Lecture 8: Algorithm Independent Principles - II

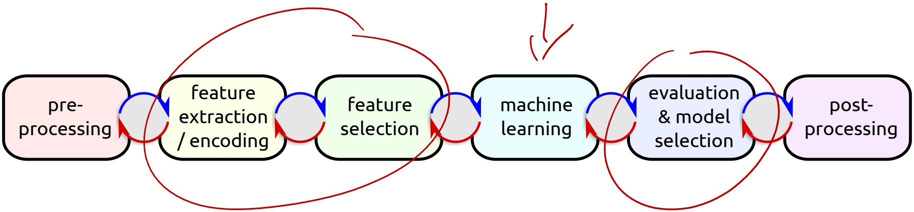
Validation, Regularization, General Issues, Model Selection

Machine Learning, Summer Term 2019

Michael Tangermann Frank Hutter Marius Lindauer

University of Freiburg

The Big Picture

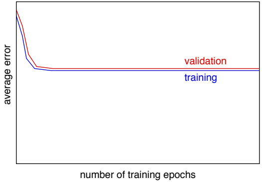


Lecture Overview

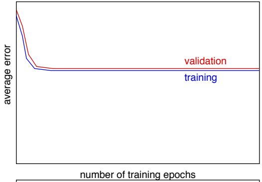
Lecture Overview

Recap: Overfitting / Underfitting

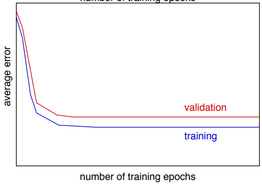
Example of underfitting:
validation and training error remain large



Example of underfitting:
validation and training error remain large

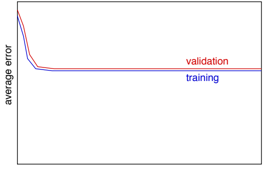


Example of successful learning:
validation error and training error monotonically decrease
$\rightsquigarrow$ good generalization

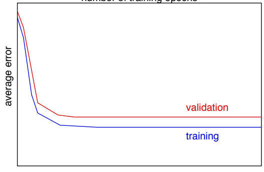


Recap: Overfitting / Underfitting

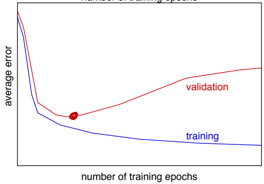
Example of underfitting:
validation and training error remain large



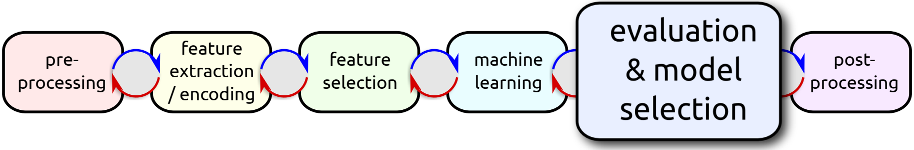
Example of successful learning:
validation error and training error monotonically decrease
$\rightsquigarrow$ good generalization



Example of overfitting:
validation error increases while training error decreases



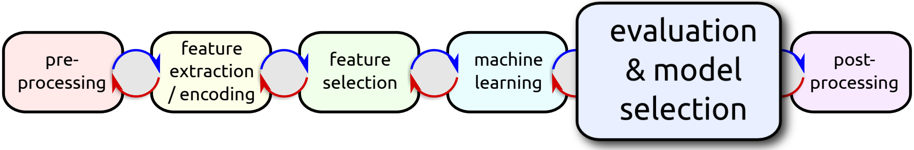
Model Selection 1/2



Evaluation:

- We want our models to generalize
  - I.e., perform well on previously unseen data points
- What does it mean to perform well?
  - See metrics covered later today
  - Speed at training time, speed at test time, memory, accuracy, . . .

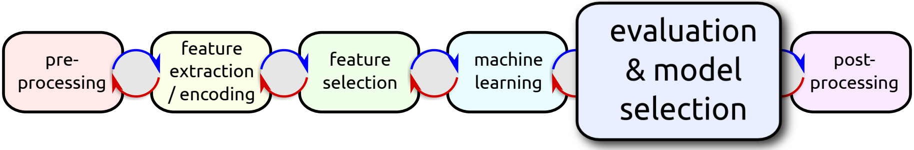
Model Selection 2/2



To obtain estimates of generalization performance using a fixed dataset

- Split dataset into training set and test set
- Lock away test set for final assessment

Model Selection 2/2

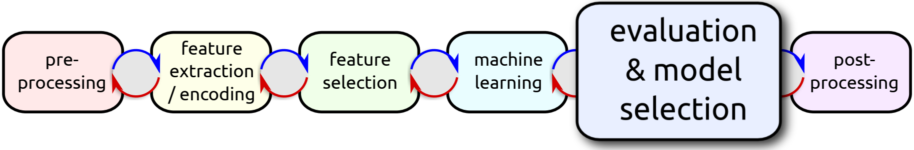


To obtain estimates of generalization performance using a fixed dataset

- Split dataset into training set and test set
- Lock away test set for final assessment

