Mini-course Machine Learning in Empirical Economic Research

Lecture 2: Introduction to supervised learning

Andreas Dzemski[1]

[1]University of Gothenburg

June 5, 2019

Supervised learning

- outcome y, regressors/features x

Supervised learning

- outcome y, regressors/features x
- $f(x)$ prediction of y using prediction rule f

Supervised learning

- outcome y, regressors/features x
- $f(x)$ prediction of y using prediction rule f
- loss function $L(y, f(x)))$
- quadratic loss

$$L(y, f(x)) = (y - f(x))^2$$

Supervised learning

- outcome y, regressors/features x
- $f(x)$ prediction of y using prediction rule f
- loss function $L(y, f(x)))$
- quadratic loss

$$L(y, f(x)) = (y - f(x))^2$$

- function class $\mathcal{F}$ of possible f

Supervised learning

- outcome y, regressors/features x
- $f(x)$ prediction of y using prediction rule f
- loss function $L(y, f(x)))$
- quadratic loss

$$L(y, f(x)) = (y - f(x))^2$$

- function class $\mathcal{F}$ of possible f
- subclasses $\mathcal{F}_\lambda \subset \mathcal{F}$
- $\lambda =$ tuning parameter

Machine learning as data-driven model selection

"Traditional" econometrics

- specify $\mathcal{F}$

Machine Learning

- specify $\mathcal{F}$ and $\lambda \mapsto \mathcal{F}_\lambda$

Machine learning as data-driven model selection

"Traditional" econometrics

- specify $\mathcal{F}$
- $\mathcal{F}$ is typically very "small"

Machine Learning

- specify $\mathcal{F}$ and $\lambda \mapsto \mathcal{F}_\lambda$
- $\mathcal{F}$ is typically "large"

Machine learning as data-driven model selection

"Traditional" econometrics

- specify $\mathcal{F}$
- $\mathcal{F}$ is typically very "small"
- fit "best" $f \in \mathcal{F}$

Machine Learning

- specify $\mathcal{F}$ and $\lambda \mapsto \mathcal{F}_\lambda$
- $\mathcal{F}$ is typically "large"

Machine learning as data-driven model selection

"Traditional" econometrics

- specify $\mathcal{F}$
- $\mathcal{F}$ is typically very "small"
- fit "best" $f \in \mathcal{F}$

Machine Learning

- specify $\mathcal{F}$ and $\lambda \mapsto \mathcal{F}_\lambda$
- $\mathcal{F}$ is typically "large"
- use data to determine a "good" $\hat{\lambda}$
- typically $\mathcal{F}_{\hat{\lambda}}$ is "small" or at least well-behaved
- fit "best" $f \in \mathcal{F}_{\hat{\lambda}}$

Fitting $f \in \mathcal{F}_\lambda$

- training data $\mathcal{T} = \{(y_i, x_i')\}_{i=1}^n$
- *training error* = average loss on $\mathcal{T}$

$$\overline{\text{err}}(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f(x_i)\right)$$

Fitting $f \in \mathcal{F}_\lambda$

- training data $\mathcal{T} = \{(y_i, x_i')\}_{i=1}^n$
- *training error* = average loss on $\mathcal{T}$

$$\overline{\text{err}}(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f(x_i)\right)$$

- fit f with minimal training error

$$\hat{f} = \hat{f}^\lambda = \arg\min_{f \in \mathcal{F}^\lambda} \overline{\text{err}}(f)$$

Fitting $f \in \mathcal{F}_\lambda$

- training data $\mathcal{T} = \{(y_i, x_i')\}_{i=1}^n$
- *training error* = average loss on $\mathcal{T}$

$$\overline{\text{err}}(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f(x_i)\right)$$

- fit f with minimal training error

$$\hat{f} = \hat{f}^\lambda = \arg\min_{f \in \mathcal{F}^\lambda} \overline{\text{err}}(f)$$

- conceptually this is the same as what we do in econometrics!

Fitting $f \in \mathcal{F}_\lambda$

- training data $\mathcal{T} = \{(y_i, x_i')\}_{i=1}^n$
- *training error* = average loss on $\mathcal{T}$

$$\overline{\text{err}}(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f(x_i)\right)$$

- fit f with minimal training error

$$\hat{f} = \hat{f}^\lambda = \arg\min_{f \in \mathcal{F}^\lambda} \overline{\text{err}}(f)$$

- conceptually this is the same as what we do in econometrics!
  - what loss functions do we use in econometrics?

