Machine Learning

Model Selection

Alberto Maria Metelli - Francesco Trovò

Definition of different models

What to do in the case the model you are considering is not performing well even by tuning properly the parameters (cross-validation)?

We have two opposite options:

- simplify the model $\rightarrow$ **model selection** (today)
- increase its complexity (next time)

Definition of different models

What to do in the case the model you are considering is not performing well even by tuning properly the parameters (cross-validation)?

We have two opposite options:

- simplify the model $\rightarrow$ **model selection** (today)
- increase its complexity (next time)

Definition of different models

What to do in the case the model you are considering is not performing well even by tuning properly the parameters (cross-validation)?

We have two opposite options:

- simplify the model $\rightarrow$ **model selection** (today)
- increase its complexity (next time)

How to Select a Model

We already discussed how to evaluate a specific model (bias/variance dilemma)

- Model Selection
  - Feature selection: choose only a subset of significant features to use
  - Feature extraction (Dimensionality reduction): project the features in another (lower) dimensional space
  - Regularization (shrinkage): introduce some penalization for complex models in the loss function
- Ensemble model
  - Bagging
  - Boosting

How to Select a Model

We already discussed how to evaluate a specific model (bias/variance dilemma)

- Model Selection
  - Feature selection: choose only a subset of significant features to use
  - Feature extraction (Dimensionality reduction): project the features in another (lower) dimensional space
  - Regularization (shrinkage): introduce some penalization for complex models in the loss function
- Ensemble model
  - Bagging
  - Boosting

How to Select a Model

We already discussed how to evaluate a specific model (bias/variance dilemma)

- Model Selection
  - Feature selection: choose only a subset of significant features to use
  - Feature extraction (Dimensionality reduction): project the features in another (lower) dimensional space
  - Regularization (shrinkage): introduce some penalization for complex models in the loss function
- Ensemble model
  - Bagging
  - Boosting

How to Select a Model

We already discussed how to evaluate a specific model (bias/variance dilemma)

- Model Selection
  - Feature selection: choose only a subset of significant features to use
  - Feature extraction (Dimensionality reduction): project the features in another (lower) dimensional space
  - Regularization (shrinkage): introduce some penalization for complex models in the loss function
- Ensemble model
  - Bagging
  - Boosting

How to Select a Model

We already discussed how to evaluate a specific model (bias/variance dilemma)

- Model Selection
  - Feature selection: choose only a subset of significant features to use
  - Feature extraction (Dimensionality reduction): project the features in another (lower) dimensional space
  - Regularization (shrinkage): introduce some penalization for complex models in the loss function
- Ensemble model
  - Bagging
  - Boosting

How to Select a Model

We already discussed how to evaluate a specific model (bias/variance dilemma)

- Model Selection
  - Feature selection: choose only a subset of significant features to use
  - Feature extraction (Dimensionality reduction): project the features in another (lower) dimensional space
  - Regularization (shrinkage): introduce some penalization for complex models in the loss function
- Ensemble model
  - Bagging
  - Boosting

How to Select a Model

We already discussed how to evaluate a specific model (bias/variance dilemma)

- Model Selection
  - Feature selection: choose only a subset of significant features to use
  - Feature extraction (Dimensionality reduction): project the features in another (lower) dimensional space
  - Regularization (shrinkage): introduce some penalization for complex models in the loss function
- Ensemble model
  - Bagging
  - Boosting

Model Selection

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - **PCA**
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - PCA
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - PCA
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - PCA
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - PCA
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - PCA
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - PCA
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - PCA
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - **PCA**
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - **PCA**
  - ICA
- Regularization (e.g., Ridge)

Model Selection

- Feature Selection
  - **Filter methods**
  - Embedded methods (e.g., Lasso)
  - Wrapper methods
    - Brute force
    - Forward Step-wise selection
    - **Backward step-wise selection**
- Feature Extraction
  - **PCA**
  - ICA
- Regularization (e.g., Ridge)

Feature Selection: Filter method

Correlation filter

- We have no hypothesis space of models as input
- For each feature $j \in \{1, \dots, M\}$ compute the **Pearson correlation coefficient** between x_k and the target y:

$$\hat{\rho}(x_j, y) = \frac{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)(y_n - \overline{y})}{\sqrt{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)^2}\sqrt{\sum_{n=1}^{N}(y_n - \overline{y})^2}}, \qquad \overline{x}_j = \frac{1}{N}\sum_{n=1}^{N} x_{j,n}, \quad \overline{y} = \frac{1}{N}\sum_{n=1}^{N} y_n.$$

- Select the features with **higher** Pearson correlation coefficient
- Captures only **linear** relationships between features and target
- There exist approaches for non-linear relationships (e.g., mutual information)

Feature Selection: Filter method

Correlation filter

- We have no hypothesis space of models as input
- For each feature $j \in \{1, \dots, M\}$ compute the **Pearson correlation coefficient** between x_k and the target y:

$$\hat{\rho}(x_j, y) = \frac{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)(y_n - \overline{y})}{\sqrt{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)^2}\sqrt{\sum_{n=1}^{N}(y_n - \overline{y})^2}}, \qquad \overline{x}_j = \frac{1}{N}\sum_{n=1}^{N} x_{j,n}, \quad \overline{y} = \frac{1}{N}\sum_{n=1}^{N} y_n.$$

- Select the features with **higher** Pearson correlation coefficient
- Captures only **linear** relationships between features and target
- There exist approaches for non-linear relationships (e.g., mutual information)

Feature Selection: Filter method

Correlation filter

- We have no hypothesis space of models as input
- For each feature $j \in \{1, \dots, M\}$ compute the **Pearson correlation coefficient** between x_k and the target y:

$$\hat{\rho}(x_j, y) = \frac{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)(y_n - \overline{y})}{\sqrt{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)^2}\sqrt{\sum_{n=1}^{N}(y_n - \overline{y})^2}}, \qquad \overline{x}_j = \frac{1}{N}\sum_{n=1}^{N} x_{j,n}, \quad \overline{y} = \frac{1}{N}\sum_{n=1}^{N} y_n.$$

- Select the features with **higher** Pearson correlation coefficient
- Captures only **linear** relationships between features and target
- There exist approaches for non-linear relationships (e.g., mutual information)