- You can do anything you want on the training set
- E.g., split it further into training and validation
  - Train different models on training set
  - Pick the one with best performance on validation set

Model Selection 2/2

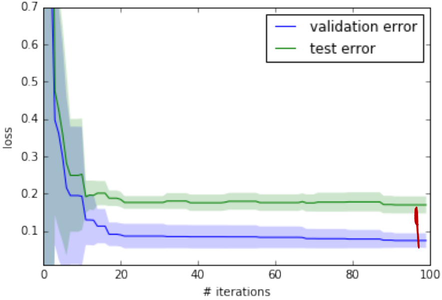


To obtain estimates of generalization performance using a fixed dataset

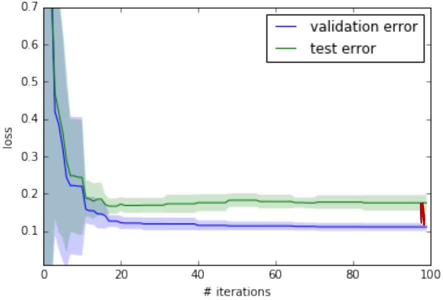
- Split dataset into training set and test set
- Lock away test set for final assessment

- You can do anything you want on the training set
- E.g., split it further into training and validation
  - Train different models on training set
  - Pick the one with best performance on validation set
- E.g., split it further into cross-validation (CV) folds and pick the model with best CV performance
  - CV performance is not an unbiased estimate of test performance
  - But better estimate than a single split into training/valid

Cross Validation Can Still Overfit 1/2



Single training-validation split (90% - 10%)



10-fold cross-validation

- Validation performance from single training-validation split is overconfident (overly optimistic)
- CV performance is still overconfident, but not as much
- In one of the next assignments, you will basically create these plots

Cross Validation Can Still Overfit 2/2

- To overfit least, when and how would you choose between different preprocessors (or feature selectors, data normalizers, etc) when you use k-fold cross-validation?
  - ★ Once: in the beginning, on all data
  - ★ Once: on all the training data
  - ★ k times: on the training split of each CV fold ✓

Cross Validation Can Still Overfit 2/2

- To overfit least, when and how would you choose between different preprocessors (or feature selectors, data normalizers, etc) when you use k-fold cross-validation?
  - ★ Once: in the beginning, on all data
  - ★ Once: on all the training data
  - ★ k times: on the training split of each CV fold

- Typical method trainModel
  - Normalizes data, drops unimportant features, etc
  - Then builds model on preprocessed data
  - Saves both these data transformations and the model

Cross Validation Can Still Overfit 2/2

- To overfit least, when and how would you choose between different preprocessors (or feature selectors, data normalizers, etc) when you use k-fold cross-validation?
  - ★ Once: in the beginning, on all data
  - ★ Once: on all the training data
  - ★ k times: on the training split of each CV fold

- Typical method trainModel
  - Normalizes data, drops unimportant features, etc
  - Then builds model on preprocessed data
  - Saves both these data transformations and the model

- Typical method applyModel
  - First apply the transformations, then the model
  - Routine used for both the validation set and the test set alike

Stratified Cross Validation

- E.g., only 10 positive data points, 90 negative ones
- How many positive data points would standard 10-fold cross-validation put into each fold?
  - ★ 1
  - ★ 10
  - ★ Between 0 and 10, depends on the random split into folds ✓

Stratified Cross Validation

- E.g., only 10 positive data points, 90 negative ones
- How many positive data points would standard 10-fold cross-validation put into each fold?
  - ★ 1
  - ★ 10
  - ★ Between 0 and 10, depends on the random split into folds

Stratified cross-validation would put exactly 1 positive and 9 negative data points into each fold

Lecture Overview

Regularization

Approaches to improve generalization

Which approaches do you know / can you think of? 

L2 reg.

early stopping

dropout

Regularization - Overview

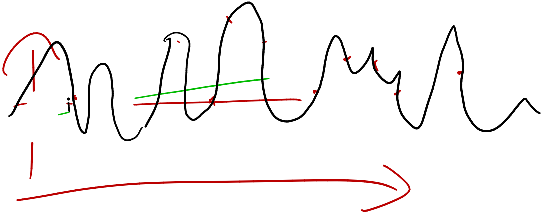
Approaches to improve generalization

Regularization - Overview

Approaches to improve generalization

Preference for small parameter values

- Early stopping
- Shrinkage methods



$$ax^n + bx^{n+1} + \dots + z$$

Regularization - Overview

Approaches to improve generalization

Preference for small parameter values

- Early stopping
- Shrinkage methods

Better task description

- More training data
- Filter training data
- Use more / less / other input features

Regularization - Overview

Approaches to improve generalization

Preference for small parameter values

- Early stopping
- Shrinkage methods

Better task description

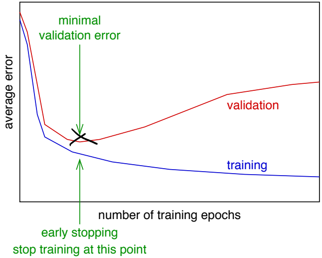
- More training data
- Filter training data
- Use more / less / other input features

