- How do we ensure that our model is not overly complex (i.e. overfit)?

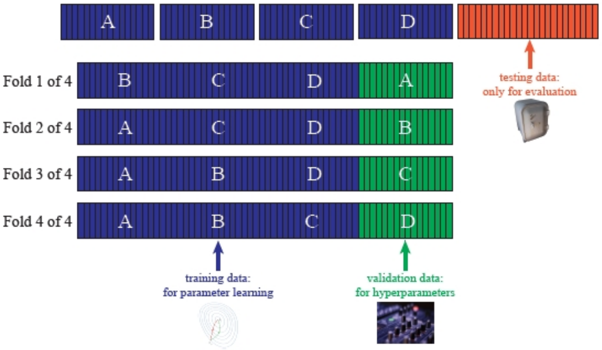
- How do we ensure that our model is not overly complex (i.e. overfit)?
- The answer is to penalize complexity

- How do we ensure that our model is not overly complex (i.e. overfit)?
- The answer is to penalize complexity
- Regularization is the way we penalize complexity

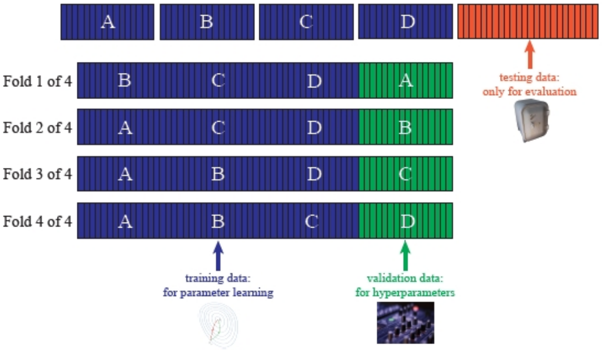
- How do we ensure that our model is not overly complex (i.e. overfit)?
- The answer is to penalize complexity
- Regularization is the way we penalize complexity
- Cross-validation is the way that we choose the optimal level of regularization

How cross-validation works (Adams 2018):

How cross-validation works (Adams 2018):

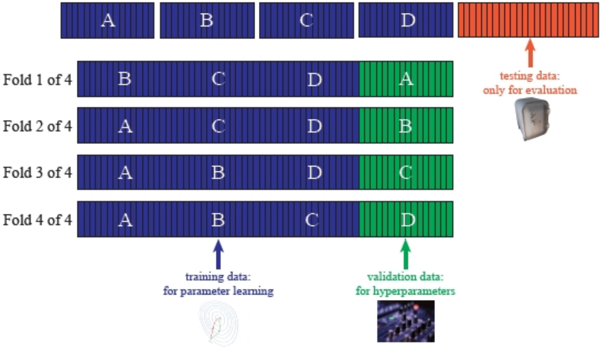


How cross-validation works (Adams 2018):



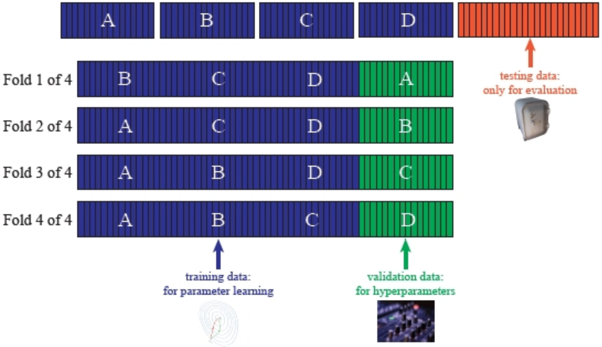
- Blue is the data that we use to estimate the model's parameters

How cross-validation works (Adams 2018):