Fitting $f \in \mathcal{F}_\lambda$

- training data $\mathcal{T} = \{(y_i, x_i')\}_{i=1}^n$
- *training error* = average loss on $\mathcal{T}$

$$\overline{\text{err}}(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f(x_i)\right)$$

- fit f with minimal training error

$$\hat{f} = \hat{f}^\lambda = \arg\min_{f \in \mathcal{F}^\lambda} \overline{\text{err}}(f)$$

- conceptually this is the same as what we do in econometrics!
  - what loss functions do we use in econometrics?
  - how do we call the fitted training error?

Fitting $f \in \mathcal{F}_\lambda$

- training data $\mathcal{T} = \{(y_i, x_i')\}_{i=1}^n$
- *training error* = average loss on $\mathcal{T}$

$$\overline{\text{err}}(f) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, f(x_i)\right)$$

- fit f with minimal training error

$$\hat{f} = \hat{f}^\lambda = \arg\min_{f \in \mathcal{F}^\lambda} \overline{\text{err}}(f)$$

- conceptually this is the same as what we do in econometrics!
  - what loss functions do we use in econometrics?
  - how do we call the fitted training error?
  - estimation as constrained optimization?

Intuition for λ

- assume $\lambda \in \mathbb{R}$
- often λ performs *regularization*

Intuition for λ

- assume $\lambda \in \mathbb{R}$
- often λ performs *regularization*
- let $\mathcal{C}(f)$ measure "complexity" of f

$$\mathcal{F}_\lambda(f) = \{f \in \mathcal{F} : \mathcal{C}(f) \leq \lambda\}$$

Intuition for λ

- assume $\lambda \in \mathbb{R}$
- often λ performs *regularization*
- let $\mathcal{C}(f)$ measure "complexity" of f

$$\mathcal{F}_\lambda(f) = \{f \in \mathcal{F} : \mathcal{C}(f) \leq \lambda\}$$

- example: fitting a smooth curve over an interval

$$\mathcal{C}(f) = \int_{x \in [a,b]} |f''(x)|^2 \, dx$$

Intuition for λ

- assume $\lambda \in \mathbb{R}$
- often λ performs *regularization*
- let $\mathcal{C}(f)$ measure "complexity" of f

$$\mathcal{F}_\lambda(f) = \{f \in \mathcal{F} : \mathcal{C}(f) \leq \lambda\}$$

- example: fitting a smooth curve over an interval

$$\mathcal{C}(f) = \int_{x \in [a,b]} |f''(x)|^2 \, dx$$

- implies nesting of classes: for $\lambda_1 \leq \lambda_2 \leq \cdots$

$$\mathcal{F}_{\lambda_1} \subset \mathcal{F}_{\lambda_2} \subset \cdots$$

Choice of λ = trade-off

small λ

- small $\mathcal{F}_\lambda$

large λ

- large $\mathcal{F}_\lambda$

Choice of λ = trade-off

small λ

- small $\mathcal{F}_\lambda$
- consisting of "simple" models

large λ

- large $\mathcal{F}_\lambda$
- consisting of "complex" models

Choice of λ = trade-off

small λ

- small $\mathcal{F}_\lambda$
- consisting of "simple" models
- potentially poor approximation (high bias)

large λ

- large $\mathcal{F}_\lambda$
- consisting of "complex" models
- potentially better approximation (low bias)

Choice of λ = trade-off

small λ

- small $\mathcal{F}_\lambda$
- consisting of "simple" models
- potentially poor approximation (high bias)
- easy to estimate (low variance)

large λ

- large $\mathcal{F}_\lambda$
- consisting of "complex" models
- potentially better approximation (low bias)
- hard to estimate (high variance)

Choice of λ = trade-off

small λ

- small $\mathcal{F}_\lambda$
- consisting of "simple" models
- potentially poor approximation (high bias)
- easy to estimate (low variance)

large λ

- large $\mathcal{F}_\lambda$
- consisting of "complex" models
- potentially better approximation (low bias)
- hard to estimate (high variance)

- we order models according to the "Occam's razor" principle

Determining $\hat{\lambda}$

- our objective = prediction
- good λ leads to estimator with good predictive properties

Determining $\hat{\lambda}$

- our objective = prediction
- good λ leads to estimator with good predictive properties
- $\mathbb{E}_{\mathcal{T}}$ = expectation operator wrt training sample
- $E_{y,x}$ = integral wrt probability measure of a new (y, x') observation