Feature Selection: Filter method

Correlation filter

- We have no hypothesis space of models as input
- For each feature $j \in \{1, \dots, M\}$ compute the **Pearson correlation coefficient** between x_k and the target y:

$$\hat{\rho}(x_j, y) = \frac{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)(y_n - \overline{y})}{\sqrt{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)^2}\sqrt{\sum_{n=1}^{N}(y_n - \overline{y})^2}}, \qquad \overline{x}_j = \frac{1}{N}\sum_{n=1}^{N} x_{j,n}, \quad \overline{y} = \frac{1}{N}\sum_{n=1}^{N} y_n.$$

- Select the features with **higher** Pearson correlation coefficient
- Captures only **linear** relationships between features and target
- There exist approaches for non-linear relationships (e.g., mutual information)

Feature Selection: Filter method

Correlation filter

- We have no hypothesis space of models as input
- For each feature $j \in \{1, \ldots, M\}$ compute the **Pearson correlation coefficient** between x_k and the target y:

$$\hat{\rho}(x_j, y) = \frac{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)(y_n - \overline{y})}{\sqrt{\sum_{n=1}^{N}(x_{j,n} - \overline{x}_j)^2}\sqrt{\sum_{n=1}^{N}(y_n - \overline{y})^2}}, \qquad \overline{x}_j = \frac{1}{N}\sum_{n=1}^{N} x_{j,n}, \quad \overline{y} = \frac{1}{N}\sum_{n=1}^{N} y_n.$$

- Select the features with **higher** Pearson correlation coefficient
- Captures only **linear** relationships between features and target
- There exist approaches for non-linear relationships (e.g., mutual information)

Feature Selection: Wrapper Methods

Brute force

- We have a hypothesis space of models $\mathcal{H}$ as input
- For each k number of features $k \in \{1, \ldots, M\}$
  - Learn all the possible $\binom{M}{k}$ possible models within $\mathcal{H}$ with k inputs
  - Select the model with the smallest loss
- Select the number of features k providing the model with the smallest loss
- Warning: model selection should be done appropriately (e.g., cross-validation)
- Problem: if M is large enough the computation of all the models is **unfeasible** (combinatorial complexity)

Feature Selection: Wrapper Methods

Brute force

- We have a hypothesis space of models $\mathcal{H}$ as input
- For each k number of features $k \in \{1, \ldots, M\}$
  - Learn all the possible $\binom{M}{k}$ possible models within $\mathcal{H}$ with k inputs
  - Select the model with the smallest loss
- Select the number of features k providing the model with the smallest loss

- Warning: model selection should be done appropriately (e.g., cross-validation)
- Problem: if M is large enough the computation of all the models is **unfeasible** (combinatorial complexity)

Feature Selection: Wrapper Methods

Brute force

- We have a hypothesis space of models $\mathcal{H}$ as input
- For each k number of features $k \in \{1, \ldots, M\}$
  - Learn all the possible $\binom{M}{k}$ possible models within $\mathcal{H}$ with k inputs
  - Select the model with the smallest loss
- Select the number of features k providing the model with the smallest loss

- Warning: model selection should be done appropriately (e.g., cross-validation)
- Problem: if M is large enough the computation of all the models is **unfeasible** (combinatorial complexity)

Feature Selection: Wrapper Methods

Brute force

- We have a hypothesis space of models $\mathcal{H}$ as input
- For each k number of features $k \in \{1, \dots, M\}$
  - Learn all the possible $\binom{M}{k}$ possible models within $\mathcal{H}$ with k inputs
  - Select the model with the smallest loss
- Select the number of features k providing the model with the smallest loss

- Warning: model selection should be done appropriately (e.g., cross-validation)
- Problem: if M is large enough the computation of all the models is **unfeasible** (combinatorial complexity)

Feature Selection: Wrapper Methods

Brute force

- We have a hypothesis space of models $\mathcal{H}$ as input
- For each k number of features $k \in \{1, \dots, M\}$
    - Learn all the possible $\binom{M}{k}$ possible models within $\mathcal{H}$ with k inputs
    - Select the model with the smallest loss
- Select the number of features k providing the model with the smallest loss

- Warning: model selection should be done appropriately (e.g., cross-validation)
- Problem: if M is large enough the computation of all the models is **unfeasible** (combinatorial complexity)

Feature Selection: Wrapper Methods

Brute force

- We have a hypothesis space of models $\mathcal{H}$ as input
- For each k number of features $k \in \{1, \dots, M\}$
  - Learn all the possible $\binom{M}{k}$ possible models within $\mathcal{H}$ with k inputs
  - Select the model with the smallest loss
- Select the number of features k providing the model with the smallest loss

- Warning: model selection should be done appropriately (e.g., cross-validation)
- Problem: if M is large enough the computation of all the models is **unfeasible** (combinatorial complexity)

Feature Selection: Wrapper Methods

Brute force

- We have a hypothesis space of models $\mathcal{H}$ as input
- For each k number of features $k \in \{1, \dots, M\}$
    - Learn all the possible $\binom{M}{k}$ possible models within $\mathcal{H}$ with k inputs
    - Select the model with the smallest loss
- Select the number of features k providing the model with the smallest loss

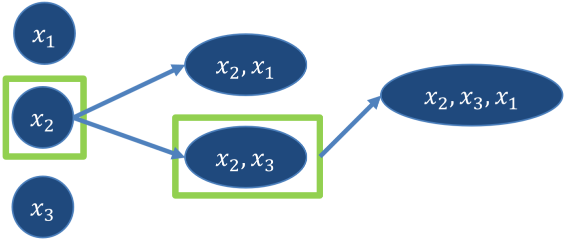
- Warning: model selection should be done appropriately (e.g., cross-validation)
- Problem: if M is large enough the computation of all the models is **unfeasible** (combinatorial complexity)

