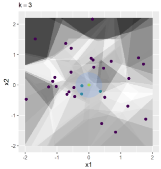
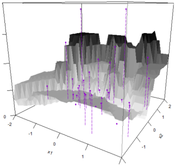
Introduction to Machine Learning



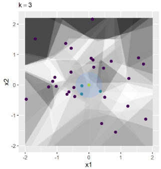


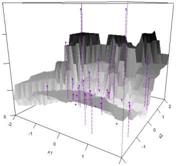
Agenda:

- *k*-NN for regression and classification
- Measures for Regression
- Measures for Classification
- Underfitting and Overfitting
- Introduction of Unsupervised learning

Introduction to Machine Learning

k-Nearest Neighbors





Learning goals

- Understand the basic idea of *k*-NN for regression and classification
- Understand that *k*-NN is a non-parametric, local model
- Know different distance measures for different scales of feature variables

K-NEAREST-NEIGHBORS

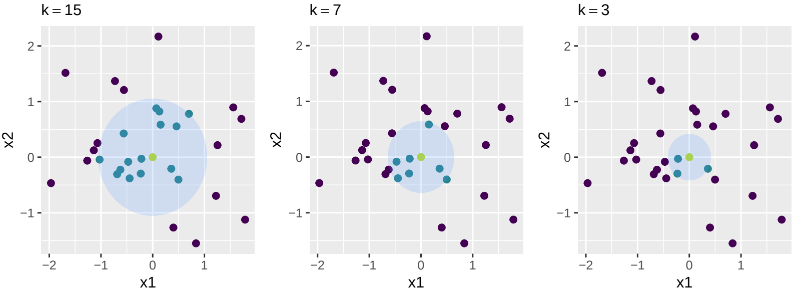
k-**NN** can be used for regression and classification.

Generates "similar" predictions for $\mathbf{x}$ to its k closest neighbors.

"Closeness" requires a distance or similarity measure.

The subset of $\mathcal{D}_{\text{train}}$ that is at least as close to $\mathbf{x}$ as its k-th closest neighbor $\mathbf{x}^{(k)}$ in $\mathcal{D}_{\text{train}}$ is called the k-**neighborhood** $N_k(\mathbf{x})$ of $\mathbf{x}$:

$$N_k(\mathbf{x}) = \{\mathbf{x}^{(i)} \in \mathcal{D}_{\text{train}} \mid d(\mathbf{x}^{(i)}, \mathbf{x}) \leq d(\mathbf{x}^{(k)}, \mathbf{x})\}$$



DISTANCE MEASURES

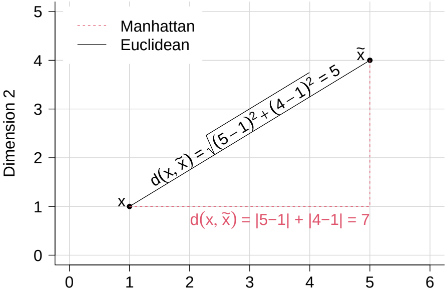
Popular for numerical features: **Minkowski** distances of the form

$$\|\mathbf{x} - \tilde{\mathbf{x}}\|_q = \left(\sum_{j=1}^{p} |x_j - \tilde{x}_j|^q\right)^{\frac{1}{q}} \quad \text{for } \mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X} \text{ with } p \text{ numeric features}$$

Especially, **Manhattan** ($q = 1$) and **Euclidean** ($q = 2$) distance

$$d_{Manhattan}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|_1 = \sum_{j=1}^{p} |x_j - \tilde{x}_j|$$

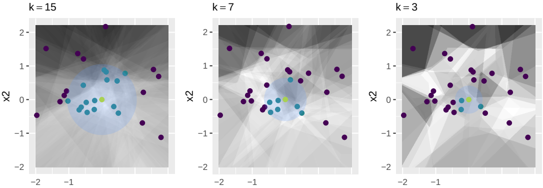
$$d_{Euclidean}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|_2 = \sqrt{\sum_{j=1}^{p} (x_j - \tilde{x}_j)^2}$$



PREDICTION - REGRESSION

Compute for each point the average output y of the k-nearest neighbours in $N_k(\mathbf{x})$:

$$\hat{f}(\mathbf{x}) = \frac{1}{k} \sum_{i:\mathbf{x}^{(i)} \in N_k(\mathbf{x})} y^{(i)} \text{ or } \hat{f}(\mathbf{x})$$



