Supervised Machine Learning

There are two main types of problems in supervised ML: regression and classification.

Regression has a continuous numeric output. (e.g. predicting a stock price)

Classification has a discrete class label output. (e.g. predicting if a student passes the final exam)

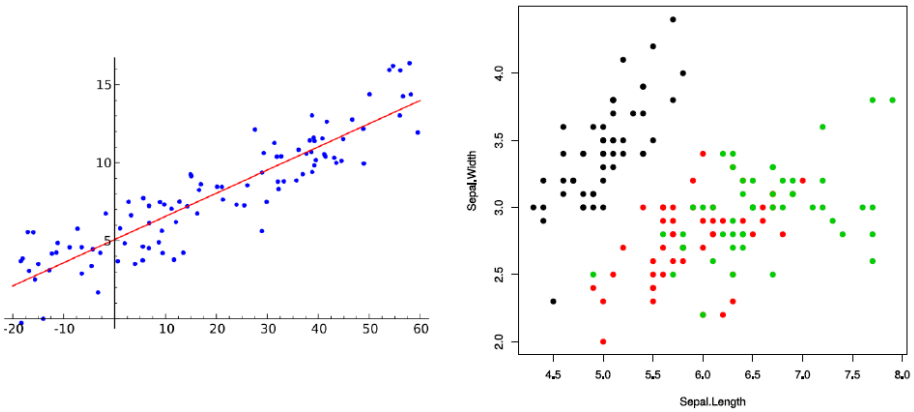

Figure 15: Regression and Classification

For regression and classification, there are different model types available.

General

Supervised machine learning uses pre-labeled data to train a model. The labels can either be measured values or labels created by human input.

Data

Supervised ML models should extrapolate relationships in the data to form a generalizing model that can predict the target variable for to the model unknown samples.

Data Splitting

To be able to know if the model performs well on previously unknown data, a held-back test set is used.

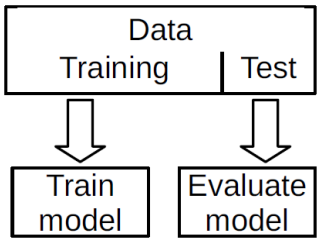

Figure 16: Train test split

The full available dataset is split into two different parts (train-test split, data partitioning etc.) as seen in fig. 16. The commonly bigger part is the training part used for training the model. The smaller test part is used to evaluate the model un unknown data and estimate the model performance.

Commonly the data is split for training/testing in partitions of 80/20 or 70/80. When a huge dataset is available the partition size for testing can be increased. In most cases, the data should be shuffled before partitioning. (except for some use cases e.g. with population, time dependence) This lowers the bias if e.g. the data recording settings were changed over the recording period.

Using the same train-test split when evaluating different models can lead to overfitting. When training multiple models one may by chance perform well on the test set. This is more likely if a high number of models is compared using the same split.

Data Amount

The better the training samples represent real data, the better the model generalizes. So if there are too less samples the resulting model will have bad generalization. This can be a possible reason for diverging train and test errors over the number of samples used as seen in fig. 17.

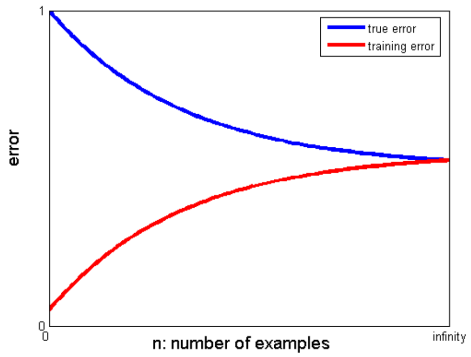

Figure 17: Am I using enough samples?