Ensemble techniques

- Bagging

Regularization Techniques - Early Stopping

- Stop learning when error on validation set has reached its minimum
- Often, training is already stopped after a few epochs
  - typically combined with small initial weights
- Simple, popular heuristic
- Needs perpetual observation of the validation error



Regularization Techniques - Shrinkage Methods

- Extend the loss function with extra term (penalty) to control overfitting

$$L(\mathbf{w}) = L_D(\mathbf{w}) + \lambda L_W(\mathbf{w})$$

Regularization Techniques - Shrinkage Methods

- Extend the loss function with extra term (penalty) to control overfitting

$$L(\mathbf{w}) = L_D(\mathbf{w}) + \lambda L_W(\mathbf{w})$$

- Common example: sum-of-squares loss function with L2 regularization

$$L(\mathbf{w}) = \underbrace{\frac{1}{2}\sum_{n=1}^{N}\{y_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2}_{L_D(\mathbf{w})} + \boxed{\underbrace{\frac{\lambda}{2}\mathbf{w}^T\mathbf{w}}_{\lambda L_W(\mathbf{w})}}$$

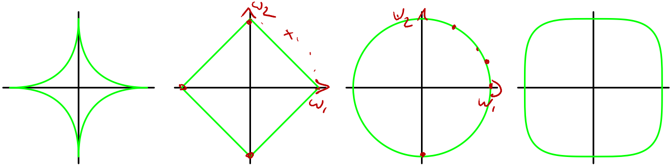
  - Allows for closed-form solution: $\mathbf{w} = (\lambda\mathbf{I} + \mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y}$. This is called ridge-regression or Tikhonov regularization in the literature.

Regularization Techniques - Shrinkage Methods

- More general form:

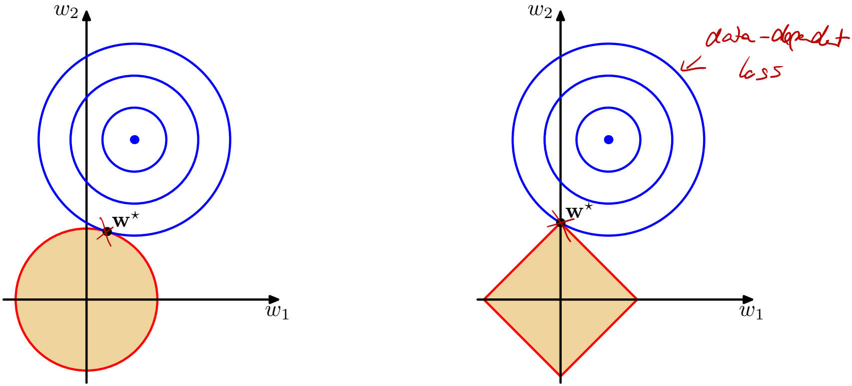
$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2 + \boxed{\frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q}$$

- Shrinkage: encourages weights to shrink towards zero
- Case $q = 2$: L2 regularizer as before
- Case $q = 1$: L1 regularizer as before (lasso in the literature) $\rightarrow$ sparse solutions



$q = 0.5$ $q = 1$ $q = 2$ $q = 4$

Regularization techniques - Shrinkage Methods



Regularization Techniques - Shrinkage Methods

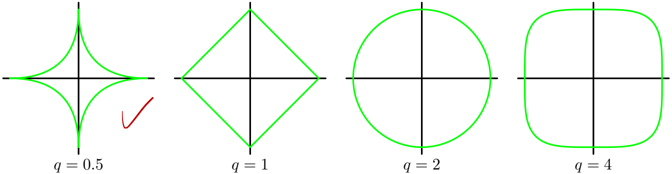
$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

- Which of these values of q encourages sparsity the most?
  - ★ q=0.5
  - ★ q=1
  - ★ q=2
  - ★ q=4

Regularization Techniques - Shrinkage Methods

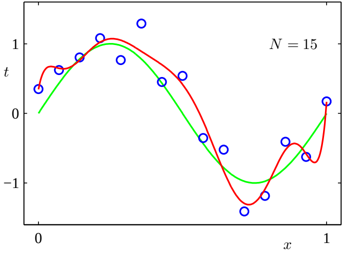
$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

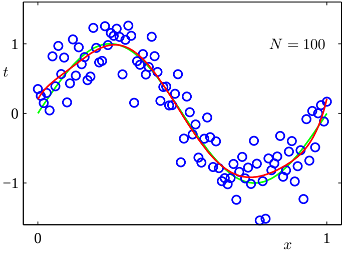
- Which of these values of q encourages sparsity the most?
  - ★ q=0.5 ✓
  - ★ q=1
  - ★ q=2
  - ★ q=4



Regularization Techniques - More Data

- Data analysts' fundamental slogan: *there's no data like more data!*
- Try to get more data; if not possible directly, think about related sources of similar data
- Although trivial, one of the most important techniques to improve ML models





Regularization Techniques - Data Augmentation

- Data augmentation is a common strategy for creating additional training data
- Example: computer vision
  - Translation, Scaling, Reflection, Rotation, Stretching