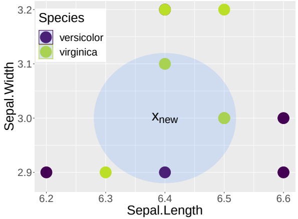
PREDICTION - CLASSIFICATION

For classification in g groups, a majority vote is used:

$$\hat{h}(\mathbf{x}) = \underset{\ell \in \{1,\dots,g\}}{\arg\max} \sum_{i:\mathbf{x}^{(i)} \in N_k(\mathbf{x})} \mathbb{I}(y^{(i)} = \ell)$$

And posterior probabilities can be estimated with:

$$\hat{\pi}_\ell(\mathbf{x}) = \frac{1}{k} \sum_{i:\mathbf{x}^{(i)} \in N_k(\mathbf{x})} \mathbb{I}(y^{(i)} = \ell)$$

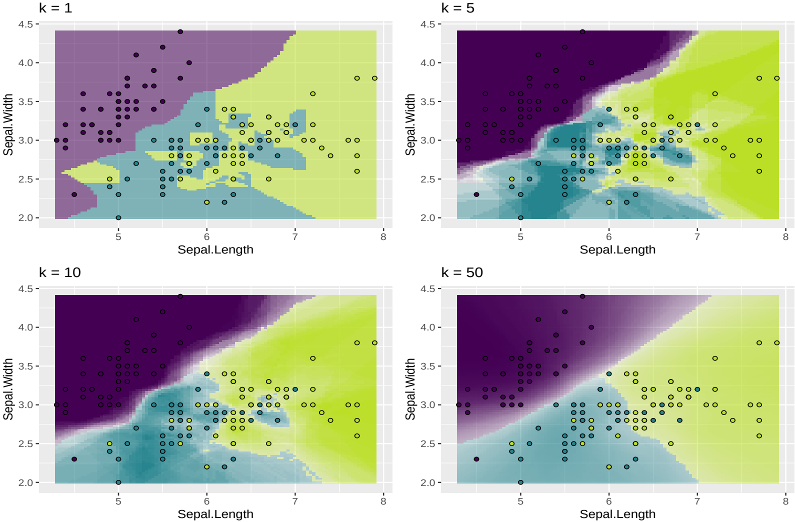


	SL	SW	Species	dist
52	6.4	3.2	versicolor	0.200
59	6.6	2.9	versicolor	0.224
75	6.4	2.9	versicolor	0.100
76	6.6	3.0	versicolor	0.200
98	6.2	2.9	versicolor	0.224
104	6.3	2.9	virginica	0.141
105	6.5	3.0	virginica	0.100
111	6.5	3.2	virginica	0.224
116	6.4	3.2	virginica	0.200
117	6.5	3.0	virginica	0.100
138	6.4	3.1	virginica	0.100
148	6.5	3.0	virginica	0.100

Example with subset of iris data ($k = 3$)

$\hat{\pi}_{setosa}(\mathbf{x}_{new}) = \frac{0}{3} = 0\%$, $\hat{\pi}_{versicolor}(\mathbf{x}_{new}) = \frac{1}{3} = 33\%$, $\hat{\pi}_{virginica}(\mathbf{x}_{new}) = \frac{2}{3} = 67\%$,

K-NN: FROM SMALL TO LARGE K



Complex, local model vs smoother, more global model

K-NN SUMMARY

k-NN is a lazy classifier, it has no real training step, it simply stores the complete data - which are needed during prediction.

Hence, its parameters are the training data, there is no real compression of information.

As the number of parameters grows with the number of training points, we call *k*-NN a non-parametric model

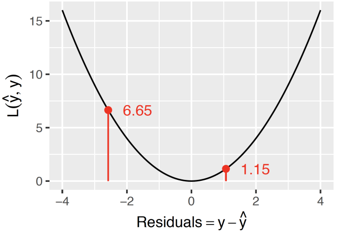
k-NN is not based on any distributional or functional assumption, and can, in theory, model data situations of arbitrary complexity.

The smaller *k*, the less stable, less smooth and more “wiggly” the decision boundary becomes.

Accuracy of *k*-NN can be severely degraded by the presence of noisy or irrelevant features, or when the feature scales are not consistent with their importance.

