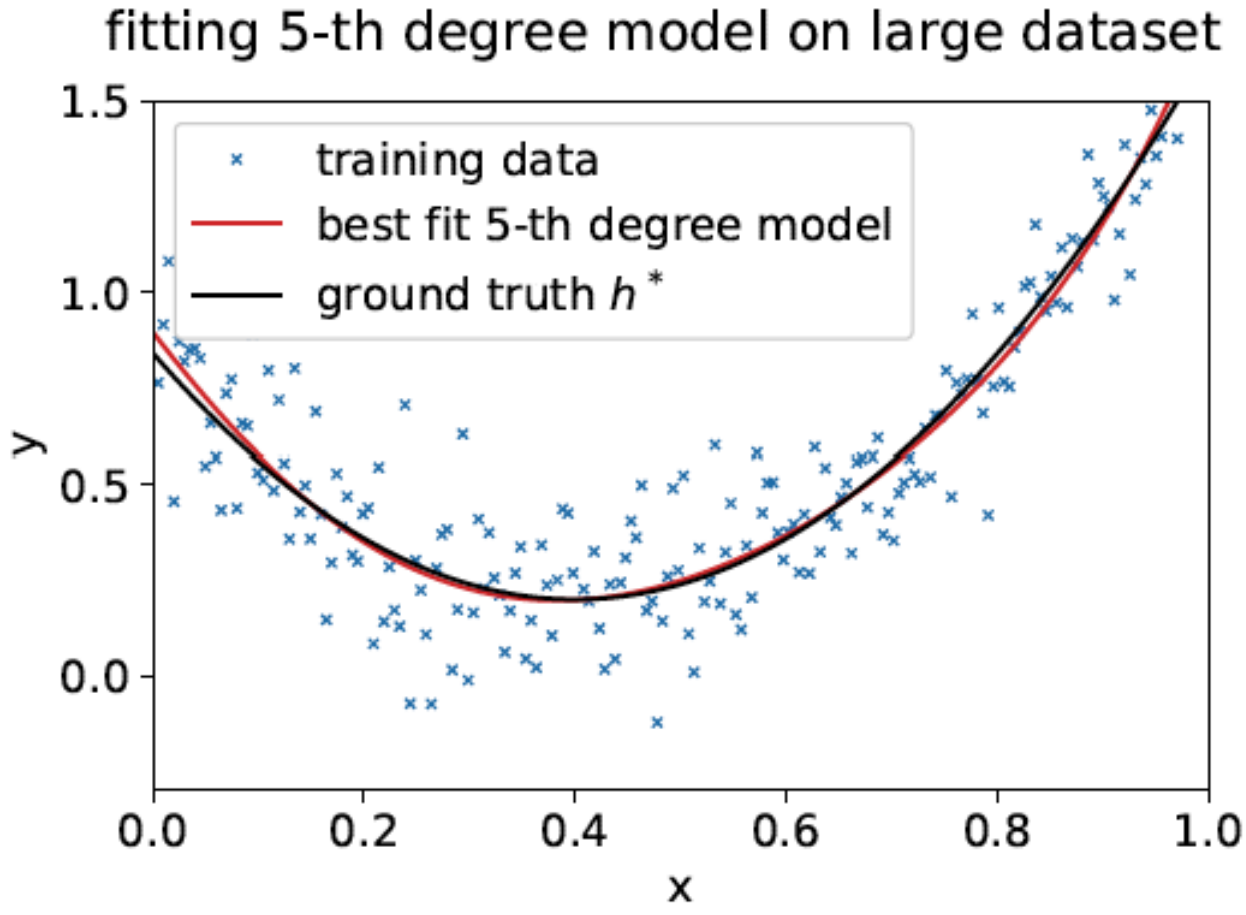


- the model fits the noise in the training set, but the noise could be different in different training sets

- the **variance** of a model is the amount of variations across models trained on different training sets