Determining $\hat{\lambda}$

- our objective = prediction
- good λ leads to estimator with good predictive properties
- $\mathbb{E}_{\mathcal{T}}$ = expectation operator wrt training sample
- $E_{y,x}$ = integral wrt probability measure of a new (y, x') observation
- expected prediction error

$$EPE(\hat{f}^{\lambda}) = \mathbb{E}_{\mathcal{T}} E_{y,x} L\left(y, \hat{f}^{\lambda}(x)\right)$$

Training error does not estimate prediction error

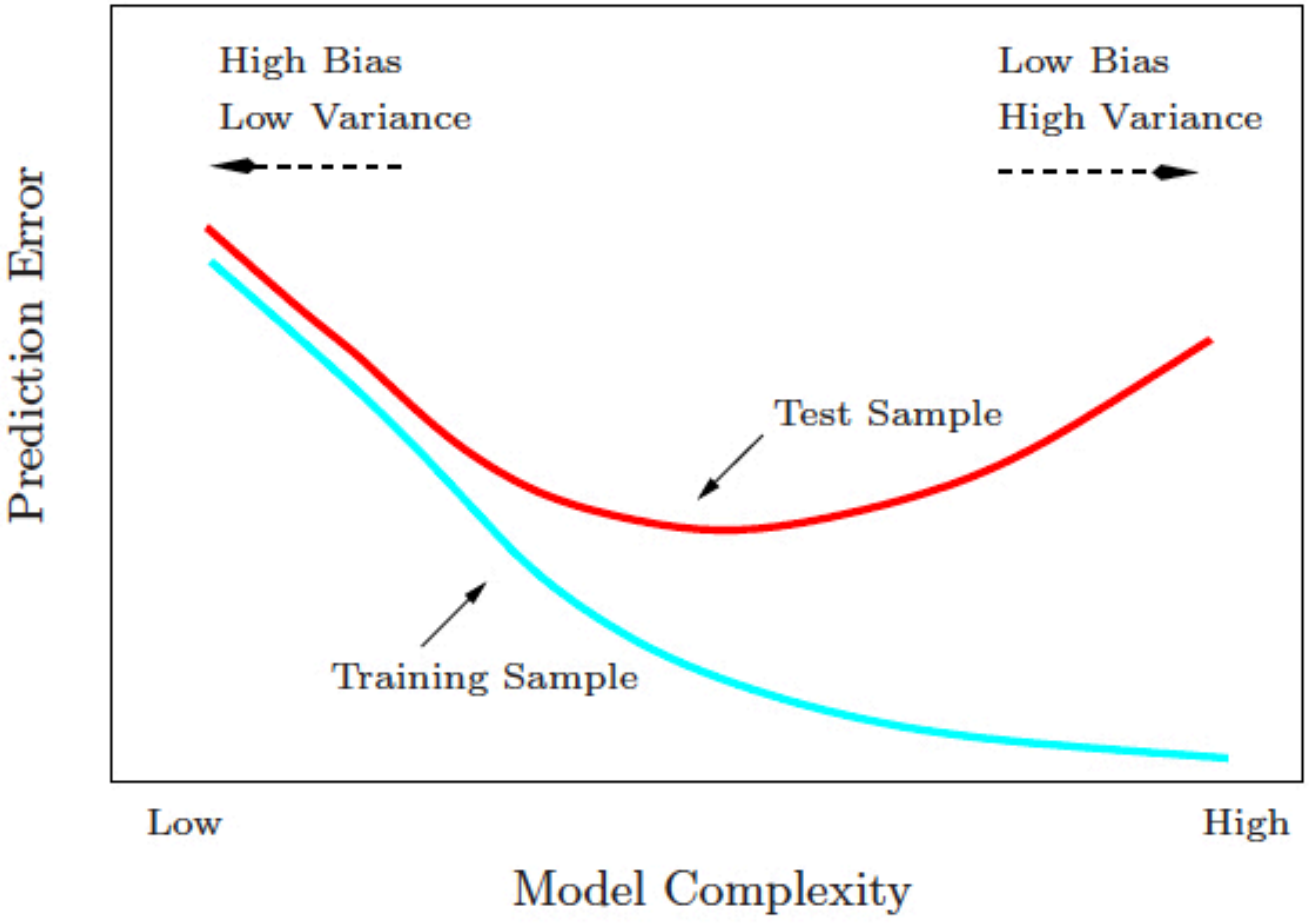

Figure: Figure 2.11 from Hastie, Tibshirani, and Friedman (2009)