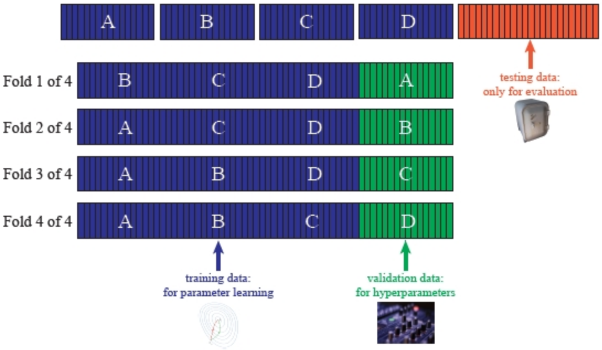
- Blue is the data that we use to estimate the model's parameters
- We randomly hold out K portions of this data one-at-a-time (Green)

How cross-validation works (Adams 2018):



- Blue is the data that we use to estimate the model's parameters
- We randomly hold out K portions of this data one-at-a-time (Green)
- We assess the performance of the model in the Green data

How cross-validation works (Adams 2018):



- Blue is the data that we use to estimate the model's parameters
- We randomly hold out K portions of this data one-at-a-time (Green)
- We assess the performance of the model in the Green data
- This tells us the optimal complexity ("hyperparameter" values)

Regularization is algorithm-specific:

Regularization is algorithm-specific:

- in tree models, complexity is the number of “leaves” on the tree

Regularization is algorithm-specific:

- in tree models, complexity is the number of “leaves” on the tree
- in linear models, complexity is the number of covariates

Regularization is algorithm-specific:

- in tree models, complexity is the number of “leaves” on the tree
- in linear models, complexity is the number of covariates
- in neural networks, complexity is the number/mapping of hidden layers

Regularization is algorithm-specific:

- in tree models, complexity is the number of “leaves” on the tree
- in linear models, complexity is the number of covariates
- in neural networks, complexity is the number/mapping of hidden layers
- in Bayesian approaches, priors act as regularization

Regularization is algorithm-specific:

- in tree models, complexity is the number of “leaves” on the tree
- in linear models, complexity is the number of covariates
- in neural networks, complexity is the number/mapping of hidden layers
- in Bayesian approaches, priors act as regularization

Whatever our algorithm, we can tune the complexity parameters using CV

There are three main types of regularization for linear-in-parameters models:

There are three main types of regularization for linear-in-parameters models:

1. L_0 regularization (Subset selection)

There are three main types of regularization for linear-in-parameters models:

1. L_0 regularization (Subset selection)

2. L_1 regularization (LASSO)

There are three main types of regularization for linear-in-parameters models:

1. L_0 regularization (Subset selection)
2. L_1 regularization (LASSO)
3. L_2 regularization (Ridge)

L_0 regularization:

L_0 regularization:

- Suppose you have L X's you may want to include in your model

L_0 regularization:

- Suppose you have L X's you may want to include in your model
- Subset selection automatically chooses which ones to include

L_0 regularization:

- Suppose you have L X's you may want to include in your model
- Subset selection automatically chooses which ones to include
- This is an automated version of what is traditionally done in econometrics

L_0 regularization:

- Suppose you have L X's you may want to include in your model
- Subset selection automatically chooses which ones to include
- This is an automated version of what is traditionally done in econometrics
- Can use Adjusted R^2 to penalize complexity (or AIC, BIC, or a penalized SSR)

L_0 regularization:

- Suppose you have L X's you may want to include in your model
- Subset selection automatically chooses which ones to include
- This is an automated version of what is traditionally done in econometrics
- Can use Adjusted R^2 to penalize complexity (or AIC, BIC, or a penalized SSR)
- Algorithm either starts from 0 X's and moves forward

L_0 regularization:

- Suppose you have L X's you may want to include in your model
- Subset selection automatically chooses which ones to include
- This is an automated version of what is traditionally done in econometrics
- Can use Adjusted R^2 to penalize complexity (or AIC, BIC, or a penalized SSR)
- Algorithm either starts from 0 X's and moves forward
- Or it starts from the full set of X's and works backward

L_0 regularization:

- Suppose you have L X's you may want to include in your model
- Subset selection automatically chooses which ones to include
- This is an automated version of what is traditionally done in econometrics
- Can use Adjusted R^2 to penalize complexity (or AIC, BIC, or a penalized SSR)
- Algorithm either starts from 0 X's and moves forward
- Or it starts from the full set of X's and works backward
- But this won't work if $L > N$! (i.e. there are more X's than observations)

Consider two different penalized versions of the OLS model:

$$\min_{\beta} \sum_{i} \left(y_i - x_i'\beta\right)^2 + \lambda \sum_{k} |\beta_k| \quad \text{(LASSO)}$$

Consider two different penalized versions of the OLS model:

$$\min_{\beta} \sum_{i} \left(y_i - x_i'\beta\right)^2 + \lambda \sum_{k} |\beta_k| \quad \text{(LASSO)}$$

- LASSO: Least Absolute Shrinkage and Selection Operator — sets some β's to be 0, others to be attenuated in magnitude

Consider two different penalized versions of the OLS model:

$$\min_{\beta} \sum_{i} \left(y_i - x_i'\beta\right)^2 + \lambda \sum_{k} |\beta_k| \quad \text{(LASSO)}$$
$$\min_{\beta} \sum_{i} \left(y_i - x_i'\beta\right)^2 + \lambda \sum_{k} \beta_k^2 \quad \text{(Ridge)}$$

- LASSO: Least Absolute Shrinkage and Selection Operator — sets some β's to be 0, others to be attenuated in magnitude
- Ridge: sets each β to be attenuated in magnitude

- We want to choose λ to optimize the bias-variance tradeoff

- We want to choose λ to optimize the bias-variance tradeoff
- We choose λ by k-fold Cross Validation

- We want to choose λ to optimize the bias-variance tradeoff
- We choose λ by k-fold Cross Validation
- We can also employ a weighted average of L_1 and L_2, known as elastic net

- We want to choose λ to optimize the bias-variance tradeoff

- We choose λ by k-fold Cross Validation

- We can also employ a weighted average of L_1 and L_2, known as elastic net

$$\min_{\beta} \sum_{i} \left(y_i - x_i'\beta\right)^2 + \lambda_1 \sum_{k} |\beta_k| + \lambda_2 \sum_{k} \beta_k^2$$

- We want to choose λ to optimize the bias-variance tradeoff

- We choose λ by k-fold Cross Validation

- We can also employ a weighted average of L_1 and L_2, known as elastic net

$$\min_{\beta} \sum_{i} \left(y_i - x_i' \beta\right)^2 + \lambda_1 \sum_{k} |\beta_k| + \lambda_2 \sum_{k} \beta_k^2$$

where we choose (λ_1, λ_2) by cross-validation

- We want to choose λ to optimize the bias-variance tradeoff

- We choose λ by k-fold Cross Validation

- We can also employ a weighted average of L_1 and L_2, known as elastic net

$$\min_{\beta} \sum_{i} \left(y_i - x_i'\beta\right)^2 + \lambda_1 \sum_{k} |\beta_k| + \lambda_2 \sum_{k} \beta_k^2$$

where we choose (λ_1, λ_2) by cross-validation

- We can apply L_1 and L_2 to anything linear-in-parameters (logit, neural net, ...)

- We want to choose λ to optimize the bias-variance tradeoff

- We choose λ by k-fold Cross Validation

- We can also employ a weighted average of L_1 and L_2, known as elastic net

$$\min_{\beta} \sum_{i} \left(y_i - x_i'\beta\right)^2 + \lambda_1 \sum_{k} |\beta_k| + \lambda_2 \sum_{k} \beta_k^2$$

where we choose (λ_1, λ_2) by cross-validation

- We can apply L_1 and L_2 to anything linear-in-parameters (logit, neural net, ...)

- L_1 and L_2 are excellent for high-dimensional problems ($L > N$)