Introduction to Machine Learning

Evaluation: Measures for Regression



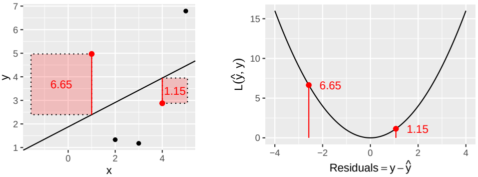
Learning goals

- Know the definitions of mean squared error (MSE) and mean absolute error (MAE)
- Understand the connections of MSE and MAE to L2 and L1 loss
- Know the definition of Spearman's ρ
- Know the definitions of R^2 and generalized R^2

MEAN SQUARED ERROR (MSE)

$$\rho_{MSE}(\mathbf{y}, \boldsymbol{F}) = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2 \in [0; \infty) \qquad \rightarrow L2 \text{ loss.}$$

Outliers with large prediction error heavily influence the MSE, as they enter quadratically.



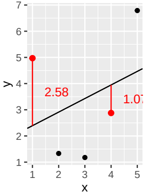
Similar measures:

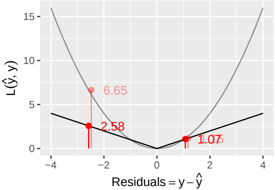
- Sum of squared errors: $\rho_{SSE}(\mathbf{y}, \boldsymbol{F}) = \sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$
- Root MSE (orig. scale): $\rho_{RMSE}(\mathbf{y}, \boldsymbol{F}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2}$

MEAN ABSOLUTE ERROR

$$\rho_{MAE}(\mathbf{y}, \boldsymbol{F}) = \frac{1}{m}\sum_{i=1}^{m}|y^{(i)} - \hat{y}^{(i)}| \in [0; \infty) \qquad \rightarrow L1 \text{ loss.}$$

More robust, less influenced by large residuals, more intuitive than MSE.





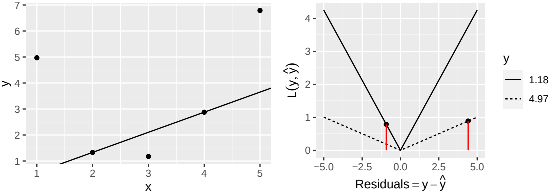
Similar measures:

- Median absolute error (for even more robustness)

MEAN ABSOLUTE PERCENTAGE ERROR

$$\rho_{MAPE}(\mathbf{y}, \boldsymbol{F}) = \frac{1}{m} \sum_{i=1}^{m} \left| \frac{y^{(i)} - \hat{y}^{(i)}}{y^{(i)}} \right| \in [0; \infty)$$

Small $|y|$ influence more strongly. Cannot handle $y = 0$.



Similar measures:

- Mean Absolute Scaled Error (MASE)
- Symmetric Mean Absolute Percentage Error (sMAPE)

METRICS FOR CLASSIFICATION

For hard-label classification, the confusion matrix is a useful representation:

		True Class y	
		$+$	$-$
Pred.	$+$	True Positive (TP)	False Positive (FP)
$\hat{y}$	$-$	False Negative (FN)	True Negative (TN)

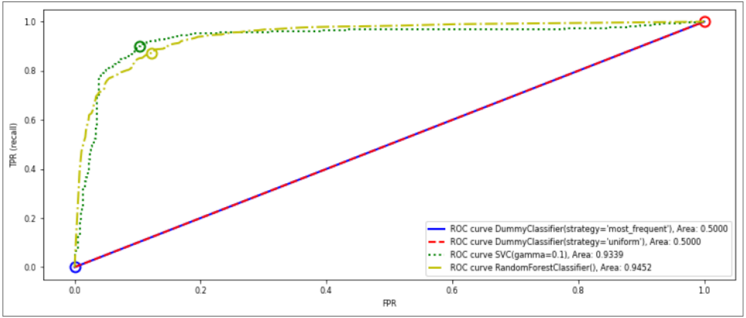
From this matrix a variety of evaluation metrics, including precision and recall, can be computed

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

RECEIVER OPERATING CHARACTERISTICS

- Trade off *true positive rate* $TPR = \frac{TP}{TP+FN}$ with *false positive rate* $FPR = \frac{FP}{FP+TN}$
- Plotting TPR against FPR *for all possible thresholds* yields a *Receiver Operating Characteristics curve*
    - Change the treshold until you find a sweet spot in the TPR-FPR trade-off
    - Lower thresholds yield higher TPR (recall), higher FPR, and vice versa



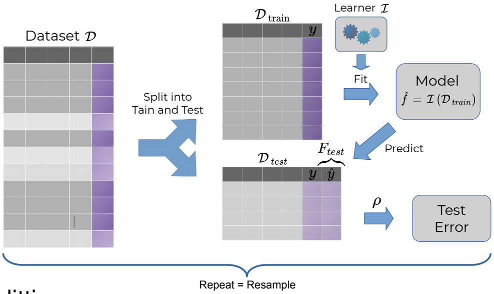
- Each point on the curve represents a pair of TPR/FPR values corresponding to a particular decision threshold
- The closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test

TRAINING ERROR and TEST ERROR

- Training error: Error on training data

- Test error: Error on test data

- Partition data, e.g., 2/3 for train and 1/3 for test.