Overfitting

- $\overline{\text{err}}(f)$ is an *optimistic* estimate of prediction error
  - the training error does not account for cost of uncertainty (variance)

Overfitting

- $\overline{\text{err}}(f)$ is an *optimistic* estimate of prediction error
  - the training error does not account for cost of uncertainty (variance)
- minimizing training error $\overline{\text{err}}(f)$ over all of $\mathcal{F}$ leads to *overfitting*

Overfitting

- $\overline{\text{err}}(f)$ is an *optimistic* estimate of prediction error
  - the training error does not account for cost of uncertainty (variance)
- minimizing training error $\overline{\text{err}}(f)$ over all of $\mathcal{F}$ leads to *overfitting*
- introductory econometrics courses often don't discuss overfitting, why?

Overfitting

- $\overline{\text{err}}(f)$ is an *optimistic* estimate of prediction error
  - the training error does not account for cost of uncertainty (variance)
- minimizing training error $\overline{\text{err}}(f)$ over all of $\mathcal{F}$ leads to *overfitting*
- introductory econometrics courses often don't discuss overfitting, why?
- Two lessons:

Overfitting

- $\overline{\text{err}}(f)$ is an *optimistic* estimate of prediction error
  - the training error does not account for cost of uncertainty (variance)
- minimizing training error $\overline{\text{err}}(f)$ over all of $\mathcal{F}$ leads to *overfitting*
- introductory econometrics courses often don't discuss overfitting, why?
- Two lessons:
  - estimation: fit $\overline{\text{err}}(f)$ keeping λ fixed

Overfitting

- $\overline{\text{err}}(f)$ is an *optimistic* estimate of prediction error
  - the training error does not account for cost of uncertainty (variance)
- minimizing training error $\overline{\text{err}}(f)$ over all of $\mathcal{F}$ leads to *overfitting*
- introductory econometrics courses often don't discuss overfitting, why?
- Two lessons:
  - estimation: fit $\overline{\text{err}}(f)$ keeping λ fixed
  - model testing: in-sample fit (aka $\overline{\text{err}}$ or R^2) is not a good measurement of fit
    - even if you are interested in predictive power don't look at R^2!

How do we validate fit in empirical economic research?

"For many years, economists have reported in-sample goodness-of-fit measures using the excuse that we had small datasets. But now that larger datasets have become available, there is no reason not to use separate training and testing sets." (Varian 2014)

Sample-splits in data-rich environments

Use independent samples for three distinct tasks:

Training fit $f \in \mathcal{F}_\lambda$

- this is how we fit $\hat{f}^\lambda$ for candidate λ
- minimize training error

Validation estimate EPE for given $\hat{f}^\lambda$

- this is how we evaluate suitability of λ candidates
- best candidate $= \hat{\lambda}$

Test estimate predictive power of final model $\hat{f}^{\hat{\lambda}}$

Estimating EPE from independent validation data set

Validation data set

$$\{(y_i^*, (x_i^*)')\}_{i=1}^{n^*}$$

Estimating EPE from independent validation data set

Validation data set

$$\{(y_i^*, (x_i^*)')\}_{i=1}^{n^*}$$

Estimated *EPE*

$$\widehat{EPE}(f) = \frac{1}{n^*} \sum_{i=1}^{n^*} L(y_i, f(x_i^*))$$

Estimating EPE from independent validation data set

Validation data set

$$\{(y_i^*, (x_i^*)')\}_{i=1}^{n^*}$$

Estimated *EPE*

$$\widehat{EPE}(f) = \frac{1}{n^*} \sum_{i=1}^{n^*} L(y_i, f(x_i^*))$$

- $\widehat{EPE}\left(\hat{f}^{\lambda}\right)$ estimates the *conditional* expected prediction error

$$EPE_{\mathcal{T}}\left(\hat{f}^{\lambda}\right) = E_{y,x} L\left(y, \hat{f}^{\lambda}(x)\right)$$

Estimating EPE using cross-validation

- validation step without need for validation data set

Estimating EPE using cross-validation

- validation step without need for validation data set
- k-fold cross-validation (CV) based on sample split *of the training sample*