Regularization Techniques - Data Augmentation

- Data augmentation is a common strategy for creating additional training data
- Example: computer vision
  - Translation, Scaling, Reflection, Rotation, Stretching

Regularization Techniques - Data Augmentation

- Data augmentation is a common strategy for creating additional training data
- Example: computer vision
  - Translation, Scaling, Reflection, Rotation, Stretching

Regularization Techniques - Data Augmentation

- Data augmentation is a common strategy for creating additional training data
- Example: computer vision
  - Translation, Scaling, Reflection, Rotation, Stretching

Regularization Techniques - Data Augmentation

- Data augmentation is a common strategy for creating additional training data
- Example: computer vision
  - Translation, Scaling, Reflection, Rotation, Stretching

Regularization Techniques - Data Augmentation

- Data augmentation is a common strategy for creating additional training data
- Example: computer vision
  - Translation, Scaling, Reflection, Rotation, Stretching

- Hypothesis learns representation invariant to the perturbations
  - Often yields very substantial improvements
  - However: 100x augmentation can slow down training 100x
  - Diminishing returns: rarely more than 100x data augmentation

Regularization Techniques - Data Augmentation

- Data augmentation is a common strategy for creating additional training data
- Example: computer vision
  - Translation, Scaling, Reflection, Rotation, Stretching

- Hypothesis learns representation invariant to the perturbations
  - Often yields very substantial improvements
  - However: 100x augmentation can slow down training 100x
  - Diminishing returns: rarely more than 100x data augmentation
- What data do we apply these augmentations to?

★ All data (✓)
★ Only the test data
★ Only the validation data
★ Only the training data ✓

Regularization Techniques - Data Augmentation

- We only want to be invariant to certain degrees of transformations
  - Like the human visual system; not these:

Regularization Techniques - Data Augmentation

- We only want to be invariant to certain degrees of transformations
  - Like the human visual system; not these:

Regularization Techniques - Data Augmentation

- We only want to be invariant to certain degrees of transformations
  - Like the human visual system; not these:

- Hyperparameters: how much of each perturbation to apply?
  - Usually specify a distribution to sample from
  - E.g., zero-mean 5-dimensional Gaussian with degrees of translation, scaling, reflection, rotation, stretching to apply to original data
  - Best hyperparameter setting: most helpful invariants

Regularization Techniques - Data Augmentation

- We only want to be invariant to certain degrees of transformations
  - Like the human visual system; not these:

- Hyperparameters: how much of each perturbation to apply?
  - Usually specify a distribution to sample from
  - E.g., zero-mean 5-dimensional Gaussian with degrees of translation, scaling, reflection, rotation, stretching to apply to original data
  - Best hyperparameter setting: most helpful invariants

- In computer vision, it is not hard to find useful image perturbations
- What could perturbations be in other applications? Domain-specific!
  - Could even lead to domain-specific insights about important invariants

Filtering Training Data

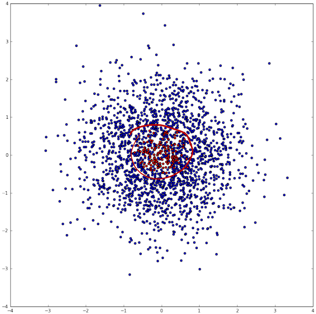
- Some training data points are much more helpful to learn the target concept than others
- Filtering means reducing the training set to the really important points that help adjusting the classification boundary/regression curve
- Techniques: oversampling, subsampling, outlier rejection, jittering
- Frequent problem: imbalanced data in classification

Balancing Data

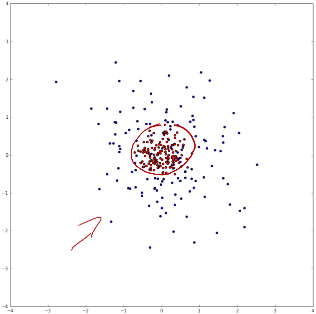
- E.g., 93% negative data, 7% positive
  - You could trivially get 93% accuracy
  - But you may want also want high recall

Balancing Data

- E.g., 93% negative data, 7% positive
  - You could trivially get 93% accuracy
  - But you may want also want high recall



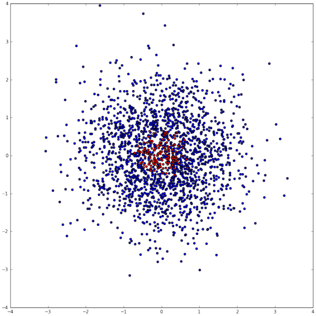
Original data



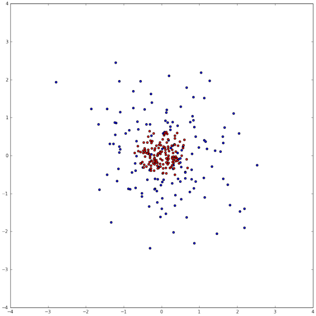
Subsampled majority class

Balancing Data

- E.g., 93% negative data, 7% positive
  - You could trivially get 93% accuracy
  - But you may want also want high recall



Original data



Subsampled majority class

- Solution 1: If your classifier supports it, weight your data
- Solution 2: Subsample the majority class
- Solution 3: Generate new data points of minority class