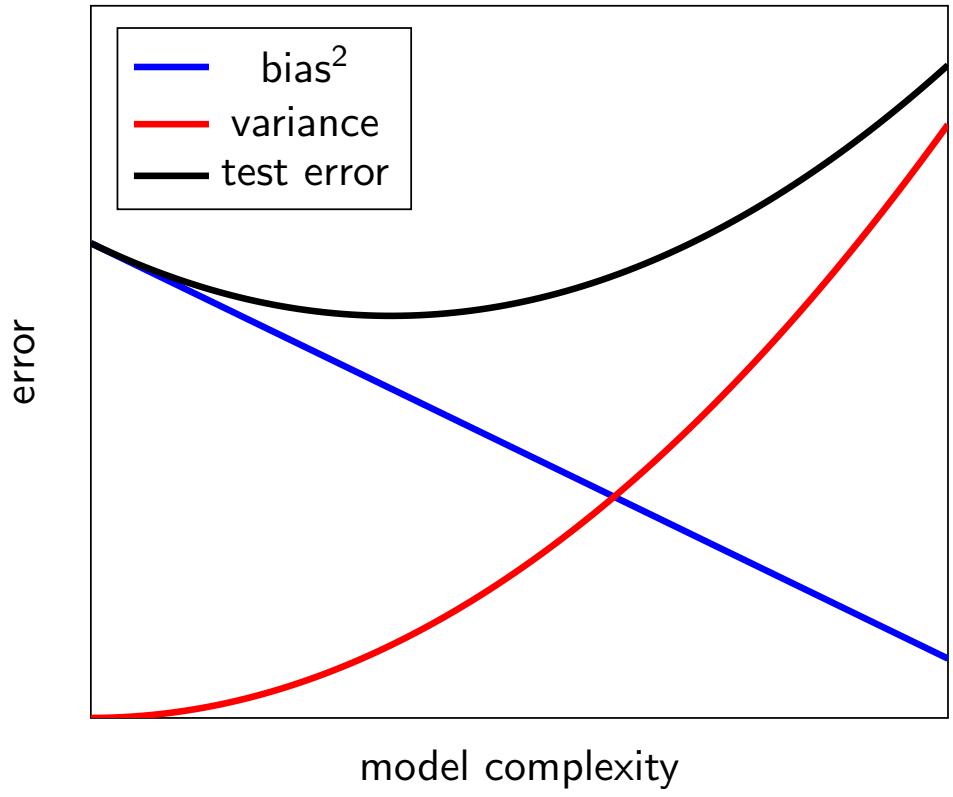
Overfits

- fit a 5-th degree polynomial with more data



- large training set helps reduce the variance of the model

Bias-variance tradeoff



The bias-variance decomposition for regression

- Draw a training dataset $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$ such that $y^{(i)} = h^{\star}(x^{(i)}) + \xi^{(i)}$ where $\xi^{(i)} \in N(0, \sigma^2)$
- Train a model on the dataset S, denoted by $\hat{h}_S$.
- Take a test example (x, y) such that $y = h^{\star}(x) + \xi$ where $\xi \sim N(0, \sigma^2)$,
- the expected test error (averaged over the random draw of the training set S and the randomness of ξ):

$$\text{MSE}(x) = \mathbb{E}_{S,\xi}\left[(y - h_S(x))^2\right]$$

The bias-variance decomposition

- conceptually useful for understanding what contributes to test error

$$
\begin{aligned}
\mathsf{MSE}(x) &= \mathbb{E}\left[(y - \hat{h}_S(x))^2\right] \\
&= \mathbb{E}\left[\left(\xi + \left(h^\star(x) - \hat{h}_S(x)\right)\right)^2\right] \\
&= \mathbb{E}\left[\xi^2\right] + \mathbb{E}\left[\left(h^\star(x) - \hat{h}_S(x)\right)^2\right] \\
&= \sigma^2 + \mathbb{E}\left[\left(h^\star(x) - \hat{h}_S(x)\right)^2\right] \\
&= \underbrace{\sigma^2}_{\text{unavoidable}} + \underbrace{\left(h^\star(x) - h_{\mathsf{avg}}(x)\right)^2}_{\triangleq \text{ bias}^2} + \underbrace{\mathbb{E}\left[\left(h_{\mathsf{avg}}(x) - \hat{h}_S(x)\right)^2\right]}_{\triangleq \text{ variance}}
\end{aligned}
$$

- in practice, the bias and variance are not directly computable

Model selection in practice

- in practice, we do not have access to the true underlying function $h^\star$
- when training data is limited, we cannot estimate $h_{\text{avg}}(x)$ or $\text{var}(h_S(x))$ accurately
- the bias-variance decomposition is a conceptual tool for understanding the test error
- there are more practical ways to estimate the test error and select models