Estimating EPE using cross-validation

- validation step without need for validation data set
- k-fold cross-validation (CV) based on sample split *of the training sample*
- example with $k = 5$ folds

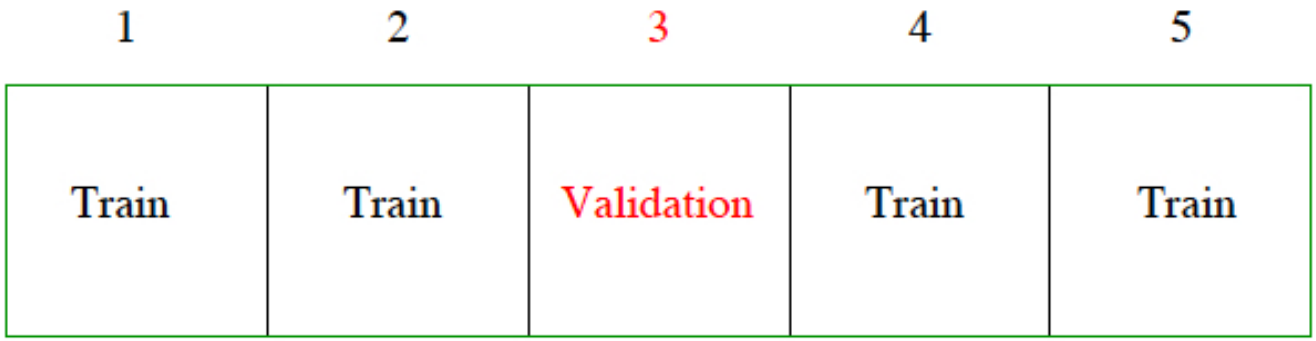

Figure: Figure from p242 of Hastie, Tibshirani, and Friedman (2009)

k-fold cross-validation

Random partition $\{\kappa_j\}_{j=1}^k$

$$\bigcup_{j=1}^{k} \kappa_j = \{1, \ldots, n\}$$

$\hat{f}^{\lambda}_{-\kappa_j}$ = fit $f \in \mathcal{F}_\lambda$ on subsample

$$\{1, \ldots, n\} \setminus \kappa_j$$

evaluate $\hat{f}_{-\kappa_j}$ on κ_j

$$CV_j(\lambda) = \frac{1}{|\kappa_j|} \sum_{i \in \kappa_j} L\big(\hat{f}^{\lambda}_{-\kappa_j}(x_i), y_i\big)$$

k-fold cross-validation

Average over partitions

$$CV(\lambda) = \frac{1}{k} \sum_{j=1}^{k} CV_j(\lambda)$$

k-fold cross-validation

Average over partitions

$$CV(\lambda) = \frac{1}{k} \sum_{j=1}^{k} CV_j(\lambda)$$

- $CV(\lambda)$ is an estimate of the *unconditional* expected prediction error

$$EPE\left(\hat{f}^{\lambda}\right) = \mathbb{E}_{\mathcal{T}} E_{y,x} L\left(\hat{f}^{\lambda}(x), y\right)$$

Standard error of cross-validation

$$SE_{CV}(\lambda) = \text{sample std of } \{CV_j(\lambda)\}_{j=1}^{k}/\sqrt{k}$$
$$= \sqrt{\frac{1}{k}\sum_{j=1}^{k}(CV_j(\lambda) - CV(\lambda))^2}/\sqrt{k}$$

- ad-hoc measure of the "sampling error" of cross-validation

Choice of number of folds

- typically values $k = 5, 10, n$
- $k = n$: leave-one-out cross-validation

few folds

- training step in CV uses *much less* observations than n

leave-one-out

- training step in CV almost n observations

Choice of number of folds

- typically values $k = 5, 10, n$
- $k = n$: leave-one-out cross-validation

few folds

- training step in CV uses *much less* observations than n
- biased

leave-one-out

- training step in CV almost n observations
- approximately unbiased

Choice of number of folds

- typically values $k = 5, 10, n$
- $k = n$: leave-one-out cross-validation

few folds

- training step in CV uses *much less* observations than n
- biased
- low variance (training sets not very similar)

leave-one-out

- training step in CV almost n observations
- approximately unbiased
- high variance (training sets very similar)