Feature Selection: Wrapper Methods

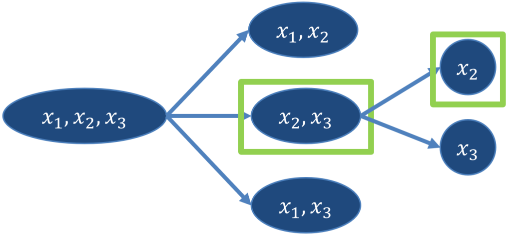
Step-wise selection

We evaluate only a subset of the possible models

Forward step-wise selection



Backward step-wise selection

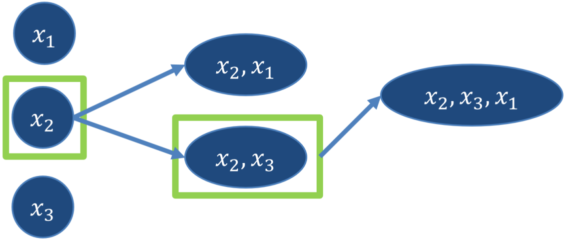


Feature Selection: Wrapper Methods

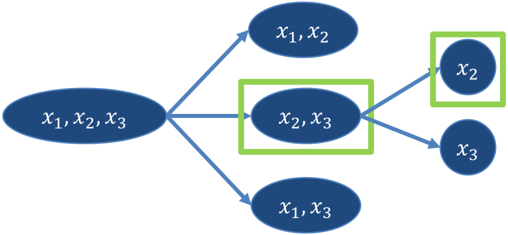
Step-wise selection

We evaluate only a subset of the possible models

Forward step-wise selection



Backward step-wise selection



Backward step-wise selection on the Iris Dataset (1)

- Assume the problem is to discriminate between Virginica and Non-Virginica iris
- We select a performance index: validation accuracy on 20% of the data
- Train a model on the full data $(x_1, x_2, x_3, x_4)^\top$: Logistic regression
- Remove one of the features and check the error:
  - Model with $(x_1, x_2, x_3)^\top$: accuracy 1
  - Model with $(x_1, x_3, x_4)^\top$: accuracy 1
  - Model with $(x_1, x_2, x_4)^\top$: accuracy 1
  - Model with $(x_2, x_3, x_4)^\top$: accuracy 1
- Removing a single feature does not change the model performance

Backward step-wise selection on the Iris Dataset (1)

- Assume the problem is to discriminate between Virginica and Non-Virginica iris
- We select a performance index: validation accuracy on 20% of the data
- Train a model on the full data $(x_1, x_2, x_3, x_4)^\top$: Logistic regression
- Remove one of the features and check the error:
  - Model with $(x_1, x_2, x_3)^\top$: accuracy 1
  - Model with $(x_1, x_3, x_4)^\top$: accuracy 1
  - Model with $(x_1, x_2, x_4)^\top$: accuracy 1
  - Model with $(x_2, x_3, x_4)^\top$: accuracy 1
- Removing a single feature does not change the model performance

Backward step-wise selection on the Iris Dataset (1)

- Assume the problem is to discriminate between Virginica and Non-Virginica iris
- We select a performance index: validation accuracy on 20% of the data
- Train a model on the full data $(x_1, x_2, x_3, x_4)^\top$: Logistic regression
- Remove one of the features and check the error:
  - Model with $(x_1, x_2, x_3)^\top$: accuracy 1
  - Model with $(x_1, x_3, x_4)^\top$: accuracy 1
  - Model with $(x_1, x_2, x_4)^\top$: accuracy 1
  - Model with $(x_2, x_3, x_4)^\top$: accuracy 1
- Removing a single feature does not change the model performance

Backward step-wise selection on the Iris Dataset (1)

- Assume the problem is to discriminate between Virginica and Non-Virginica iris
- We select a performance index: validation accuracy on 20% of the data
- Train a model on the full data $(x_1, x_2, x_3, x_4)^\top$: Logistic regression
- Remove one of the features and check the error:
  - Model with $(x_1, x_2, x_3)^\top$: accuracy 1
  - Model with $(x_1, x_3, x_4)^\top$: accuracy 1
  - Model with $(x_1, x_2, x_4)^\top$: accuracy 1
  - Model with $(x_2, x_3, x_4)^\top$: accuracy 1
- Removing a single feature does not change the model performance

Backward Feature selection on the Iris Dataset (2)

- Let us remove one of the features at random x_4
- Remove another feature and check the error:
  - Model with $(x_1, x_2)^\top$: accuracy 0.96
  - Model with $(x_1, x_3)^\top$: accuracy 0.96
  - Model with $(x_2, x_3)^\top$: accuracy 1
- The model with (x_2, x_3) is performing better than the others
- Iterate *one more time*

Backward Feature selection on the Iris Dataset (2)

- Let us remove one of the features at random x_4
- Remove another feature and check the error:
  - Model with $(x_1, x_2)^\top$: accuracy 0.96
  - Model with $(x_1, x_3)^\top$: accuracy 0.96
  - Model with $(x_2, x_3)^\top$: accuracy 1
- The model with (x_2, x_3) is performing better than the others
- Iterate *one more time*

Backward Feature selection on the Iris Dataset (2)

- Let us remove one of the features at random x_4
- Remove another feature and check the error:
  - Model with $(x_1, x_2)^\top$: accuracy 0.96
  - Model with $(x_1, x_3)^\top$: accuracy 0.96
  - Model with $(x_2, x_3)^\top$: accuracy 1
- The model with (x_2, x_3) is performing better than the others
- Iterate *one more time*

Backward Feature selection on the Iris Dataset (2)

- Let us remove one of the features at random x_4
- Remove another feature and check the error:
  - Model with $(x_1, x_2)^\top$: accuracy 0.96
  - Model with $(x_1, x_3)^\top$: accuracy 0.96
  - Model with $(x_2, x_3)^\top$: accuracy 1
- The model with (x_2, x_3) is performing better than the others
- Iterate *one more time*

Backward Feature selection on the Iris Dataset (2)

- Let us remove one of the features at random x_4
- Remove another feature and check the error:
  - Model with $(x_1, x_2)^\top$: accuracy 0.96
  - Model with $(x_1, x_3)^\top$: accuracy 0.96
  - Model with $(x_2, x_3)^\top$: accuracy 1
- The model with (x_2, x_3) is performing better than the others
- Iterate *one more time*