A.k.a. holdout splitting.

RESAMPLING

There exist a few well established resampling strategies:

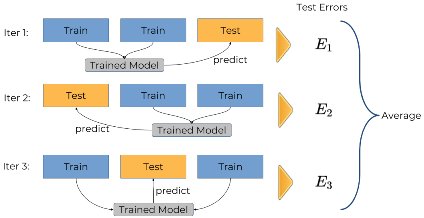
- (Repeated) Hold-out / Subsampling
- Cross validation
- Bootstrap

All methods aim to generate $\mathcal{J}$ by splitting the full data set (repeatedly) into a train and test set. The model is trained on the respective train set and evaluated on the test set.

CROSS-VALIDATION

- Split the data into k roughly equally-sized partitions.
- Each part is test set once, join $k-1$ parts for training.
- Obtain k test errors and average.
- Fraction $(k-1)/k$ is used for training, so 90% for 10CV
- Each observation is tested exactly once.

Example: 3-fold CV

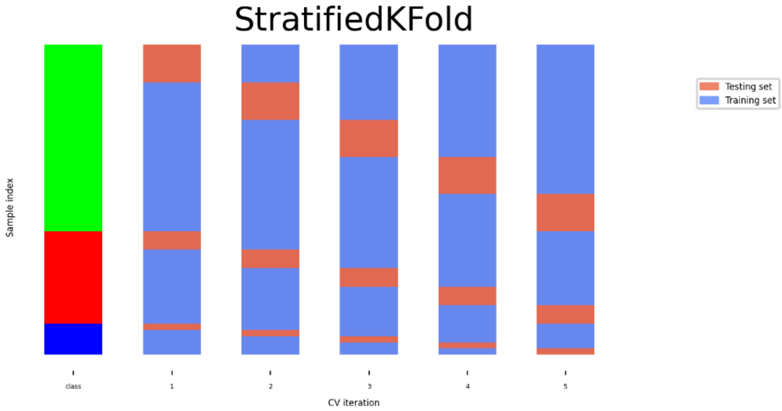


CROSS-VALIDATION - STRATIFICATION

Used when target classes are very imbalanced

Proportions between classes are conserved in each fold

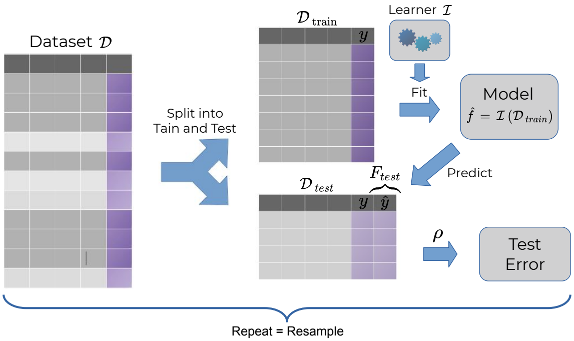
For classes: simply CV-split the class data, then join



SUBSAMPLING

Repeated hold-out with averaging, a.k.a. Monte Carlo CV.

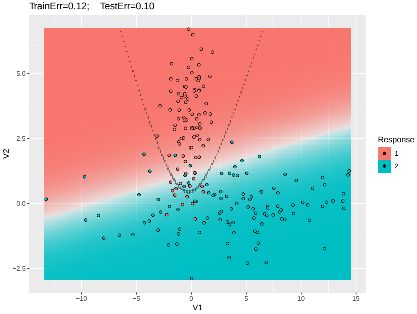
Typical choices for splitting: $\frac{4}{5}$ or $\frac{9}{10}$ for training.



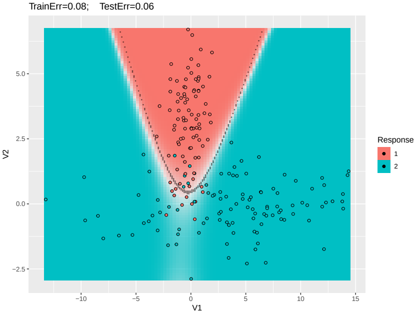
UNDERFITTING

- Occurs if model does not reflect true shape of underlying function
- Hence, predictions will be less good as they could be
- High train error and high test error
- Hard to detect, as we don't know what the Bayes error is for a task

