universität wien

Introduction to Machine Learning

Model Validation and Selection

Nils M. Kriege

WS 2023

Data Mining and Machine Learning
Faculty of Computer Science
University of Vienna

Credit: Slides based on the IML Lectures by Sebastian Tschiatschek and Andreas Krause

Recap: Achieving generalization

- Fundamental assumption: Our data set is generated independently and identically distributed (iid) from some unknown distribution P:

$$(\mathbf{x}_i, y_i) \sim P(\mathbf{X}, Y)$$

- Our goal is to minimize the expected error (true risk) under P:

$$\begin{aligned} R(\mathbf{w}) &= \int P(\mathbf{x}, y)(y - \mathbf{w}^T\mathbf{x})^2 d\mathbf{x}d\mathbf{y} \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y}}[(y - \mathbf{w}^T\mathbf{x})^2] \end{aligned}$$

Recap: Evaluating predictive performance

- Training error (empirical risk) systematically underestimates true risk:

$$\mathbb{E}_{\mathcal{D}}[\hat{R}_{\mathcal{D}}(\hat{\mathbf{w}}_{\mathcal{D}})] \leq \mathbb{E}_{\mathcal{D}}[R(\hat{\mathbf{w}}_{\mathcal{D}})]$$

More realistic evaluation?

- Want to avoid underestimating the prediction error
- Idea: Use separate test set from the same distribution P
- Obtain training and test data $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$
- Optimize $\mathbf{w}$ on training set:

$$\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}} = \arg\min_{\mathbf{w}} \hat{R}_{\mathcal{D}_{\text{train}}}(\mathbf{w})$$

- Evaluate on test set:

$$\hat{R}_{\mathcal{D}_{\text{test}}}(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}}) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{test}}} (y - \hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}}^T \mathbf{x})^2$$

- Then:

$$\mathbb{E}_{\mathcal{D}_{\text{train}},\mathcal{D}_{\text{test}}}[\hat{R}_{\mathcal{D}_{\text{test}}}(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}})] = \mathbb{E}_{\mathcal{D}_{\text{train}}}[R(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}})]$$

Why?

Recap: Evaluating predictive performance

- Training error (empirical risk) systematically underestimates true risk:

$$\mathbb{E}_{\mathcal{D}}[\hat{R}_{\mathcal{D}}(\hat{\mathbf{w}}_{\mathcal{D}})] \leq \mathbb{E}_{\mathcal{D}}[R(\hat{\mathbf{w}}_{\mathcal{D}})]$$

- Using an independent test set avoids this bias:

$$\mathbb{E}_{\mathcal{D}_{\text{train}},\mathcal{D}_{\text{test}}}[\hat{R}_{\mathcal{D}_{\text{test}}}(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}})] = \mathbb{E}_{\mathcal{D}_{\text{train}}}[R(\hat{\mathbf{w}}_{\mathcal{D}_{\text{train}}})]$$

First Attempt: Evaluation for Model Selection

- Obtain training and test data $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$
- Fit each candidate model (e.g., degree m of polynomial):

$$\hat{\mathbf{w}}_m = \operatorname*{argmin}_{\mathbf{w}:\ \text{degree}(\mathbf{w}) \leq m} \hat{R}_{\text{train}}(\mathbf{w})$$

- Pick one which does best on test set:

$$\hat{m} = \operatorname*{argmin}_{m} \hat{R}_{\text{test}}(\hat{\mathbf{w}}_m)$$

First Attempt: Evaluation for Model Selection

- Obtain training and test data $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$
- Fit each candidate model (e.g., degree m of polynomial):