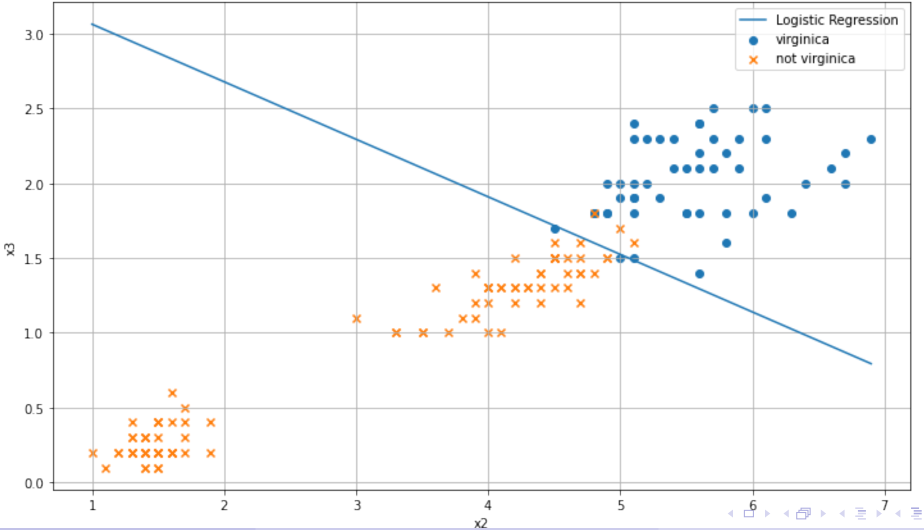
Backward Feature selection on the Iris Dataset (2)

- Let us remove one of the features at random x_4
- Remove another feature and check the error:
  - Model with $(x_1, x_2)^\top$: accuracy 0.96
  - Model with $(x_1, x_3)^\top$: accuracy 0.96
  - Model with $(x_2, x_3)^\top$: accuracy 1
- The model with (x_2, x_3) is performing better than the others
- Iterate *one more time*

Backward Feature selection on the Iris Dataset (2)

- Let us remove one of the features at random x_4
- Remove another feature and check the error:
  - Model with $(x_1, x_2)^\top$: accuracy 0.96
  - Model with $(x_1, x_3)^\top$: accuracy 0.96
  - Model with $(x_2, x_3)^\top$: accuracy 1
- The model with (x_2, x_3) is performing better than the others
- Iterate *one more time*

Results on the Iris Dataset



Feature Extraction: Principal Component Analysis (PCA)

- unsupervised **dimensionality reduction** technique
    - extract some low dimensional features from a dataset
- perform a **linear transformation** of the original data $\mathbf{X}$
    - the largest variance lies on the first transformed feature
    - the second largest variance on the second transformed feature
    - ...
- At last, we only keep some of the features we extract

Feature Extraction: Principal Component Analysis (PCA)

- unsupervised **dimensionality reduction** technique
  - extract some low dimensional features from a dataset
- perform a **linear transformation** of the original data $\mathbf{X}$
  - the largest variance lies on the first transformed feature
  - the second largest variance on the second transformed feature
  - ...
- At last, we only keep some of the features we extract

Feature Extraction: Principal Component Analysis (PCA)

- unsupervised **dimensionality reduction** technique
  - extract some low dimensional features from a dataset
- perform a **linear transformation** of the original data $\mathbf{X}$
  - the largest variance lies on the first transformed feature
  - the second largest variance on the second transformed feature
  - ...
- At last, we only keep some of the features we extract

Feature Extraction: Principal Component Analysis (PCA)

- unsupervised **dimensionality reduction** technique
  - extract some low dimensional features from a dataset
- perform a **linear transformation** of the original data $\mathbf{X}$
  - the largest variance lies on the first transformed feature
  - the second largest variance on the second transformed feature
  - ...
- At last, we only keep some of the features we extract

Feature Extraction: Principal Component Analysis (PCA)

- unsupervised **dimensionality reduction** technique
  - extract some low dimensional features from a dataset
- perform a **linear transformation** of the original data $\mathbf{X}$
  - the largest variance lies on the first transformed feature
  - the second largest variance on the second transformed feature
  - ...
- At last, we only keep some of the features we extract

Feature Extraction: Principal Component Analysis (PCA)

- unsupervised **dimensionality reduction** technique
  - extract some low dimensional features from a dataset
- perform a **linear transformation** of the original data $\mathbf{X}$
  - the largest variance lies on the first transformed feature
  - the second largest variance on the second transformed feature
  - ...
- At last, we only keep some of the features we extract

Feature Extraction: Principal Component Analysis (PCA)

- unsupervised **dimensionality reduction** technique
  - extract some low dimensional features from a dataset
- perform a **linear transformation** of the original data $\mathbf{X}$
  - the largest variance lies on the first transformed feature
  - the second largest variance on the second transformed feature
  - $\dots$
- At last, we only keep some of the features we extract

Procedure

- Translate the original data $\mathbf{X}$ to $\tilde{\mathbf{X}}$ s.t. they have zero mean
- Compute the covariance matrix of $\tilde{\mathbf{X}}$, $\mathbf{C} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$
- The eigenvectors of $\mathbf{C}$ are the **principal components**
  - The eigenvector $\mathbf{e}_1$ corresponding to the largest eigenvalue λ_1 is the first principal component
  - The eigenvector $\mathbf{e}_2$ corresponding to the second largest eigenvalue λ_2 is the second principal component
  - ...
- The computation of the eigenvectors can be done with Singular Value Decomposition (SVD)

Procedure

- Translate the original data $\mathbf{X}$ to $\tilde{\mathbf{X}}$ s.t. they have zero mean
- Compute the covariance matrix of $\tilde{\mathbf{X}}$, $\mathbf{C} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$
- The eigenvectors of $\mathbf{C}$ are the **principal components**
  - The eigenvector $\mathbf{e}_1$ corresponding to the largest eigenvalue λ_1 is the first principal component
  - The eigenvector $\mathbf{e}_2$ corresponding to the second largest eigenvalue λ_2 is the second principal component
  - ...
- The computation of the eigenvectors can be done with Singular Value Decomposition (SVD)

Procedure

- Translate the original data $\mathbf{X}$ to $\tilde{\mathbf{X}}$ s.t. they have zero mean
- Compute the covariance matrix of $\tilde{\mathbf{X}}$, $\mathbf{C} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$
- The eigenvectors of $\mathbf{C}$ are the **principal components**
  - The eigenvector $\mathbf{e}_1$ corresponding to the largest eigenvalue λ_1 is the first principal component
  - The eigenvector $\mathbf{e}_2$ corresponding to the second largest eigenvalue λ_2 is the second principal component
  - $\dots$
- The computation of the eigenvectors can be done with Singular Value Decomposition (SVD)