Example

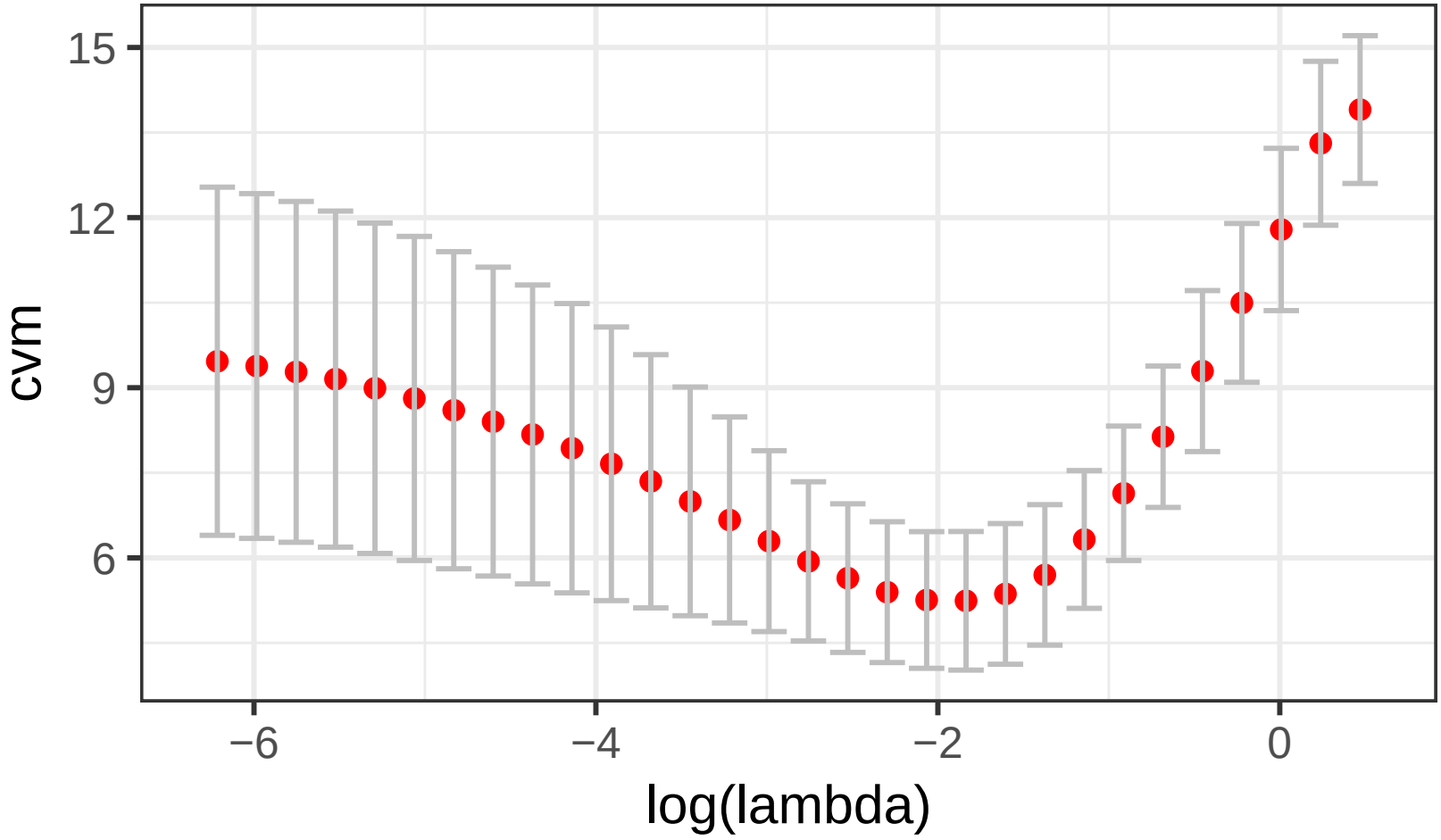

Figure: Cross-validation $CV(\lambda)$, $k = 10$ folds

Choice of λ based on CV

- optimize λ for prediction: minimial λ

$$\hat{\lambda} = \hat{\lambda}_{\text{min}} = \arg\min_{\lambda} CV(\lambda)$$

Choice of λ based on CV

- optimize λ for prediction: minimial λ

$$\hat{\lambda} = \hat{\lambda}_{\text{min}} = \arg\min_{\lambda} CV(\lambda)$$

- impose additional regularization: 1-σ rule

$$\begin{aligned}\hat{\lambda} =& \hat{\lambda}_{1\sigma} \\ =& \max\{\lambda \in \mathbb{R} : CV(\lambda) \leq CV(\hat{\lambda}_{\text{min}}) + SE_{CV}(\hat{\lambda}_{\text{min}})\}\end{aligned}$$