Balance For classification problems, the balance in a dataset has to be considered as well. A balanced dataset is defined as a dataset where each class includes the same number of samples. (imbalanced is when one or more classes are significantly overrepresented)

Possible countermeasures for imbalanced datasets include:

- Weight training samples according to their class frequency
- Weight error of samples according to their class frequency

- Create a balanced dataset using upsampling/downsampling (upsampling: generate artificial samples for the smaller class downsampling: pick the same number of samples from bigger class as the small class)
- Using performance metrics that can work better with imbalanced classes

K-Fold Cross Validation (CV)

K-Fold CV is a widely used resampling technique (= using samples more than once) and a more sophisticated split than the train-test split. K-Fold CV is usually used in combination with data splitting to still keep a held-back test set for the final evaluation.

The main idea is to split the data into K similarly sized parts (folds). Multiple models are trained, each using another fold as the validation (test) data (see fig. 18). The test data is used to acquire performance metrics for each model split. Those performance metrics are averaged to obtain a single performance estimate for the given configuration.

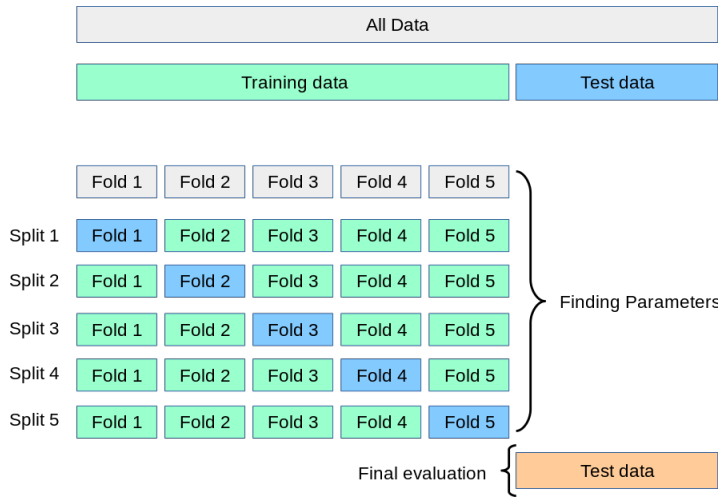

Figure 18: 5-Fold Cross Validation

This approach is commonly used to find the best hyperparameters for a given problem. The performance estimate provides a comparable metric to select the best possible configuration while still keeping a held-back test set.

Regression

Linear Regression

Linear regression has the goal of matching a linear function (e.g. $y = kx + d$) to given samples. In Fig. 19 a simple two-dimensional example for linear regression is shown. When a new value x is given the linear regression can predict the value for y. This linear function gets more complex for models with more input features.

The relationship between the dependent variable y_i and the vector of independent variables/features $(x_0 \dots x_d)$ is defined as follows:

$$y_i = w_0 + w_1 x_{i,1} + \ldots + w_d x_{i,d} + \epsilon_i$$

Linear regression has the goal to find this relationship/weights $w_0 \dots w_d$. ϵ_i is the stochastic error which is de difference between the observed and "true" value. This error is random, not systematic, cannot be learned and is also called noise.

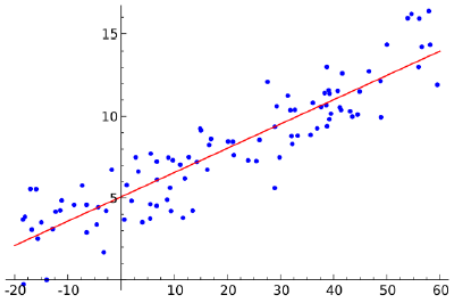

Figure 19: Linear regression

The optimal linear regression model can be obtained using either equations (closed-form) or solved iteratively (gradient-descent). The resulting model which is a plane in a high-dimensional space is commonly called a hyperplane.