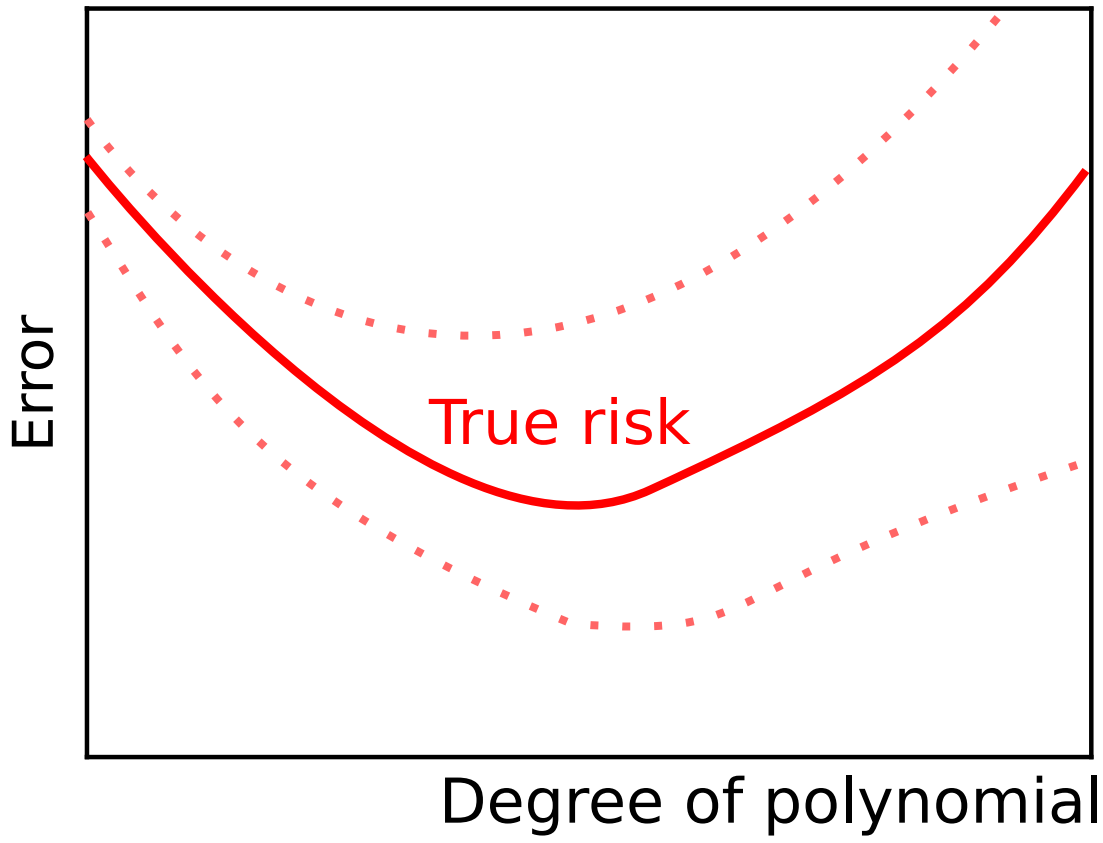
$$\hat{\mathbf{w}}_m = \operatorname*{argmin}_{\mathbf{w}:\ \text{degree}(\mathbf{w}) \leq m} \hat{R}_{\text{train}}(\mathbf{w})$$

- Pick one which does best on test set:

$$\hat{m} = \operatorname*{argmin}_{m} \hat{R}_{\text{test}}(\hat{\mathbf{w}}_m)$$

- Do you see a problem?

Overfitting to test set



- Test error is itself random! Variance usually increases for more complex models
- Optimizing for single test set creates bias

Solution: Pick Multiple Test Sets!

- Key idea: Instead of using a single test set, use multiple test sets and average to decrease variance
- Dilemma: Any data I use for testing I can't use for training

Solution: Pick Multiple Test Sets!

- Key idea: Instead of using a single test set, use multiple test sets and average to decrease variance
- Dilemma: Any data I use for testing I can't use for training
- $\Rightarrow$ Using multiple independent test sets is expensive and wasteful

Evaluation for Model Selection

- For each candidate model m (e.g., polynomial degree) repeat the following procedure for $i = 1, \ldots, k$:
  1. Split the same data into training and validation set:

$$\mathcal{D} = \mathcal{D}_{\text{train}}^{(i)} \uplus \mathcal{D}_{\text{val}}^{(i)}$$

  2. Train the model:

$$\hat{\mathbf{w}}_i = \operatorname*{argmin}_{\mathbf{w}} \hat{R}_{\text{train}}(\mathbf{w})$$

  3. Estimate error:

$$\hat{R}_m^{(i)} = \hat{R}_{\text{val}}^{(i)}(\hat{\mathbf{w}}_i)$$

- Select model:

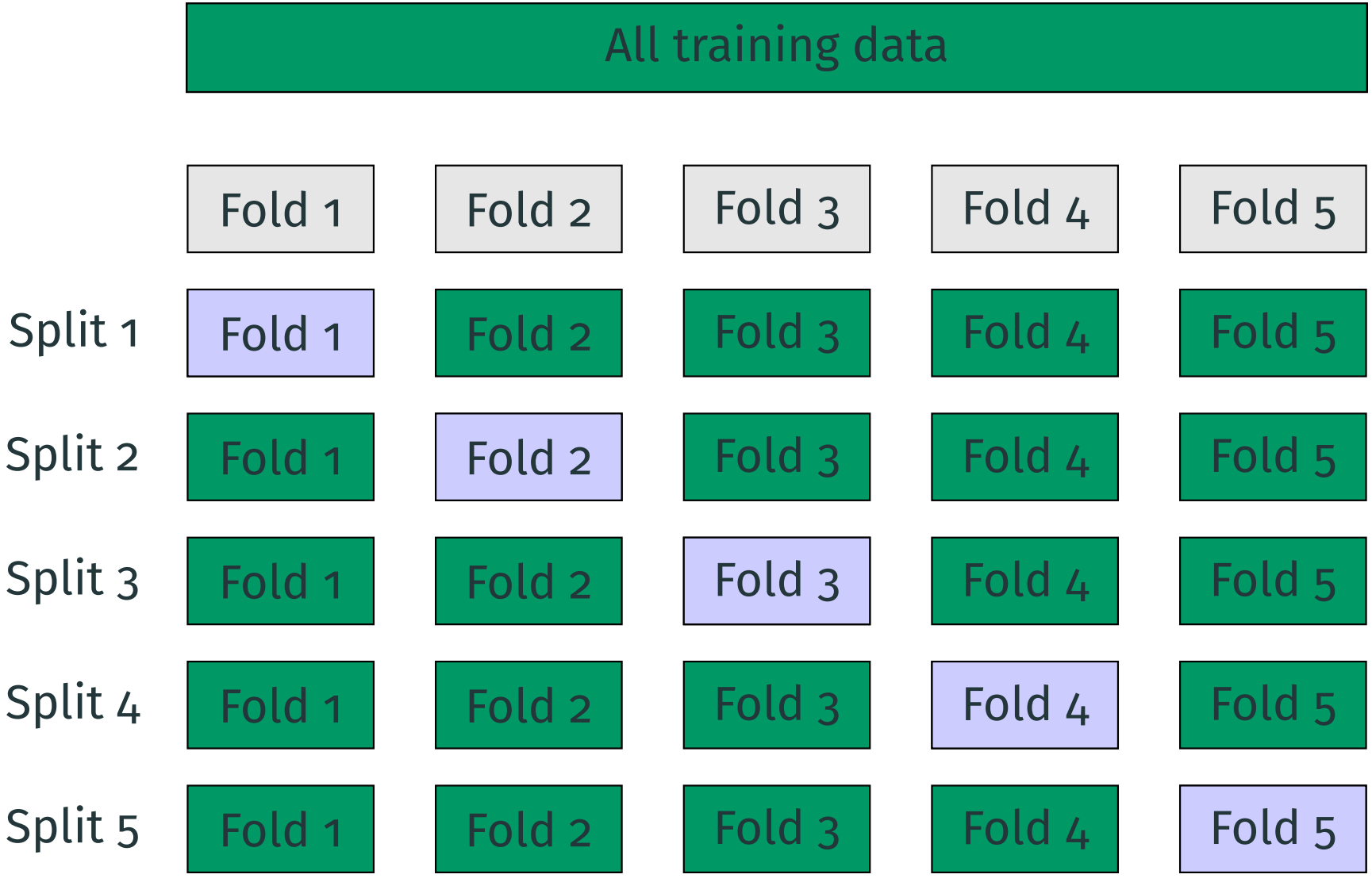
$$\hat{m} = \operatorname*{argmin}_m \frac{1}{k} \sum_{i=1}^{k} \hat{R}_m^{(i)}$$

How Should we do the Splitting?

