Overfitting: how to fool the linear regression

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?

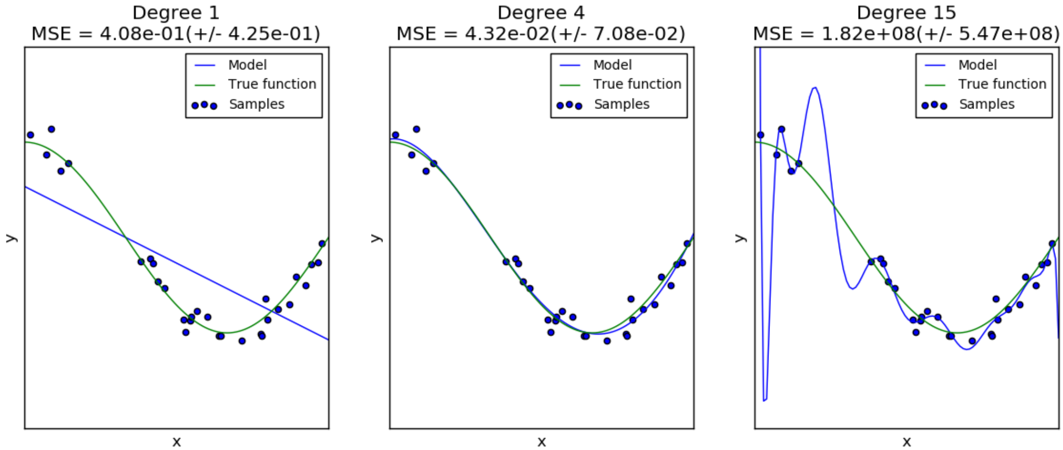
Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?
- Consider an example:
  - $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, $x \sim \mathrm{Uniform}[0, 1]$
  - Features: $\{x\}$, $\{x, x^2, x^3, x^4\}$, $\{x, \ldots, x^{15}\}$
  - The model is linear w. r. t. features: $f(\mathbf{x}) = \mathbf{w}^\intercal \phi(\mathbf{x})$

Generalization and overfitting

- Training set memorization: for seen $(\mathbf{x}, y) \in X^\ell$, $h(\mathbf{x}) = y$
- Generalization: equally good performance on both new and seen instances
- How to assess model's generalization ability?
- Consider an example:
  - $y = \cos(1.5\pi x) + \mathcal{N}(0, 0.01)$, $x \sim \text{Uniform}[0, 1]$
  - Features: $\{x\}$, $\{x, x^2, x^3, x^4\}$, $\{x, \dots, x^{15}\}$
  - The model is linear w. r. t. features: $f(\mathbf{x}) = \mathbf{w}^\intercal \phi(\mathbf{x})$
- How well do the regression models perform?

Polynomial fits of different degrees



Model validation and selection

- We have free parameters in models:

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?
- Yes, but we will likely get overly optimistic performance estimate

Model validation and selection

- We have free parameters in models:
  - polynomial degree d, subset of features in multivariate regression, kernel width in kernel density estimates, ...
- Model selection: how to select optimal hyperparameters for a given classification problem?
- Validation: how to estimate true model performance?
- Can we use entire dataset to fit the model?
- Yes, but we will likely get overly optimistic performance estimate
- The solution: rely on held-out data to assess model performance

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\text{VAL}})$

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

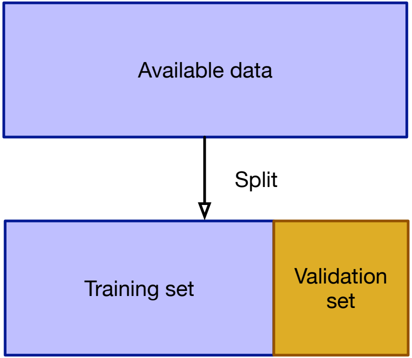
- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\text{VAL}})$
- Data-hungry: can we afford the ”luxury” of setting aside a portion of the data for testing?

Assessing generalization ability: train / validation

- Split training set into two subsets:

$$X^\ell = X^\ell_{\text{TRAIN}} \cup X^\ell_{\text{VAL}}$$

- Train a model h on X^ℓ_{TRAIN}
- Evaluate model h on X^ℓ_{VAL}
- Assess quality using $Q(h, X^\ell_{\text{VAL}})$
- **Data-hungry:** can we afford the "luxury" of setting aside a portion of the data for testing?
- **May be imprecise:** the holdout estimate of error rate will be misleading if we happen to get an "unfortunate" split



Assessing generalization ability: cross-validation

- Split training set into subsets of equal size
  $X^\ell = X^\ell_1 \cup \ldots \cup X^\ell_K$

Assessing generalization ability: cross-validation

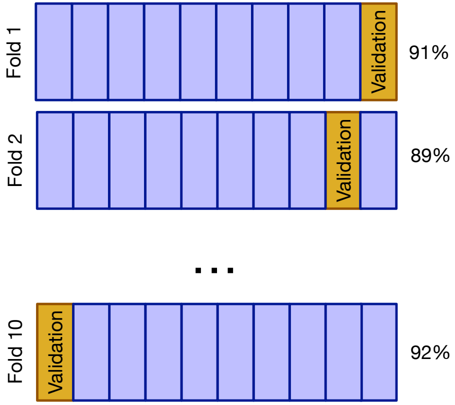
- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ

Assessing generalization ability: cross-validation

- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ
- Assess quality using $\mathrm{CV} = \frac{1}{K} \sum_{k=1}^{K} Q(h_k, X_k^\ell)$ (K-fold)

Assessing generalization ability: cross-validation

- Split training set into subsets of equal size $X^\ell = X_1^\ell \cup \ldots \cup X_K^\ell$
- Train K models $h_1, \ldots, h_K$ where each model h_k is trained on all subsets but X_k^ℓ
- Assess quality using $\mathrm{CV} = \frac{1}{K} \sum_{k=1}^{K} Q(h_k, X_k^\ell)$ (K-fold)
- Leave-one-out cross-validation: $X_k^\ell = \{(\mathbf{x}_k, y_k)\}$ (yes, train ℓ models!)



Cross-validation method: drawbacks

$$\mathrm{CV} = \frac{1}{K}\sum_{k=1}^{K} Q(h_k, X_k^\ell)$$

Many folds:

- Small bias: the estimator will be very accurate
- Large variance: due to small split sizes
- Costly: many experiments, large computational time

Cross-validation method: drawbacks

$$\mathrm{CV} = \frac{1}{K} \sum_{k=1}^{K} Q(h_k, X_k^{\ell})$$

Many folds:

- Small bias: the estimator will be very accurate
- Large variance: due to small split sizes
- Costly: many experiments, large computational time

Few folds:

- Cheap, computationally effective: few experiments
- Small variance: average over many samples
- Large bias: estimated error rate conservative or smaller than the true error rate

Regularization