Regression Error Functions

To be able to get the "best" model for a given problem the quality has to be quantifiable. Error/loss functions are available to quantify regression error:

- Absolute Error $= \sum |E_n|$, scales with amount of samples
- Mean absolute error $MAE = \sum \frac{|E_n|}{n}$
- Median absolute error $= median(|E_n|)$
- Sum of squared errors $SSE = \sum E_n^2$, higher weight for outliers, popular for linear regression
- Mean squared error $MSE = \sum (\frac{E_n}{n})^2$, normalized SSE
- Root mean squared error $RMSE = \sqrt{\sum (\frac{E_n}{n})^2}$, frequently used

Different error functions weigh outliers differently. This leads to different results when fitting a linear function to the given data. As shown in Fig. 20 the RMSE assigns outliers a higher weight compared to MAE. Despite it being really easy to spot in the given example, with more features and thus multidimensional data this is harder to spot.

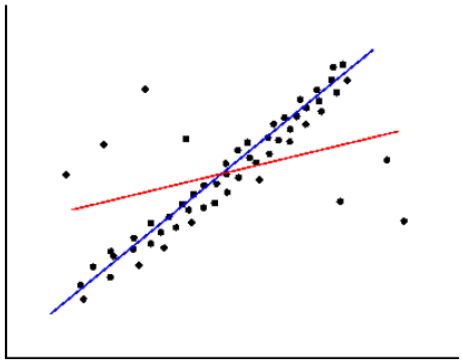

Figure 20: Linear regression with red: RMSE, blue: MAE

Variance and R2 Score

The variance as shown in Fig. 21 is the visible spread of the values. This would be the error if a "dumb" model is used which is predicting the sample's mean value independent of the input input.

The unexplained variance in contrast is the difference between the predicted and observed (true) values. This variance couldn't be explained by the model.

When combining the explained and unexplained variance the fraction of variance unexplained (FVU) can be formed which combines the error variance (MSE) and total variance ("MSE" from total variance).

$$FVU = \frac{VAR_{err}}{VAR_{tot}}$$

Using the FVU the R^2 score can be calculated. The R^2 score is a common metric to evaluate the model performance. It tells us what fraction of the variance can be explained by the regression line (model). Also, a metric gives us a comparison between the "dumbest possible" model compared to the trained model.

$$R^2 = 1 - FVU$$

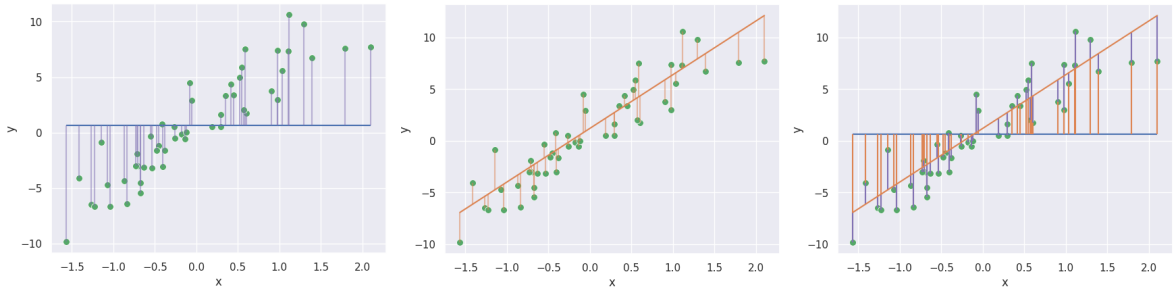

Figure 21: Variance, Unexplained Variance, Combined Unexplained and Explained Variance

Residual Analysis

The first plot in Fig. 22 visualizes the observed and predicted values of a regression model. Ideally, the predicted and observed values match up and form a line in this graph. This ideal model is usually not reachable or even desired (ref. over/under-fitting) due to the noise in the data.

After calculating the residual (observed value - predicted value) this can be plotted against the predicted value or any given feature. A symmetric distribution around the x-axis which appears random as shown in Fig. 22 is desired. If the residual plot shows some kind of curve this is a hint to non-linear, more complex relationships present in the data which are not yet represented by the model. The model type is not suitable for the given problem.