Regularization Techniques - Input Features

- Removing features may reduce overfitting by helping the model avoid fitting pseudo relationships
  - Extreme: think of a feature that is just random noise
- Dimensionality reduction is related: PCA, ICA

Regularization Techniques - Input Features

- Removing features may reduce overfitting by helping the model avoid fitting pseudo relationships
  - Extreme: think of a feature that is just random noise
- Dimensionality reduction is related: PCA, ICA
- Of course, adding features can also help improve performance
  - If they are related to the desired output
  - Very often, domain experts can come up with useful additional features
  - Can also add non-linear transformations of features

Regularization Techniques - Committee / Ensemble Approaches

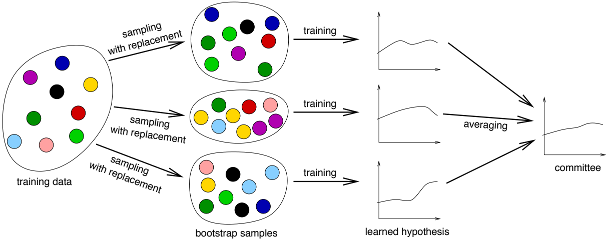
- If you ask one expert, the expert may fail
- Ask a committee of experts: the majority has a better chance to be right

Regularization Techniques - Committee / Ensemble Approaches

- If you ask one expert, the expert may fail
- Ask a committee of experts: the majority has a better chance to be right
- Premise: experts are experienced and diverse

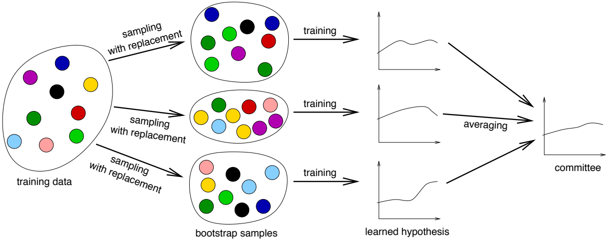
Committees - Bagging [Breimann, 1996]

- Train several models on bootstrap samples of the training data
  - Data is drawn randomly with replacements $\Rightarrow$ some data points may occur twice or more, others don't occur at all
- Average the output of all trained models



Committees - Bagging [Breimann, 1996]

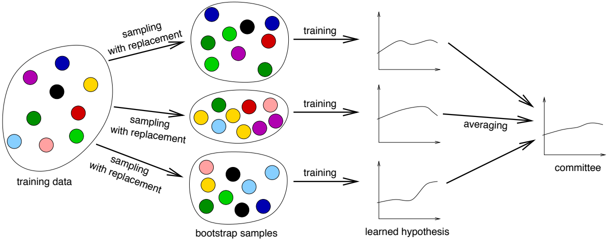
- Train several models on bootstrap samples of the training data
  - Data is drawn randomly with replacements $\Rightarrow$ some data points may occur twice or more, others don't occur at all
- Average the output of all trained models



- Single members of the committee might produce a higher test-set error; but the committee error can still improve

Committees - Bagging [Breimann, 1996]

- Train several models on bootstrap samples of the training data
  - Data is drawn randomly with replacements ⇒ some data points may occur twice or more, others don't occur at all
- Average the output of all trained models



- Single members of the committee might produce a higher test-set error; but the committee error can still improve
- We'll cover committee methods / ensemble methods in more detail in the module on tree-based methods

Lecture Overview

Metrics of Success 1/3

- In different applications, different metrics of success apply
- Example metrics for classification
  - E.g., 0/1 loss, higher loss for false positives than false negatives, etc
  - E.g., AUC, F1, precision/recall, ... (see next slide)

Metrics of Success 1/3

- In different applications, different metrics of success apply
- Example metrics for classification
  - E.g., 0/1 loss, higher loss for false positives than false negatives, etc
  - E.g., AUC, F1, precision/recall, ... (see next slide) , cross entropy
- Example metrics for regression
  - E.g., RMSE (root mean squared error)
  - E.g., MAE (mean absolute error)
  - Log-likelihood: $\log(P(\text{observations} \mid \text{model}))$

Metrics of Success 1/3

- In different applications, different metrics of success apply
- Example metrics for classification
  - E.g., 0/1 loss, higher loss for false positives than false negatives, etc
  - E.g., AUC, F1, precision/recall, . . . (see next slide)
- Example metrics for regression
  - E.g., RMSE (root mean squared error)
  - E.g., MAE (mean absolute error)
  - Log-likelihood: $\log(P(\text{observations} \mid \text{model}))$
- Internal loss function being optimized often differs from external metric of success
  - E.g., internally, to fit the model, we may need a differentiable surrogate loss function, such as cross entropy
  - E.g., 0/1 loss is not differentiable

Metrics of Success 2/3

- Binary classification: detecting a signal (e.g., disease)
  - True positive rate, sensitivity or recall:
    percentage of positive data points that are classified as positive
    $\rightarrow$ want this to be high