Underfitted model



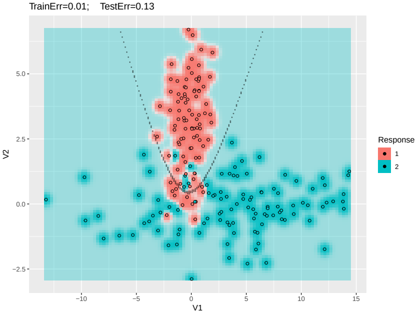
Appropriate model



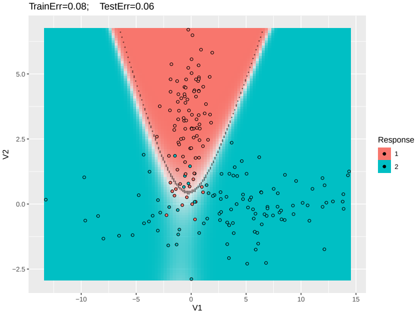
OVERFITTING

- Overfitting occurs when the model reflects noise or artifacts in training data, which do not generalize
- Small train error, at cost of test high error
- Hence, predictions of overfitting models cannot be trusted - but proper ML evaluation workflows should make it visible

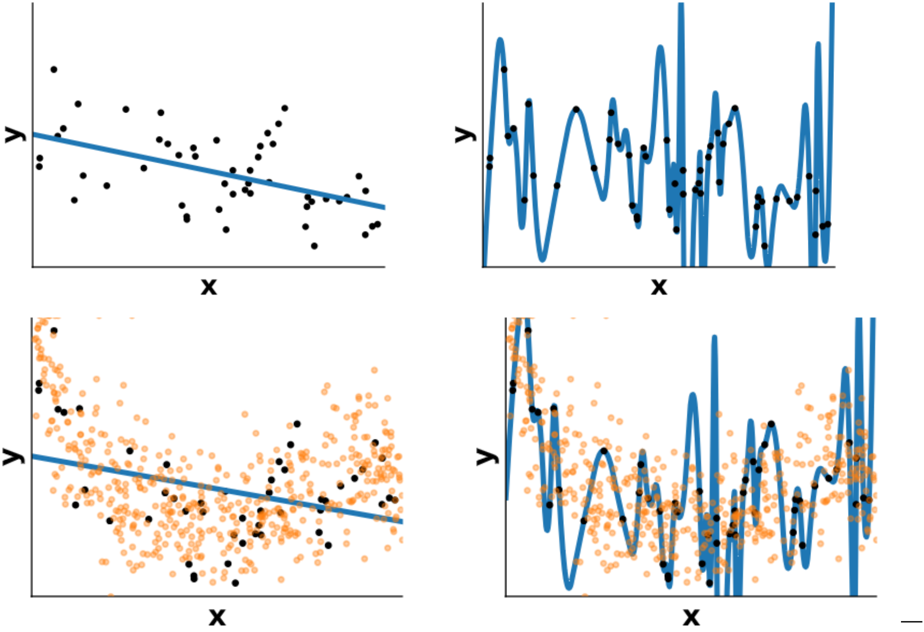
Overfitted model



Appropriate model

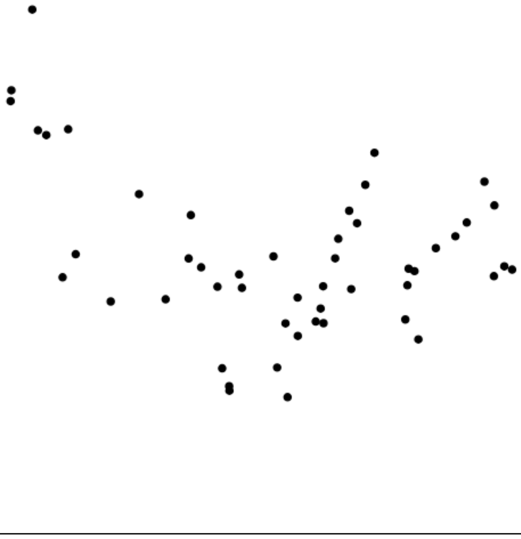


UNDER- AND OVERFITTING IN REGRESSION



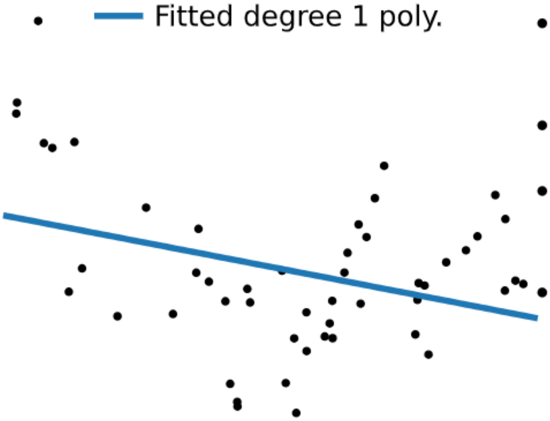
On new data

Varing model complexity



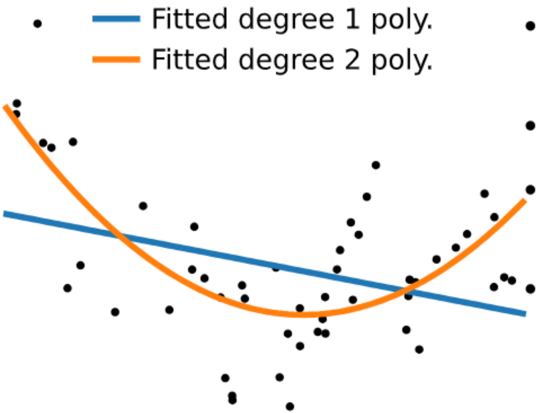
- Data generated by a random process
- This process is unknown
- We can only access the observations

Varing model complexity