- Randomly (Monte Carlo cross-validation
  - Pick training set of given size uniformly at random
  - Validate on remaining points
  - Estimate prediction error by averaging the validation error over multiple random trials
- k-fold cross-validation (default)
  - Partition the data into k “folds”
  - Train on $(k-1)$ folds, evaluating on the remaining fold
  - Estimate prediction error by averaging the validation error obtained while varying the validation fold

k-fold Cross-validation

Example for $k = 5$:



Accuracy of Cross-validation

- Cross-validation error estimate is very nearly unbiased for large enough k
- Demo

[model-selection/cross-validation-bias.ipynb]

Cross-validation

- How large should we pick k?
- Too small:
  - Risk of overfitting to the test set
  - Using too little data for training
  - Risk of underfitting on the training set
- Too large:
  - In general better performance! $k = n$ is perfectly fine (called leave-one-out cross-validation)
  - Higher computational costs
- In practice, $k = 5$ or $k = 10$ is often used and works well

Best Practice for Evaluating Supervised Learning

- Split data set into training and test set
- Never look at the test set when fitting the model. For example, use *k*-fold cross-validation on training set
- Report final accuracy on test set
  (but never optimize on test set!)

Best Practice for Evaluating Supervised Learning

- Split data set into training and test set
- Never look at the test set when fitting the model. For example, use k-fold cross-validation on training set
- Report final accuracy on test set
  (but never optimize on test set!)

- Caveat: this only works if the data is i.i.d.
- Be careful, for example, if there are temporal trends or other dependencies

Supervised learning summary so far

Representation/ features	Linear hypotheses, nonlinear hypotheses through feature transformations
Model/ objective	Loss-function (squared loss, ℓ_p loss)
Method	Exact solution, Gradient Descent
Evaluation metric	Empirical risk = (mean) squared error
Model selection	k-fold cross-validation, Monte Carlo cross-validation

Model Selection More Generally

- For polynomial regression, model complexity is naturally controlled by the degree
- In general, there may not be an ordering of the features that aligns with the complexity:
  - For example, how should we order words in the bag-of-words model?
  - Collection of nonlinear feature transformations:

$$x \mapsto \log(x + c)$$

$$x \mapsto x^{\alpha}$$

$$x \mapsto \sin(ax + b)$$

- Now model complexity is no longer naturally "ordered"

Demo: Overfitting $\rightarrow$ Large weights

[model-selection/overfitting-leads-to-large-weights.ipynb]

Regularization

- If we only seek to minimize our loss (optimize data fit), we can get very complex models (large weights)
- Solution?

Regularization

- If we only seek to minimize our loss (optimize data fit), we can get very complex models (large weights)
- Solution?
- Regularization!
  Encourage small weights via penalty functions (regularizers)

Ridge regression

- Regularized optimization problem:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2 \quad \lambda \geq 0$$

- Can optimize using gradient descent, or still find analytical solution:

$$\hat{w} = (\mathbf{X}^T \mathbf{X} + \lambda n \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

- Note that now the scale of **x** matters

Renormalizing Data: Standardization

- Ensure that each feature has zero mean and unit variance:

$$\tilde{x}_{i,j} = (x_{i,j} - \hat{\mu}_j)/\hat{\sigma}_j$$

- Hereby $x_{i,j}$ is the value of the j-th feature of the i-th data point:

$$\hat{\mu}_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j} \qquad \hat{\sigma}_j^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{i,j} - \hat{\mu}_j)^2$$

Gradient Descent for Ridge Regression

Demo: Regularization

[model-selection/regularization.ipynb]

How to choose the regularization parameter?