Metrics of Success 2/3

- Binary classification: detecting a signal (e.g., disease)
  - True positive rate, sensitivity or recall:
    percentage of positive data points that are classified as positive
    $\rightarrow$ want this to be high
  - False positive rate:
    percentage of negative data points that are classified as positive
    $\rightarrow$ want this to be low

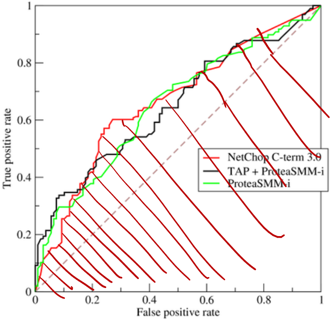
Metrics of Success 2/3

- Binary classification: detecting a signal (e.g., disease)
  - True positive rate, sensitivity or recall:
    percentage of positive data points that are classified as positive
    $\rightarrow$ want this to be high
  - False positive rate:
    percentage of negative data points that are classified as positive
    $\rightarrow$ want this to be low
  - Precision: percentage of positively-classified data points that are truly positive
    $\rightarrow$ want this to be high

Metrics of Success 2/3

- Binary classification: detecting a signal (e.g., disease)
  - True positive rate, sensitivity or recall:
    percentage of positive data points that are classified as positive
    $\rightarrow$ want this to be high
  - False positive rate:
    percentage of negative data points that are classified as positive
    $\rightarrow$ want this to be low
  - Precision: percentage of positively-classified data points that are truly positive
    $\rightarrow$ want this to be high
  - F-measure or F_1 score: $F = 2 \cdot (\text{precision} \cdot \text{recall}) / (\text{precision} + \text{recall})$

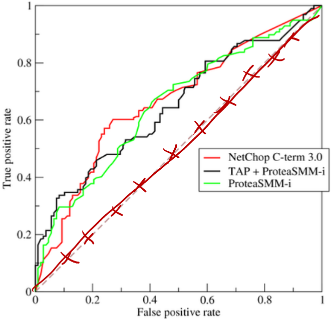
- Receiver operating characteristic (ROC curve)
  - Plots true positive rate vs. false positive rate
  - At various threshold settings of a classifier
  - Single number for a classifier: Area Under the ROC Curve (AUC)



ROC curve of three predictors of peptide cleaving in the proteasome.
Source: `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`

Metrics of Success 3/3

- Receiver operating characteristic (ROC curve)
  - Plots true positive rate vs. false positive rate
  - At various threshold settings of a classifier
  - Single number for a classifier: Area Under the ROC Curve (AUC)

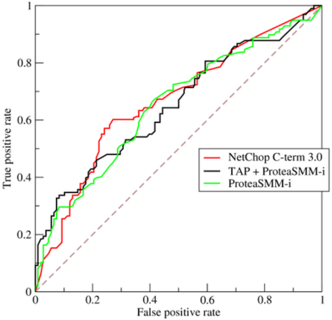


ROC curve of three predictors of peptide cleaving in the proteasome.
Source: `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`

- With which algorithm can you trivially achieve the diagonal? 🙋

Metrics of Success 3/3

- Receiver operating characteristic (ROC curve)
  - Plots true positive rate vs. false positive rate
  - At various threshold settings of a classifier
  - Single number for a classifier: Area Under the ROC Curve (AUC)



ROC curve of three predictors of peptide cleaving in the proteasome.
Source: `https://en.wikipedia.org/wiki/Receiver_operating_characteristic`

- With which algorithm can you trivially achieve the diagonal? ✋🏾✋🏽
- Another, similar type of curve: precision-recall curve

Metrics of Success: Take Home Message

- There are several metrics of success
- The correct metric depends on your application
- For example, for spam classification you should not only consider accuracy
  - The predictor should not classify an important message as spam
  - similar cases in medicine

Metrics of Success: Take Home Message

- There are several metrics of success
- The correct metric depends on your application
- For example, for spam classification you should not only consider accuracy
  - The predictor should not classify an important message as spam
  - similar cases in medicine
- In case of doubt, you should consider several metrics

The i.i.d. Assumption

- So far, we've assumed an underlying distribution $p(\mathbf{x}, y)$
- More precisely, we made the standard assumption in supervised learning: data points $p(\mathbf{x}, y)$ are independently and identically distributed (i.i.d.)

The i.i.d. Assumption

- So far, we've assumed an underlying distribution $p(\mathbf{x}, y)$
- More precisely, we made the standard assumption in supervised learning: data points $p(\mathbf{x}, y)$ are independently and identically distributed (i.i.d.)
  - Independent: learning the label A of one data point doesn't tell you anything about the label B of another data point: $P(x \geq A) = P(x \geq A \mid y \geq B)$, for all x and y

The i.i.d. Assumption

- So far, we've assumed an underlying distribution $p(\mathbf{x}, y)$
- More precisely, we made the standard assumption in supervised learning: data points $p(\mathbf{x}, y)$ are independently and identically distributed (i.i.d.)
  - Independent: learning the label A of one data point doesn't tell you anything about the label B of another data point: $P(x \geq A) = P(x \geq A \mid y \geq B)$, for all x and y
  - Identically distributed: data points (including their labels) are drawn from the same probability distribution: $P(x \geq A) = P(x \geq B)$, for all x