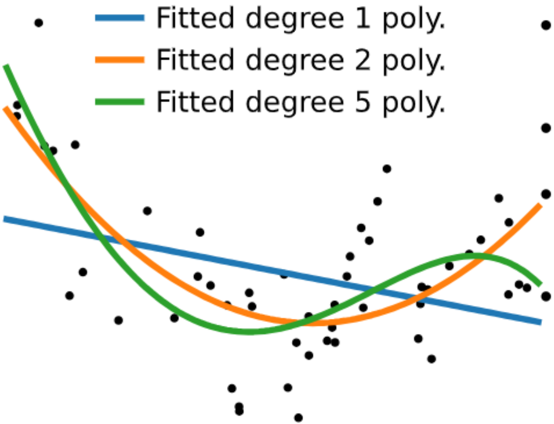
- Data generated by a random process
- This process is unknown
- We can only access the observations
- Fit polynomials of various degrees

Varing model complexity



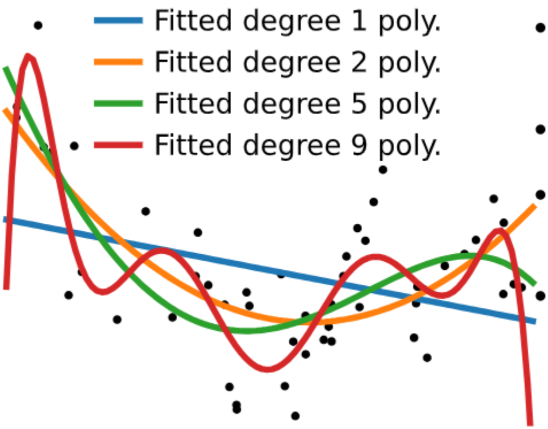
- Data generated by a random process
- This process is unknown
- We can only access the observations
- Fit polynomials of various degrees

Varing model complexity



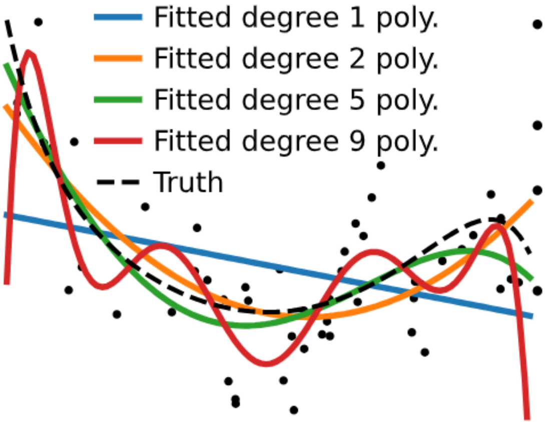
- Data generated by a random process
- This process is unknown
- We can only access the observations
- Fit polynomials of various degrees

Varing model complexity



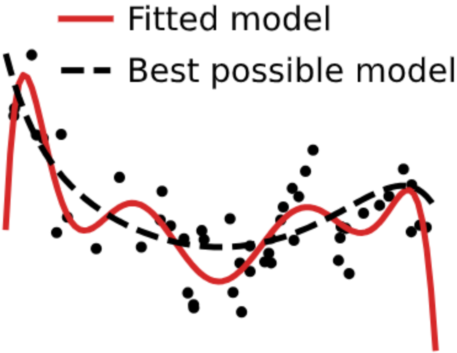
- Data generated by a random process
- This process is unknown
- We can only access the observations
- Fit polynomials of various degrees