The regression error can be split into two different categories bias and variance. The bias is visible as an offset and the variance as a high spread in the residual plot.

Feature and Model Parameter Space

To be able to display features and model parameters in graphs two important multi-dimensional spaces are the feature and model parameter space.

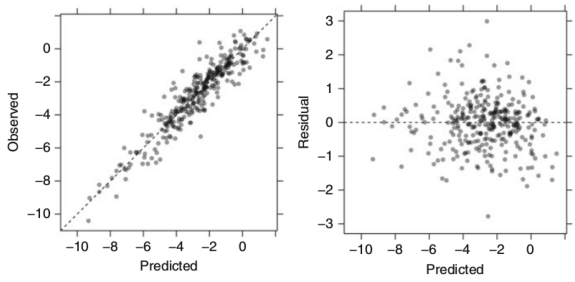

Figure 22: Residual analysis

Feature Space

The feature space uses the samples' features as coordinates to visualize the samples. Each sample is represented by a specific point in the feature space. A typical plot of a feature space is shown in Fig. 23.

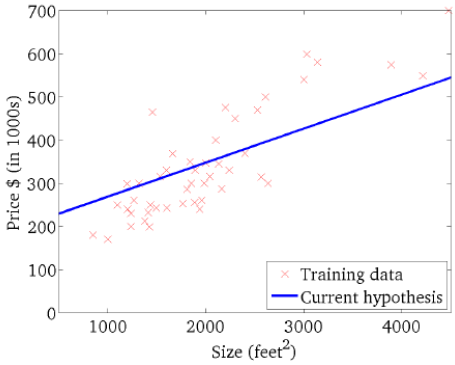

Figure 23: Feature space

Model parameter space

The model parameter space uses the models' parameters (weights) as coordinates. So each given point in the model parameter space represents a given combination of model parameters which in turn represent a single possible model. This can help to display the fitness landscape which shows possible models in combination with the error. The model representing the `x` in Fig. 24 is also the regression line/function shown in Fig. 23.

To be able to draw the fitness landscape as shown each individual model parameter combination has to be tested. This is not feasible for most use cases - if we would have the processing power to try out all possible options we would always find the best solution.

Gradient Descent

If a model cannot be solved closed-form an iterative approach can be used. To progress in an iterative way we need to know how to change the model parameters after each iteration to get a better performing model. Gradient descent uses a partial derivative of the error/loss function to get direction and steepness information. The direction and steepness define how the model parameters have to be changed to get model parameters that fit the samples slightly better. Or when using the fitness landscape or error

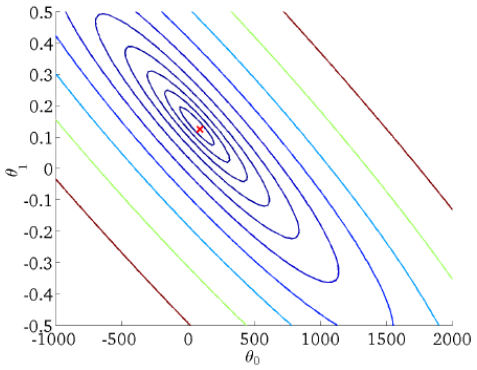

Figure 24: Parameter space/fitness landscape

contours as shown in Fig. 24 as a visualization the direction and step size define the progress to the model with the lowest error.

To start the iterative solution the model parameters need to be initialized first. The starting parameters are usually random values. Gradient descent needs stopping criteria to define how many steps/iterations are performed. Stopping criteria commonly used are convergence (reaching a specific quality) or the maximum number of steps.