The i.i.d. Assumption

- So far, we've assumed an underlying distribution $p(\mathbf{x}, y)$
- More precisely, we made the standard assumption in supervised learning: data points $p(\mathbf{x}, y)$ are independently and identically distributed (i.i.d.)
  - Independent: learning the label A of one data point doesn't tell you anything about the label B of another data point:
    $P(x \geq A) = P(x \geq A \mid y \geq B)$, for all x and y
  - Identically distributed: data points (including their labels) are drawn from the same probability distribution:
    $P(x \geq A) = P(x \geq B)$, for all x
- In practice, especially independence can be broken in many ways
- Example: weather prediction: one data point per day
  - ★ Data points from consecutive days are independent.
  - ★ Data points from consecutive days are dependent. ✓

Validation for the Non-i.i.d. Setting

- We want to obtain an unbiased estimate of the performance that we would achieve in practice (on a hold-out private test set)

Validation for the Non-i.i.d. Setting

- We want to obtain an unbiased estimate of the performance that we would achieve in practice (on a hold-out private test set)
- Example: weather forecast
  - To predict tomorrow's weather, we cannot use data from the future
  - Standard option: when predicting for validation data point at time t, use training set up to $t - 1$

Validation for the Non-i.i.d. Setting

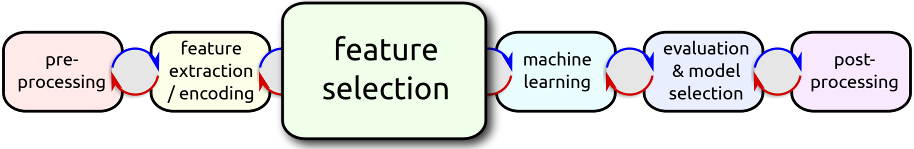
- We want to obtain an unbiased estimate of the performance that we would achieve in practice (on a hold-out private test set)
- Example: weather forecast
  - To predict tomorrow's weather, we cannot use data from the future
  - Standard option: when predicting for validation data point at time t, use training set up to $t-1$

- Example: 50 data points measured for each of 100 patients
- What would the right 10-fold cross-validation protocol be to get an unbiased estimate of how well we'll predict on a new patient?
  - ★ Split data randomly into 10 folds of 500 data points each
  - ★ In each fold: use 5 data points of each of the 100 patients 500
  - ★ In each fold: use all 50 data points of 10 patients each ✓ 500

Lecture Overview

1. Validation (cont'd)
2. Regularization
3. General Considerations
4. Model Selection and Feature Selection
5. Wrapup

Feature Selection



- Feature selection: pick a subset of features that performs best
- Exactly the same problem of generalization as for model selection
- Special mechanisms exist to evaluate feature importance, etc
  - Here a quick preview

Feature Selection: Forward Selection

Forward Selection

- Build up your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by adding each of the unused features
  2. Add the feature with the highest performance improvements

Feature Selection: Forward Selection

Forward Selection

- Build up your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by adding each of the unused features
  2. Add the feature with the highest performance improvements

$\rightsquigarrow$ small feature set, but dependencies between features are maybe missed

Feature Selection: Forward Selection

Forward Selection

- Build up your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by adding each of the unused features
  2. Add the feature with the highest performance improvements

$\rightsquigarrow$ small feature set, but dependencies between features are maybe missed

Example:

Iteration	selected	f_1	f_2	f_3	f_4
1st	$\{\}$	20.0	17.7	$\mathbf{30.2}$	20.1

Feature Selection: Forward Selection

Forward Selection

- Build up your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by adding each of the unused features
  2. Add the feature with the highest performance improvements

$\leadsto$ small feature set, but dependencies between features are maybe missed

Example:

Iteration	selected	f_1	f_2	f_3	f_4
1st	$\{\}$	20.0 >	17.7	$\mathbf{30.2}$	20.1
2nd	$\{f_3\}$	5.1 <	$\mathbf{8.2}$	–	4.9

Feature Selection: Forward Selection

Forward Selection

- Build up your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by adding each of the unused features
  2. Add the feature with the highest performance improvements

$\rightsquigarrow$ small feature set, but dependencies between features are maybe missed

Example:

Iteration	selected	f_1	f_2	f_3	f_4
1st	$\{\}$	20.0	17.7	$\mathbf{30.2}$	20.1
2nd	$\{f_3\}$	5.1	$\mathbf{8.2}$	–	4.9
3rd	$\{f_3, f_2\}$	$\mathbf{0.8}$	–	–	0.5

Feature Selection: Forward Selection

Forward Selection

- Build up your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by adding each of the unused features
  2. Add the feature with the highest performance improvements

$\rightsquigarrow$ small feature set, but dependencies between features are maybe missed

Example:

Iteration	selected	f_1	f_2	f_3	f_4
1st	$\{\}$	20.0	17.7	$\mathbf{30.2}$	20.1
2nd	$\{f_3\}$	5.1	$\mathbf{8.2}$	–	4.9
3rd	$\{f_3, f_2\}$	$\mathbf{0.8}$	–	–	0.5
4th	$\{f_3, f_2, f_1\}$	–	–	–	-0.2