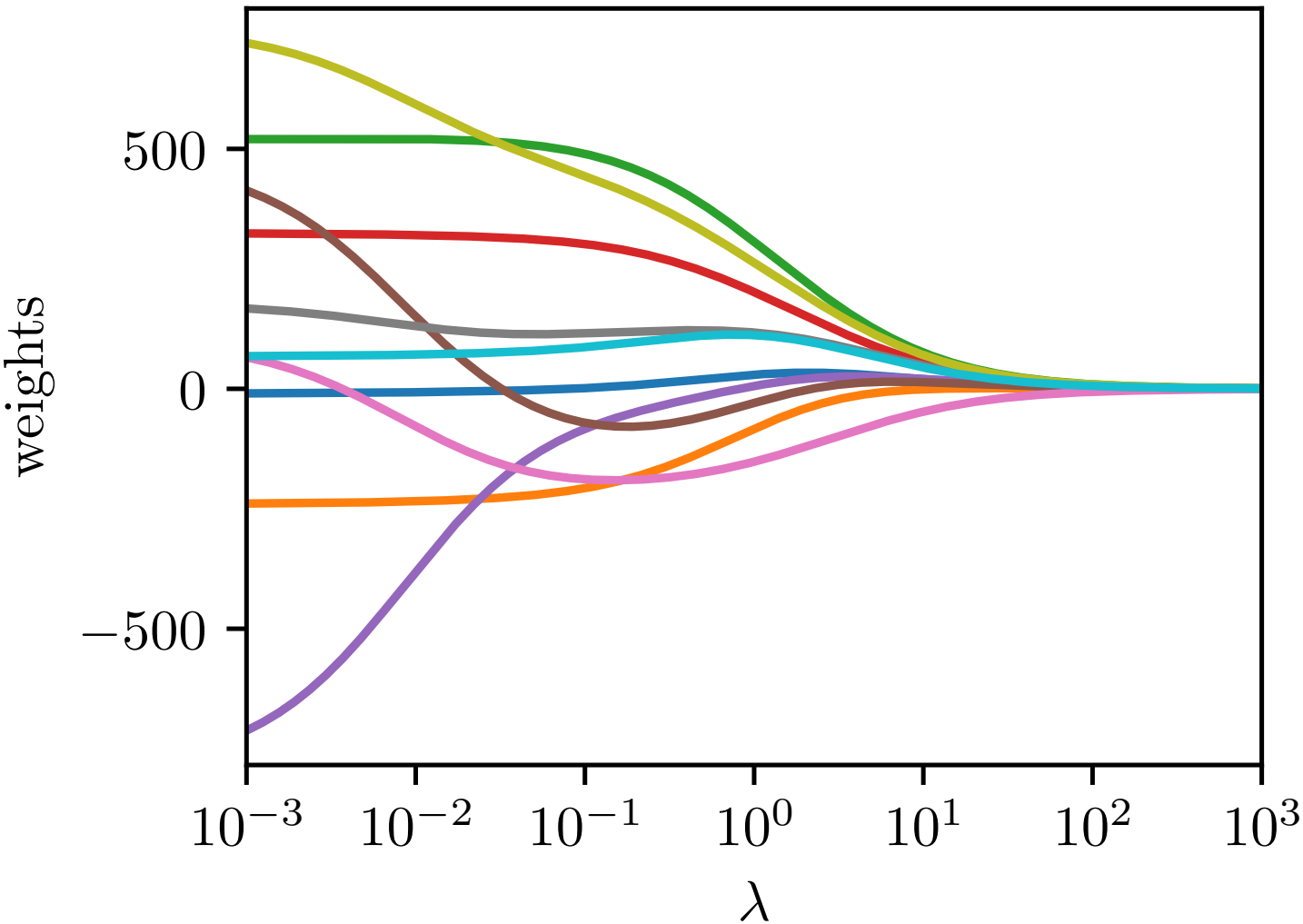
$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2 \quad \lambda \geq 0$$

How to choose the regularization parameter?

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2 \quad \lambda \geq 0$$

- Cross-validation
- Typically pick λ logarithmically spaced

Regularization Path



[model-selection/regression_path.py]

Outlook: Fundamental Tradeoff in ML

- Need to trade loss (goodness of fit) and simplicity
- A lot of supervised learning problems can be written in this way:

$$\min_{\mathbf{w}} \hat{R}(\mathbf{w}) + \lambda C(\mathbf{w})$$

- Can control complexity by varying regularization parameter λ
- Many other types of regularizers exist and are very useful

Supervised learning summary so far

Representation/ features	Linear hypotheses, non-linear hypotheses through feature transformations
Model/ objective	Loss-function (squared loss, ℓ_p loss) + Regularization (ℓ_2 norm)
Method	Exact solution, Gradient Descent
Evaluation metric	Empirical risk = (mean) squared error
Model selection	k-fold cross-validation, Monte Carlo cross-validation

What you need to know

- Linear regression as model and optimization problem:
  - How do you solve it?
  - Closed form vs gradient descent
  - Can represent non-linear functions using basis functions
- Model validation:
  - Resampling; cross-validation
- Model selection for regression:
  - Comparing different models via cross-validation
- Regularization:
  - Adding penalty function to control magnitude of weights
  - Choose regularization parameter via cross-validation