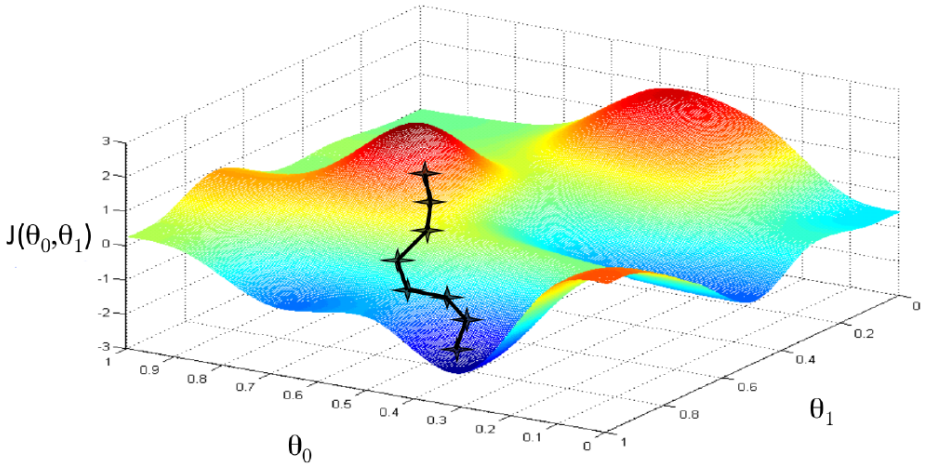

Figure 25: Gradient descent

Problems

Finding the best solution (global optima) using gradient descent is not always possible. Gradient descent can get stuck in local optima, especially in big and complex fitness landscapes.

- Global optima: is the overall best choice of model parameters resulting in the lowest error.
- Local optima: is a local best choice in comparison to its neighbors but worse than the global optimum. If the global optimum is hard to find it is commonly mistaken for a global optimum.

As shown in Fig. 25 there are multiple local optima and depending on the initial values gradient descent could end up in one of the local optima.

Learning Rate

To restrict the amount of change applied to the model parameters in each iteration a learning rate (α) is used. It is usually defined as a factor $[0, 1]$ with 0 = no change and 1 = max change.

Problems:

- Learning rate too low: only small changes, optimization can take very long, large areas of solution space are not visited at all
- Learning rate too high: big changes, can "jump" over the optimal solution and miss it, can take long to find a good solution too. Might cause parameter oscillation!
- Both might end up in local optima, but the latter one possibly "jumps out" again

The effects of a too-high or too-low learning rate are shown in Fig. 26. Learning rates are not only used for gradient descent but also in other machine learning concepts.

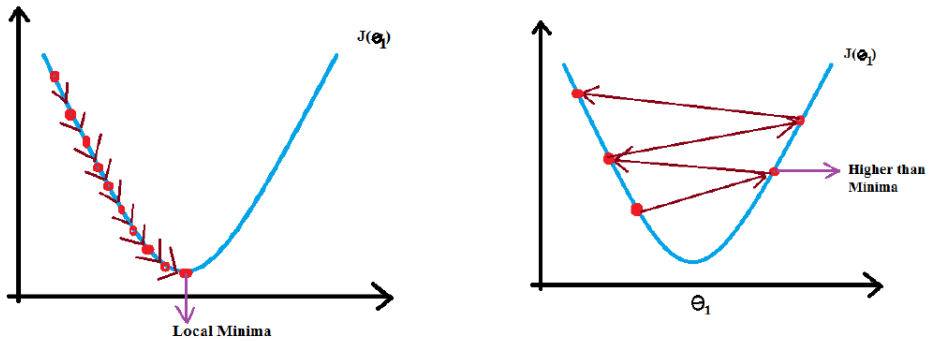

Figure 26: Learning Rate Problems wingshore.wordpress.com

Batch Updates vs. Stochastic Updates

When applying the gradient descent the model parameters can either be applied after each individual sample or after multiple/all samples (batch). As shown in 27 applying batch updates the learning progress is very directed but takes a lot of iterations. In comparison applying stochastic updates after each sample can lead to a faster, but less directed progress.