Procedure

- Translate the original data $\mathbf{X}$ to $\tilde{\mathbf{X}}$ s.t. they have zero mean
- Compute the covariance matrix of $\tilde{\mathbf{X}}$, $\mathbf{C} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$
- The eigenvectors of $\mathbf{C}$ are the **principal components**
  - The eigenvector $\mathbf{e}_1$ corresponding to the largest eigenvalue λ_1 is the first principal component
  - The eigenvector $\mathbf{e}_2$ corresponding to the second largest eigenvalue λ_2 is the second principal component
  - ...
- The computation of the eigenvectors can be done with Singular Value Decomposition (SVD)

Procedure

- Translate the original data $\mathbf{X}$ to $\tilde{\mathbf{X}}$ s.t. they have zero mean
- Compute the covariance matrix of $\tilde{\mathbf{X}}$, $\mathbf{C} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$
- The eigenvectors of $\mathbf{C}$ are the **principal components**
    - The eigenvector $\mathbf{e}_1$ corresponding to the largest eigenvalue λ_1 is the first principal component
    - The eigenvector $\mathbf{e}_2$ corresponding to the second largest eigenvalue λ_2 is the second principal component
    - ...
- The computation of the eigenvectors can be done with Singular Value Decomposition (SVD)

Procedure

- Translate the original data $\mathbf{X}$ to $\tilde{\mathbf{X}}$ s.t. they have zero mean
- Compute the covariance matrix of $\tilde{\mathbf{X}}$, $\mathbf{C} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$
- The eigenvectors of $\mathbf{C}$ are the **principal components**
  - The eigenvector $\mathbf{e}_1$ corresponding to the largest eigenvalue λ_1 is the first principal component
  - The eigenvector $\mathbf{e}_2$ corresponding to the second largest eigenvalue λ_2 is the second principal component
  - $\ldots$
- The computation of the eigenvectors can be done with Singular Value Decomposition (SVD)

Procedure

- Translate the original data $\mathbf{X}$ to $\tilde{\mathbf{X}}$ s.t. they have zero mean
- Compute the covariance matrix of $\tilde{\mathbf{X}}$, $\mathbf{C} = \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$
- The eigenvectors of $\mathbf{C}$ are the **principal components**
  - The eigenvector $\mathbf{e}_1$ corresponding to the largest eigenvalue λ_1 is the first principal component
  - The eigenvector $\mathbf{e}_2$ corresponding to the second largest eigenvalue λ_2 is the second principal component
  - $\ldots$
- The computation of the eigenvectors can be done with Singular Value Decomposition (SVD)

Given a sample vector $\tilde{\mathbf{x}}$, its transformed version $\mathbf{t}$ can be computed using:

$$\mathbf{T} = \tilde{\mathbf{X}}\mathbf{W}$$

- **loadings**: $\mathbf{W} = (\mathbf{e}_1|\mathbf{e}_2|\dots|\mathbf{e}_M)$ matrix of the principal components
- **scores**: $\mathbf{W}$ transformation of the input dataset $\tilde{\mathbf{X}}$
- **variance**: $(\lambda_1, \dots, \lambda_M)^\top$ vector of the variance of principal components

Given a sample vector $\tilde{\mathbf{x}}$, its transformed version $\mathbf{t}$ can be computed using:

$$\mathbf{T} = \tilde{\mathbf{X}}\mathbf{W}$$

- **loadings**: $\mathbf{W} = (\mathbf{e}_1|\mathbf{e}_2|\dots|\mathbf{e}_M)$ matrix of the principal components
- **scores**: $\mathbf{W}$ transformation of the input dataset $\tilde{\mathbf{X}}$
- **variance**: $(\lambda_1, \dots, \lambda_M)^\top$ vector of the variance of principal components

Given a sample vector $\tilde{\mathbf{x}}$, its transformed version $\mathbf{t}$ can be computed using:

$$\mathbf{T} = \tilde{\mathbf{X}}\mathbf{W}$$

- **loadings**: $\mathbf{W} = (\mathbf{e}_1|\mathbf{e}_2|\dots|\mathbf{e}_M)$ matrix of the principal components
- **scores**: $\mathbf{W}$ transformation of the input dataset $\tilde{\mathbf{X}}$
- **variance**: $(\lambda_1, \dots, \lambda_M)^\top$ vector of the variance of principal components

How Many Features

There are a few different methods to determine how many feature to choose

- Keep all the principal components until we have a **cumulative variance** of 90%-95%

$$\text{cumulative variance with } k \text{ components} = \frac{\sum_{j=1}^{k} \lambda_i}{\sum_{j=1}^{M} \lambda_i}$$

- Keep all the principal components which have more than 5% of variance (discard only those which have low variance)
- Find the **elbow** in the cumulative variance

How Many Features

There are a few different methods to determine how many feature to choose

- Keep all the principal components until we have a **cumulative variance** of 90%-95%

$$\text{cumulative variance with } k \text{ components} = \frac{\sum_{j=1}^{k} \lambda_i}{\sum_{j=1}^{M} \lambda_i}$$

- Keep all the principal components which have more than 5% of variance (discard only those which have low variance)
- Find the **elbow** in the cumulative variance

How Many Features

There are a few different methods to determine how many feature to choose

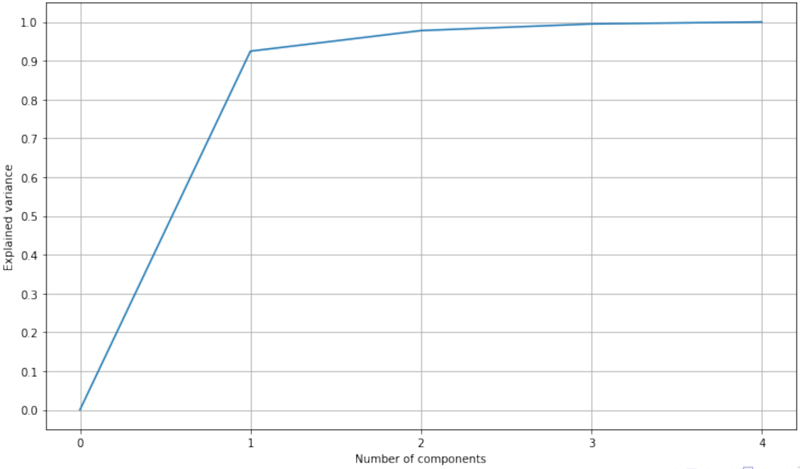
- Keep all the principal components until we have a **cumulative variance** of 90%-95%

$$\text{cumulative variance with } k \text{ components} = \frac{\sum_{j=1}^{k} \lambda_i}{\sum_{j=1}^{M} \lambda_i}$$