Varing model complexity



- Data generated by a random process
- This process is unknown
- We can only access the observations
- Fit polynomials of various degrees

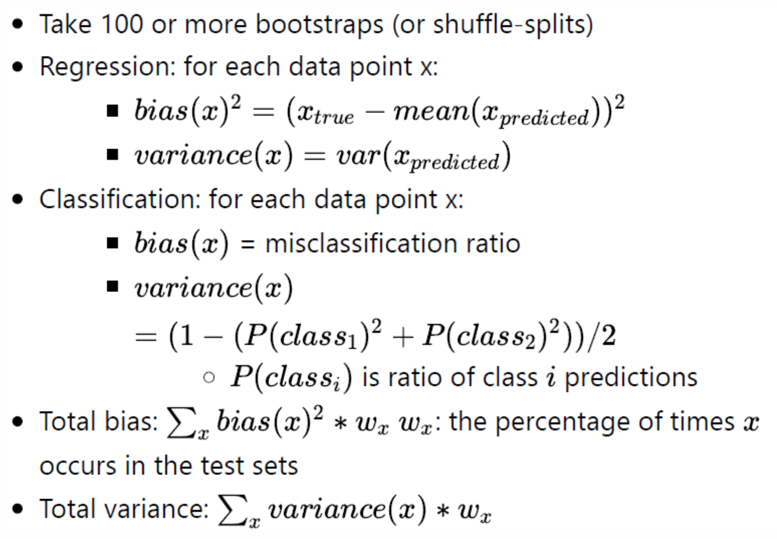
Overfit: Model too complex



Model too complex for the data:

- Its best possible fit would approximate well the generative process
- However, its flexibility captures noise

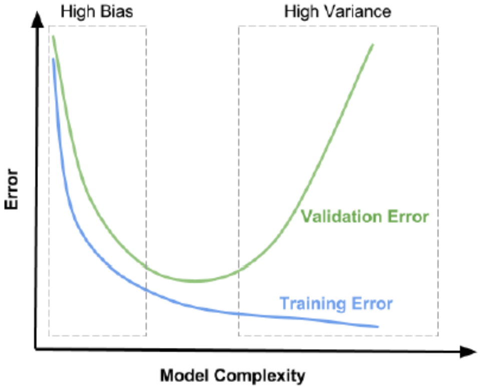
Bias and Variance



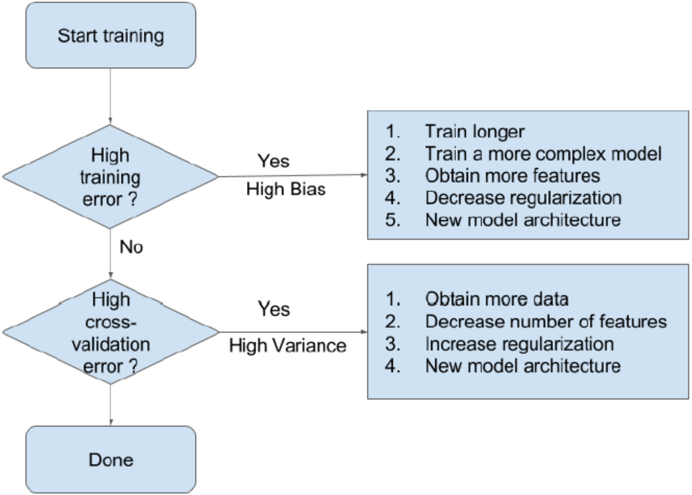
- Take 100 or more bootstraps (or shuffle-splits)
- Regression: for each data point x:
  - $bias(x)^2 = (x_{true} - mean(x_{predicted}))^2$
  - $variance(x) = var(x_{predicted})$
- Classification: for each data point x:
  - $bias(x)$ = misclassification ratio
  - $variance(x)$
    $= (1 - (P(class_1)^2 + P(class_2)^2))/2$
    - $P(class_i)$ is ratio of class i predictions
- Total bias: $\sum_x bias(x)^2 * w_x$ w_x: the percentage of times x occurs in the test sets
- Total variance: $\sum_x variance(x) * w_x$

Bias and Variance, Underfitting and Overfitting

- High variance means that you are likely overfitting
    - Use more regularization or use a simpler model
- High bias means that you are likely underfitting
    - Do less regularization or use a more flexible/complex model



Summary



Start training

High training error ?