Feature Selection: Forward Selection

Forward Selection

- Build up your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by adding each of the unused features
  2. Add the feature with the highest performance improvements

$\rightsquigarrow$ small feature set, but dependencies between features are maybe missed

Example:

Iteration	selected	f_1	f_2	f_3	f_4
1st	$\{\}$	20.0	17.7	$\mathbf{30.2}$	20.1
2nd	$\{f_3\}$	5.1	$\mathbf{8.2}$	–	4.9
3rd	$\{f_3, f_2\}$	$\mathbf{0.8}$	–	–	0.5
4th	$\{f_3, f_2, f_1\}$	–	–	–	-0.2

$\rightsquigarrow$ we could also consider to stop after the second iteration, if the improvement by adding f_1 is considered too small.

Feature Selection: Backward Elimination

Backward Elimination

- Reduce your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by removing each of the considered features
  2. Remove the feature with the smallest performance loss

$\rightsquigarrow$ "larger" feature set, but dependencies between features are preserved

Feature Selection: Backward Elimination

Backward Elimination

- Reduce your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by removing each of the considered features
  2. Remove the feature with the smallest performance loss

$\rightsquigarrow$ "larger" feature set, but dependencies between features are preserved

Example:

Iteration	selected	f_1	f_2	f_3	f_4
1st	$\{f_1, f_2, f_3, f_4\}$	$\mathbf{0.8}$	0.0	-10.0	0.2

Feature Selection: Backward Elimination

Backward Elimination

- Reduce your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by removing each of the considered features
  2. Remove the feature with the smallest performance loss

$\rightsquigarrow$ "larger" feature set, but dependencies between features are preserved

Example:

Iteration	selected	f_1	f_2	f_3	f_4
1st	$\{f_1, f_2, f_3, f_4\}$	$\mathbf{0.8}$	0.0	-10.0	0.2
2nd	$\{f_2, f_3, f_4\}$	**–**	$\mathbf{-0.5}$	-12.3	-2.3

Feature Selection: Backward Elimination

Backward Elimination

- Reduce your feature set step by step
- Iterate:
  1. Evaluate the predictor performance by removing each of the considered features
  2. Remove the feature with the smallest performance loss

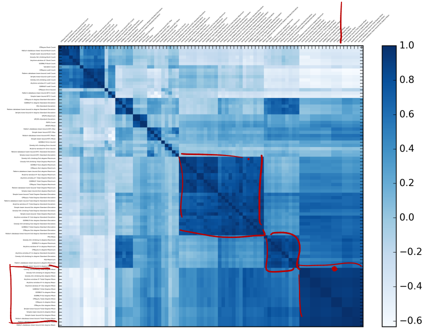
$\rightsquigarrow$ "larger" feature set, but dependencies between features are preserved

Example:

Iteration	selected	f_1	f_2	f_3	f_4
1st	$\{f_1, f_2, f_3, f_4\}$	$\mathbf{0.8}$	0.0	-10.0	0.2
2nd	$\{f_2, f_3, f_4\}$	–	$\mathbf{-0.5}$	-12.3	-2.3
3rd	$\{f_3, f_4\}$	–	–	-18.3	-8.5

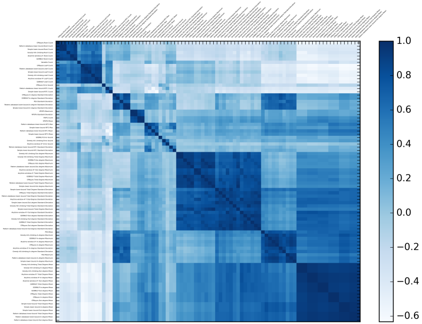
$\rightsquigarrow$ Forward selection and backward elimination often do not recover the same set of features.

Feature Selection: Correlation Analysis



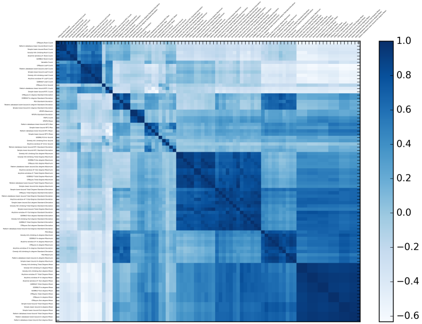
- highly correlated features often hurt the training process more than they help

Feature Selection: Correlation Analysis



- highly correlated features often hurt the training process more than they help
- you should consider to remove correlated features all but one
  - a PCA is implicitly doing something similar

Feature Selection: Correlation Analysis



- highly correlated features often hurt the training process more than they help
- you should consider to remove correlated features all but one
  - a PCA is implicitly doing something similar
- Pearson correlation coefficient

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

Lecture Overview

Summary by learning goals

Having heard this lecture, you can now . . .

- Explain how to check performance using cross-validation
- Identify over- and underfitting based on learning curves
- Explain different regularization approaches
- Select features (using basic approaches)
- Explain the i.i.d. assumption and where it can break down
- Handle imbalanced data
- Choose the right metric of success for a new application
- Explain the appropriate cross-validation splits for several settings