- Keep all the principal components which have more than 5% of variance (discard only those which have low variance)
- Find the **elbow** in the cumulative variance

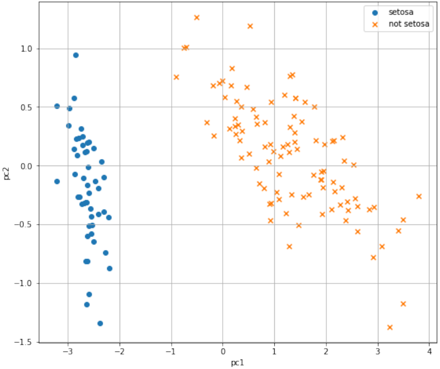
Cumulated Variance Plot

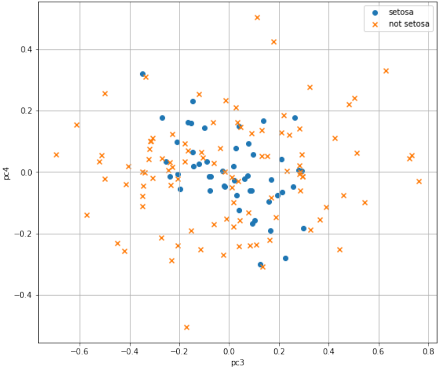
Using the Iris dataset inputs:



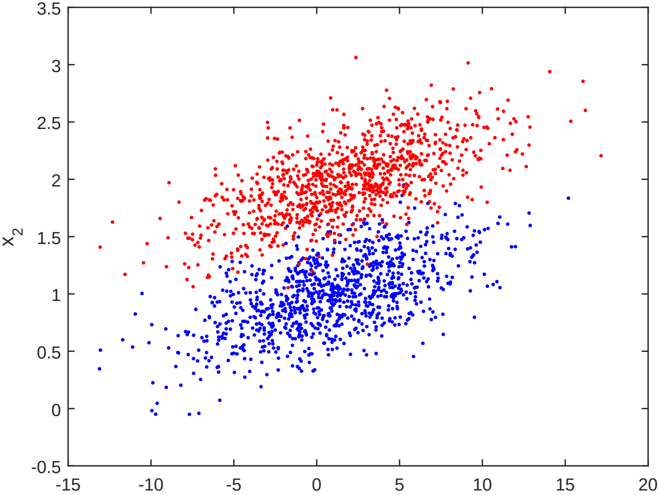
Principal Components

If we separate the first two components from the second twos:





Simpson's Paradox



PCA Different Purposes

- **Feature Extraction**: reduce the dimensionality of the dataset by selecting only the number of principal components retaining information about the problem
- **Compression**: keep the first k principal components and get $\mathbf{T}_k = \tilde{\mathbf{X}}\mathbf{W}_k$. The linear transformation $\mathbf{W}_k$ minimizes the **reconstruction error**:

$$\min_{\mathbf{W}_k \in \mathbb{R}^{M \times k}} \|\mathbf{T}\mathbf{W}_k^\top - \tilde{\mathbf{X}}\|_2^2$$

- **Data visualization**: reduce the dimensionality of the input dataset to 2 or 3 to be able to visualize the data

PCA Different Purposes

- **Feature Extraction**: reduce the dimensionality of the dataset by selecting only the number of principal components retaining information about the problem
- **Compression**: keep the first k principal components and get $\mathbf{T}_k = \tilde{\mathbf{X}}\mathbf{W}_k$. The linear transformation $\mathbf{W}_k$ minimizes the **reconstruction error**:

$$\min_{\mathbf{W}_k \in \mathbb{R}^{M \times k}} \|\mathbf{T}\mathbf{W}_k^\top - \tilde{\mathbf{X}}\|_2^2$$

- **Data visualization**: reduce the dimensionality of the input dataset to 2 or 3 to be able to visualize the data

PCA Different Purposes

- **Feature Extraction**: reduce the dimensionality of the dataset by selecting only the number of principal components retaining information about the problem
- **Compression**: keep the first k principal components and get $\mathbf{T}_k = \tilde{\mathbf{X}}\mathbf{W}_k$. The linear transformation $\mathbf{W}_k$ minimizes the **reconstruction error**:

$$\min_{\mathbf{W}_k \in \mathbb{R}^{M \times k}} \|\mathbf{T}\mathbf{W}_k^\top - \tilde{\mathbf{X}}\|_2^2$$

- **Data visualization**: reduce the dimensionality of the input dataset to 2 or 3 to be able to visualize the data

Regularization

Already known regularization procedure:

- Ridge:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- Lasso:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_1$$

- Elastic net:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda_1}{2}\|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2}\|\mathbf{w}\|_1$$

- They can be applied to the linear regression techniques, it can be extended for other methods
- For classification we will see some specific methods

Regularization

Already known regularization procedure:

- Ridge:
$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$
- Lasso:
$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_1$$
- Elastic net:
$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda_1}{2}\|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2}\|\mathbf{w}\|_1$$

- They can be applied to the linear regression techniques, it can be extended for other methods
- For classification we will see some specific methods

Regularization

Already known regularization procedure:

- Ridge:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- Lasso:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_1$$

- Elastic net:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda_1}{2}\|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2}\|\mathbf{w}\|_1$$

- They can be applied to the linear regression techniques, it can be extended for other methods
- For classification we will see some specific methods

Regularization

Already known regularization procedure:

- Ridge:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- Lasso:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_1$$

- Elastic net:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda_1}{2}\|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2}\|\mathbf{w}\|_1$$

- They can be applied to the linear regression techniques, it can be extended for other methods
- For classification we will see some specific methods

Regularization

Already known regularization procedure:

- Ridge:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- Lasso:

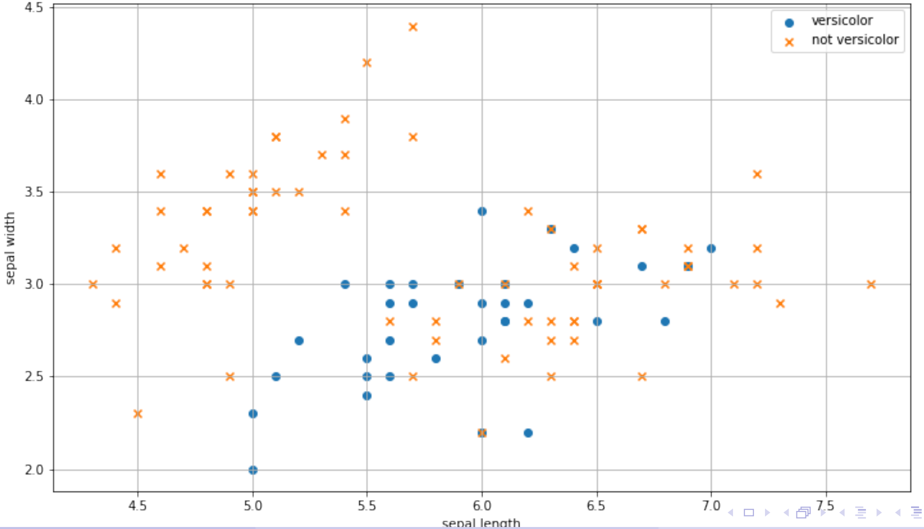
$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_1$$

- Elastic net:

$$L(\mathbf{w}) = \frac{1}{2}\mathrm{RSS}(\mathbf{w}) + \frac{\lambda_1}{2}\|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2}\|\mathbf{w}\|_1$$

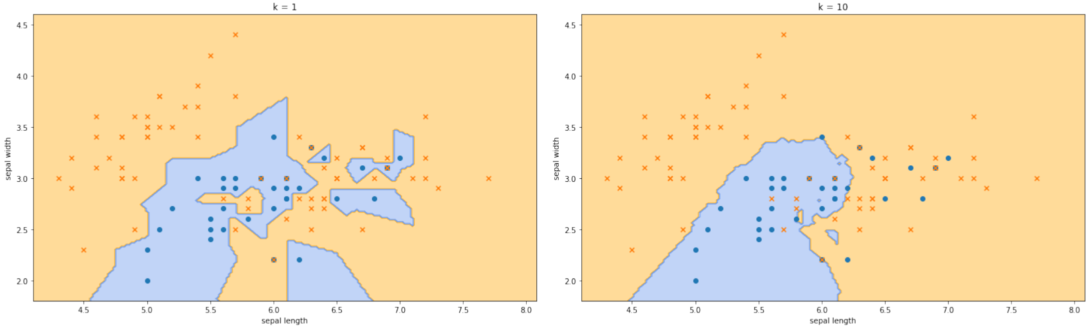
- They can be applied to the linear regression techniques, it can be extended for other methods
- For classification we will see some specific methods

A hard problem



K-Nearest Neighbour

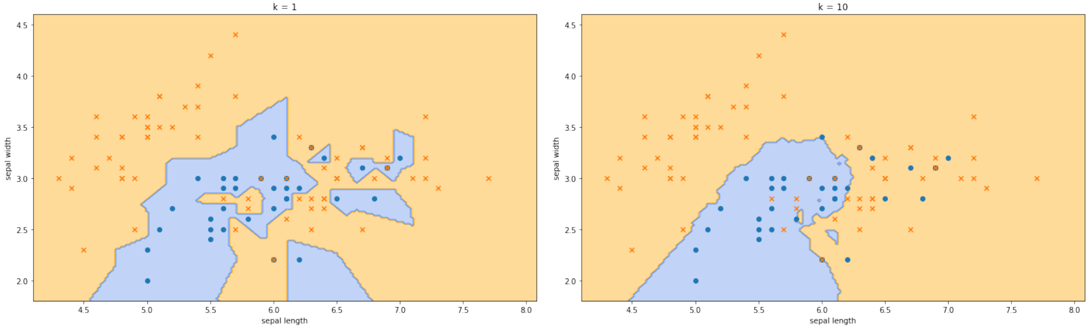
Different values of the K parameter



The larger the value of K, the more the model is regularized ($1/K$ acts as a regularization hyperparameter)

K-Nearest Neighbour

Different values of the K parameter



The larger the value of K, the more the model is regularized ($1/K$ acts as a regularization hyperparameter)

Ensemble Methods

Bagging

- **Goal**: achieve a **small variance** without increasing the bias
- This is achieved by **training (possibly) in parallel** N learners:
  1. Generate a dataset applying random sampling with replacement (**bootstrapping**)
  2. Train the model on the dataset
- To compute the **prediction** for new samples, apply all the trained models and combine the outputs with **majority voting** (classification) or **averaging**
- Bagging is generally helpful and reduce the variance, although the **sampled datasets are not independent**
- It helps with **unstable learners**, i.e., learners that change significantly with even small changes in the dataset (low bias and high variance) (regression)

Bagging

- **Goal**: achieve a **small variance** without increasing the bias
- This is achieved by **training (possibly) in parallel** N learners:
  1. Generate a dataset applying random sampling with replacement (**bootstrapping**)
  2. Train the model on the dataset
- To compute the **prediction** for new samples, apply all the trained models and combine the outputs with **majority voting** (classification) or **averaging**
- Bagging is generally helpful and reduce the variance, although the **sampled datasets are not independent**
- It helps with **unstable learners**, i.e., learners that change significantly with even small changes in the dataset (low bias and high variance) (regression)

Bagging

- **Goal**: achieve a **small variance** without increasing the bias
- This is achieved by **training (possibly) in parallel** N learners:
  1. Generate a dataset applying random sampling with replacement (**bootstrapping**)
  2. Train the model on the dataset
- To compute the **prediction** for new samples, apply all the trained models and combine the outputs with **majority voting** (classification) or **averaging**
- Bagging is generally helpful and reduce the variance, although the **sampled datasets are not independent**
- It helps with **unstable learners**, i.e., learners that change significantly with even small changes in the dataset (low bias and high variance) (regression)

Bagging

- **Goal**: achieve a **small variance** without increasing the bias
- This is achieved by **training (possibly) in parallel** N learners:
  1. Generate a dataset applying random sampling with replacement (**bootstrapping**)
  2. Train the model on the dataset
- To compute the **prediction** for new samples, apply all the trained models and combine the outputs with **majority voting** (classification) or **averaging**
- Bagging is generally helpful and reduce the variance, although the **sampled datasets are not independent**
- It helps with **unstable learners**, i.e., learners that change significantly with even small changes in the dataset (low bias and high variance) (regression)