Yes

High Bias

1. Train longer
2. Train a more complex model
3. Obtain more features
4. Decrease regularization
5. New model architecture

No

High cross-validation error ?

Yes

High Variance

1. Obtain more data
2. Decrease number of features
3. Increase regularization
4. New model architecture

Done

Unsupervised Machine Learning

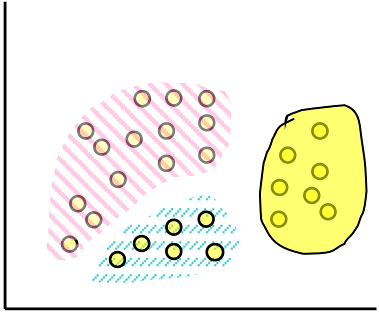
- **Unlabeled data, or data with unknown structure**
- **Explore the structure of the data to extract information**
- **Many types, we'll just discuss two.**

Clustering

- **Organize information into meaningful subgroups (clusters)**
- **Objects in cluster share certain degree of similarity (and dissimilarity to other clusters)**
- **Example: distinguish different types of customers**

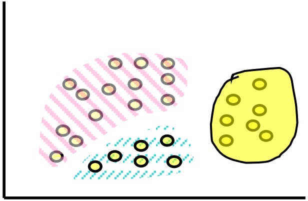
Clustering

- What we have
  - a set of un-labeled data points, each with a set of attributes
  - a similarity measure
- What we need
  - find "natural" partitioning of data, or groups of similar/close items



Clustering
(unsupervised learning)

- A set of data points, each with a set of attributes and a similarity measure, find clusters such that
  - Data points in one cluster are more similar
  - Data points in separate clusters are less similar to one another
- Key: measure of similarity between instances
  - Euclidean or Manhattan distance
  - Hamming distance
  - Other problem specific measures

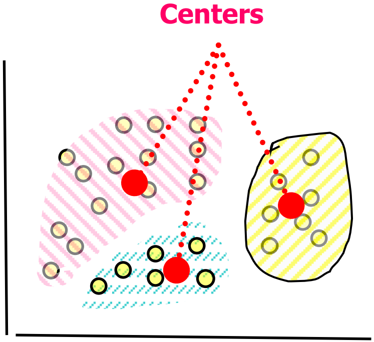


CLUSTERING

- Partitioning-based clustering
  - **K-means clustering**
  - K-medoids clustering
- Density-based clustering
  - Separate regions of dense points by sparser regions of relatively low density
  - DBSCAN

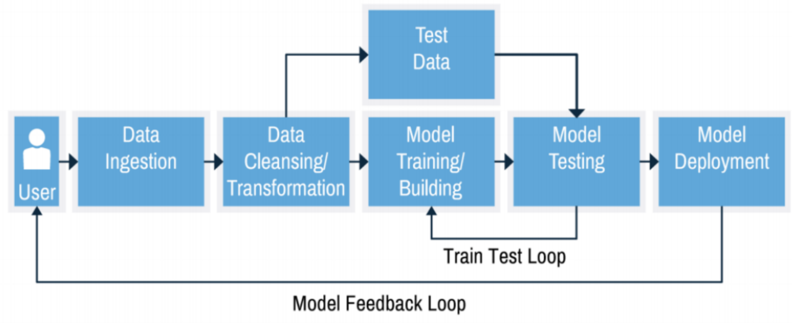
PARTITIONING-BASED CLUSTERING: K-MEANS

- Goal: minimise sum of square of distance
  - Between each point and centers of the cluster.
  - Between each pair of points in the cluster
- Algorithm:
  - Initialize K cluster centroids
    - Randomly initialize K separated points
  - Repeat until stabilization:
    - Assign each point to its closest cluster centroid
    - Update the centroid's coordinate, which are the averages of the items categorized in that cluster so far



- Visualizing K-means Algorithm: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Building machine learning



- Preprocessing: Raw data is rarely ideal for learning
  - Feature scaling: bring values in same range
  - Encoding: make categorical features numeric
  - Discretization: make numeric features categorical
  - Label imbalance correction (e.g. downsampling)
  - Feature selection: remove uninteresting/correlated features
  - Dimensionality reduction can also make data easier learn

- Learning and evaluation
  - Every algorithm has its own biases
  - No single algorithm is always best
  - *Model selection* compares and selects the best models
    - Different algorithms, different hyperparameter settings
  - Split data in training, validation, and test sets
- Prediction
  - Final optimized model can be used for prediction
  - Expected performance is performance measured on *independent* test set

WHAT COMES NEXT

- Neural network