Model selection in practice

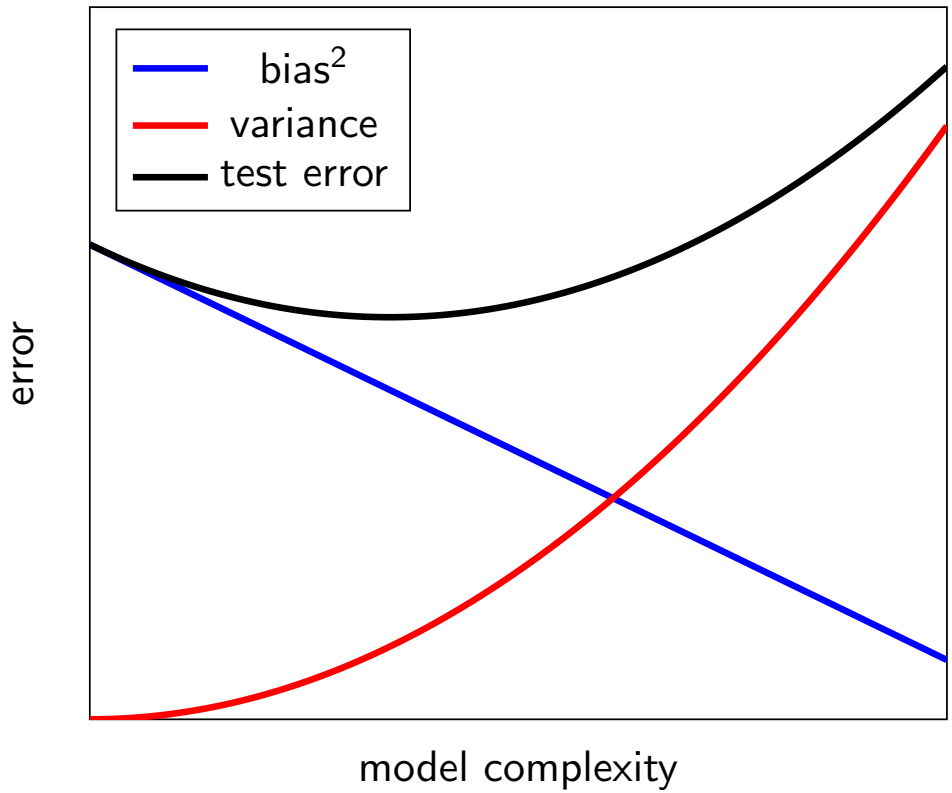
- the most common approach is to split the dataset into training, validation, and test sets

- the training set is used to train models
- the validation set is used to estimate the test error and select models
- the test set is used to evaluate the final model; should be kept in a "vault" and be brought out only at the end of evaluating the model
- if the test set is used repeatedly to select models with smallest test error, the test error of the final chosen model will underestimate the true test error

Outline

Bias-variance tradeoff

- find the right balance between bias and variance



Model complexity

- intuitively, model complexity could be measured by the number of parameters, such as the degree of a polynomial, the number of layers and nodes in a neural network etc.
- model complexity is better measured by functions of the parameters, such as the norm of the parameters
- regularization is a technique to control the complexity of the model and prevent overfitting

Regularization

- regularization is a technique to prevent overfitting by adding a penalty term to the loss function
- the regularized loss function $J_\lambda(\theta)$

$$J_\lambda(\theta) = J(\theta) + \lambda R(\theta)$$

- $J(\theta)$ is the original loss function
- $\lambda \geq 0$ is the regularization parameter
- $R(\theta)$ is a non-negative function of the parameters measuring the complexity of the model

Regularization

- regularized loss function

$$J_\lambda(\theta) = J(\theta) + \lambda R(\theta)$$

- the regularized loss function is optimized instead of the original loss function
- the regularization parameter λ controls the tradeoff between the original loss and the regularization term
- when $\lambda = 0$, the regularized loss is the same as the original loss
- when $\lambda \to \infty$, the regularized loss is dominated by the regularization term

ℓ_2 regularization

- ℓ_2 regularization is the most common form of regularization
- the regularization term is the squared norm of the parameters

$$R(\theta) = \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

- also known as weight decay

$$\begin{aligned} \theta &\leftarrow \theta - \eta \nabla J_\lambda(\theta) \\ &= \theta - \eta\lambda\theta - \eta \nabla J(\theta) \\ &= \underbrace{(1 - \lambda\eta)\theta}_{\text{decaying weights}} - \eta \nabla J(\theta) \end{aligned}$$

- it penalizes large weights and pushes the weights towards zero

ℓ_2 regularization

- most optimization libraries have built-in support for ℓ_2 regularization

```
torch.optim.SGD(params, lr=0.001, momentum=0, ...,
weight_decay=0, ...)

torch.optim.Adam(params, lr=0.001, ...,
weight_decay=0, ...)

optax.adamw(...) # Adam with weight decay regularization.
```

ℓ_1 regularization

- another form of regularization
- uses the sum of the absolute values of the parameters

$$R(\theta) = \sum_{j=1}^{m} |\theta_j|$$

- encourages sparsity in parameters and is useful for feature selection
- not differentiable at zero, so special optimization techniques are needed
- best known example: LASSO (Least Absolute Shrinkage and Selection Operator)

Choose λ via cross-validation

- λ controls the tradeoff between the original loss and the regularization term
- split the training set into training and validation sets
- train the model with different values of λ on the training set
- evaluate the model on the validation set
- choose the λ that gives the best performance on the validation set
- retrain the model on the entire training set with the chosen λ
- evaluate the model on the test set
- the test set should not be used to select the λ

k-fold cross-validation

- the training set is split into k equal-sized folds
- each fold is used as the validation set once
- the model is trained on the remaining $k-1$ folds
- the average performance across all folds is used to select the λ
- more computationally expensive but gives a more reliable estimate of the performance
- $k = 5$ or $k = 10$ are common choices
- $k = n$ is called leave-one-out cross-validation