Bagging

- **Goal**: achieve a **small variance** without increasing the bias
- This is achieved by **training (possibly) in parallel** N learners:
  1. Generate a dataset applying random sampling with replacement (**bootstrapping**)
  2. Train the model on the dataset
- To compute the **prediction** for new samples, apply all the trained models and combine the outputs with **majority voting** (classification) or **averaging**
- Bagging is generally helpful and reduce the variance, although the **sampled datasets are not independent**
- It helps with **unstable learners**, i.e., learners that change significantly with even small changes in the dataset (low bias and high variance) (regression)

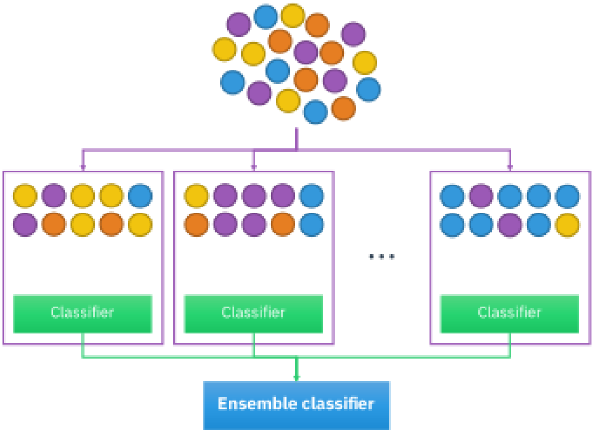
Bagging

- **Goal**: achieve a **small variance** without increasing the bias
- This is achieved by **training (possibly) in parallel** N learners:
  1. Generate a dataset applying random sampling with replacement (**bootstrapping**)
  2. Train the model on the dataset
- To compute the **prediction** for new samples, apply all the trained models and combine the outputs with **majority voting** (classification) or **averaging**
- Bagging is generally helpful and reduce the variance, although the **sampled datasets are not independent**
- It helps with **unstable learners**, i.e., learners that change significantly with even small changes in the dataset (low bias and high variance) (regression)

Bagging

- **Goal**: achieve a **small variance** without increasing the bias
- This is achieved by **training (possibly) in parallel** N learners:
  1. Generate a dataset applying random sampling with replacement (**bootstrapping**)
  2. Train the model on the dataset
- To compute the **prediction** for new samples, apply all the trained models and combine the outputs with **majority voting** (classification) or **averaging**
- Bagging is generally helpful and reduce the variance, although the **sampled datasets are not independent**
- It helps with **unstable learners**, i.e., learners that change significantly with even small changes in the dataset (low bias and high variance) (regression)

Bagging



Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

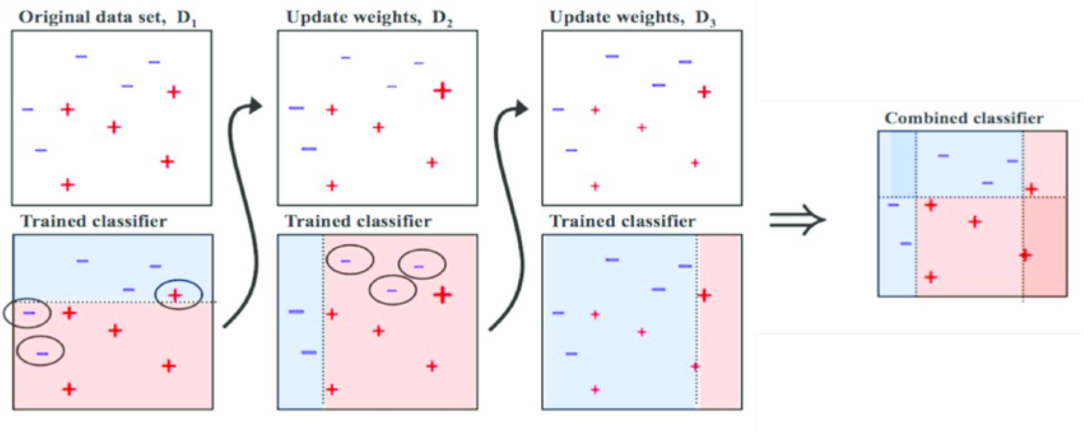
Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

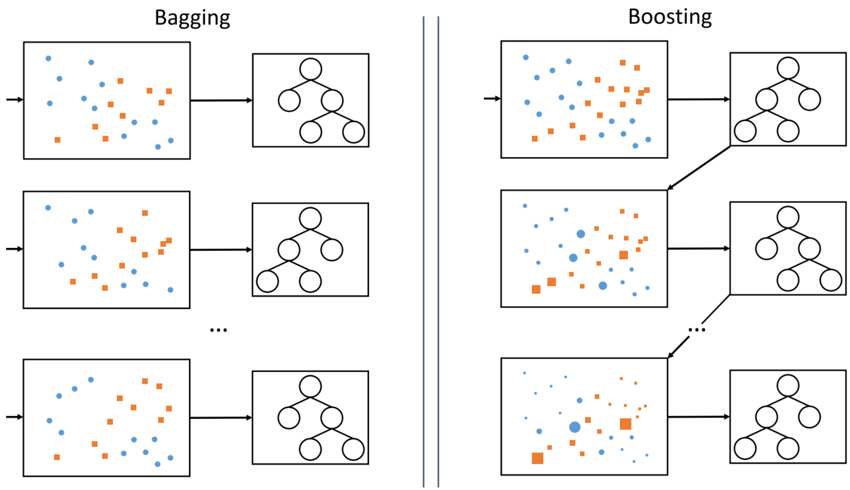
Boosting

- **Goal**: achieve a **small bias** by using on simple (**weak**) learners
- At the same time, using simple learners, aims at keeping a **small variance**
- This is achieved by **sequentially training** weak learners:
  1. Give an **equal weight** to all the samples in the training set
  2. Train a weak learner on the weighted training set
  3. Compute the error of the trained model on the **weighted training set**
  4. **Increase the weights** of samples missclassfied by the model
  5. Repeat from 2 until some criteria is met
- The ensemble of models learned can be applied on new samples by computing the **weighted prediction** of each model (more accurate models weight more)

Boosting


Original data set, D1
Update weights, D2
Update weights, D3
Trained classifier
Trained classifier
Trained classifier
Combined classifier

Bagging vs Boosting



Bagging vs Boosting

Bagging

- Reduces variance
- Not good for stable learners
- Can be applied with noisy data
- Usually helps but the difference might be small
- Parallel

Boosting

- Reduces bias (generally without overfitting)
- Works with stable learners
- Might have problem with noisy data
- Not always helps but it can makes the difference
- Sequential