Example

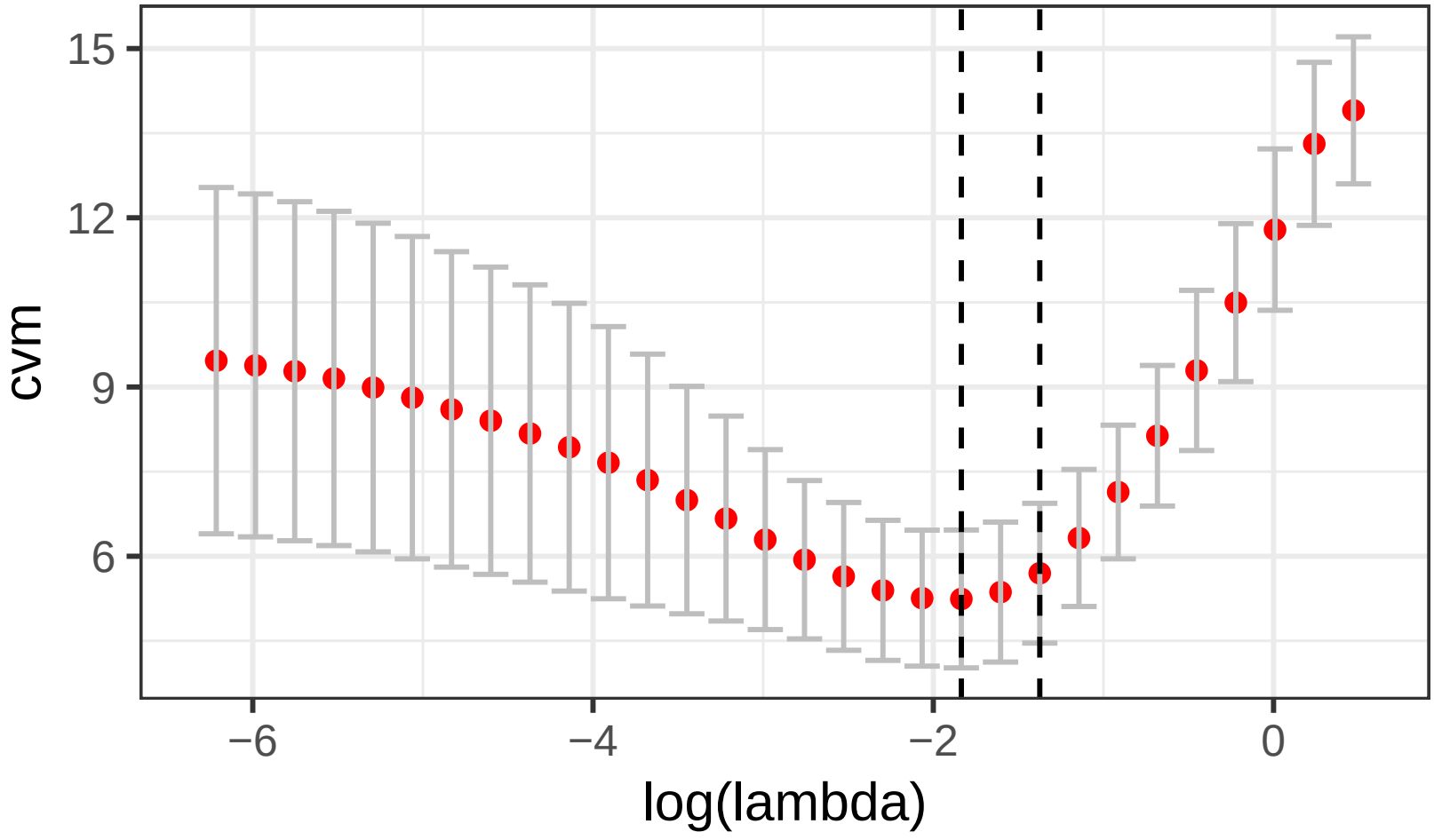

Figure: Optimal choice of λ, $k = 10$ folds

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.
- $\hat{f}^{\lambda}$ is *learned*

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.
- $\hat{f}^{\lambda}$ is *learned*

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.
- $\hat{f}^{\lambda}$ is *learned*

Hyperparameter optimization

- choose λ

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.
- $\hat{f}^{\lambda}$ is *learned*

Hyperparameter optimization

- choose λ
- find the λ that gives $\hat{f}^{\lambda}$ with the best generalization (= out-of-sample) performance

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.
- $\hat{f}^{\lambda}$ is *learned*

Hyperparameter optimization

- choose λ
- find the λ that gives $\hat{f}^{\lambda}$ with the best generalization (= out-of-sample) performance
  - independent validation sample

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.
- $\hat{f}^{\lambda}$ is *learned*

Hyperparameter optimization

- choose λ
- find the λ that gives $\hat{f}^{\lambda}$ with the best generalization (= out-of-sample) performance
  - independent validation sample
  - cross-validation

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.
- $\hat{f}^{\lambda}$ is *learned*

Hyperparameter optimization

- choose λ
- find the λ that gives $\hat{f}^{\lambda}$ with the best generalization (= out-of-sample) performance
  - independent validation sample
  - cross-validation
  - other

Applying a supervised learning method

- have to choose a *tuning parameter* λ
  - regularization parameter
  - hyperparameter
- can be a vector

Training the model

- For given tuning parameter λ, fit the best model $\hat{f}^{\lambda}$ using training data.
- $\hat{f}^{\lambda}$ is *learned*

Hyperparameter optimization

- choose λ
- find the λ that gives $\hat{f}^{\lambda}$ with the best generalization (= out-of-sample) performance
  - independent validation sample
  - cross-validation
  - other
- compute the training step many times

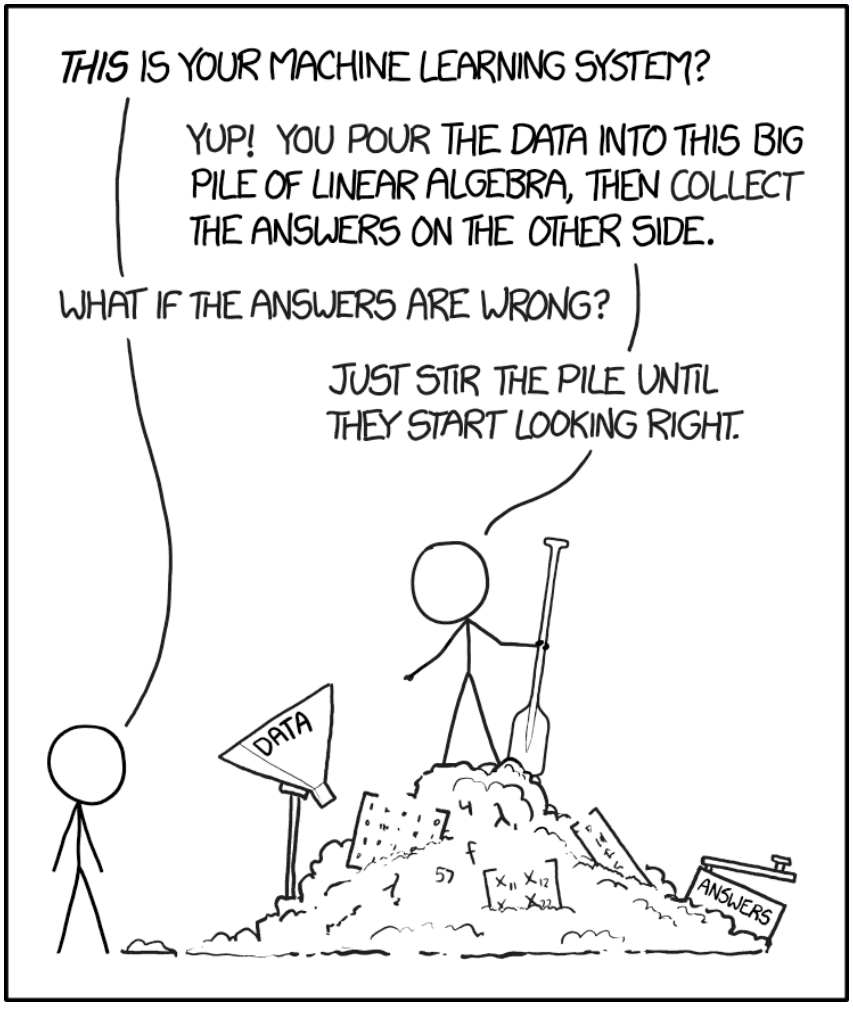

Figure: Source: https://xkcd.com/1838/