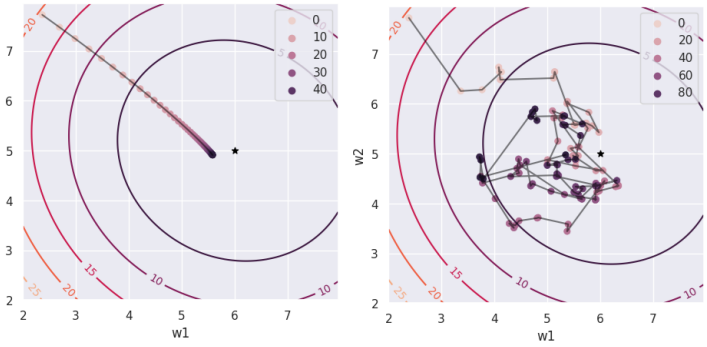

Figure 27: Batch Updates vs. Stochastic Updates

Non-linear Problems

Besides the linearity of linear regression, it can also be used to solve non-linear problems. To solve non-linear problems the training data has to be transformed into another feature space. This feature transformation has to be applied before training a model and before each application of the model.

The features can be transformed using different "basis"-functions, e.g. polynomial (see fig. 28), radial, sigmoidal, Fourier. Different basis functions can also be used simultaneously. Using the transformed data for training still keeps the easy solvability of linear regression models with the added benefit of higher-degree problems being solvable.

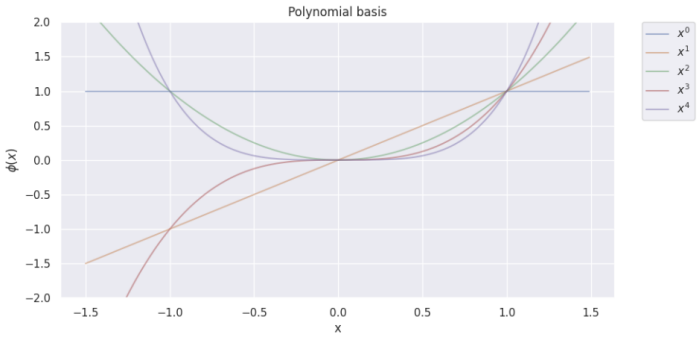

Figure 28: Polynomial basis functions

The need for different basis functions can be spotted with the residual analysis as seen in fig. 29. When there are curves or other patterns visible in the residual plots, a non-linear problem can be spotted.

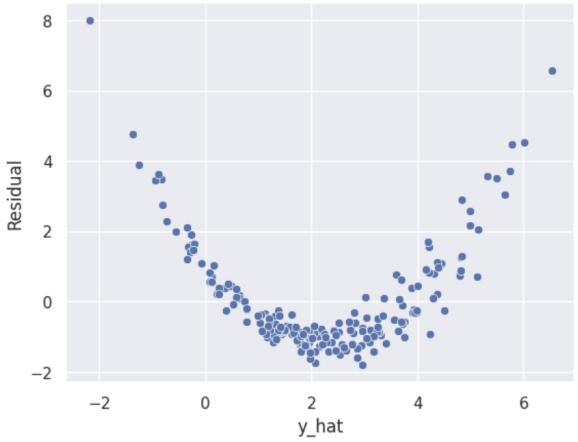

Figure 29: Residual analysis

Challenges and Problems

Overfitting/Underfitting

Over- and underfitting are phenomenons where the resulting model is not able to correctly abstract the given problem.

Underfitting as shown in Fig. 30 means the model is not capturing the underlying relations in the data. (not generalizing) This can have different reasons e.g. a model type/hyperparameters that are not able to create a model with the same complexity or the model is not trained long enough.

Overfitting by contrast fits all the models given when creating the model but is not correctly predicting unknown samples. This can also have different reasons. The most common are: a too complex model type/hyperparamter-configuration or training too long

As shown in Fig. 30 the overfitting model is predicting negative house prices for higher house sizes. This and the high predicted prices for small houses are unrealistic and don't match the trend visible in the given data.

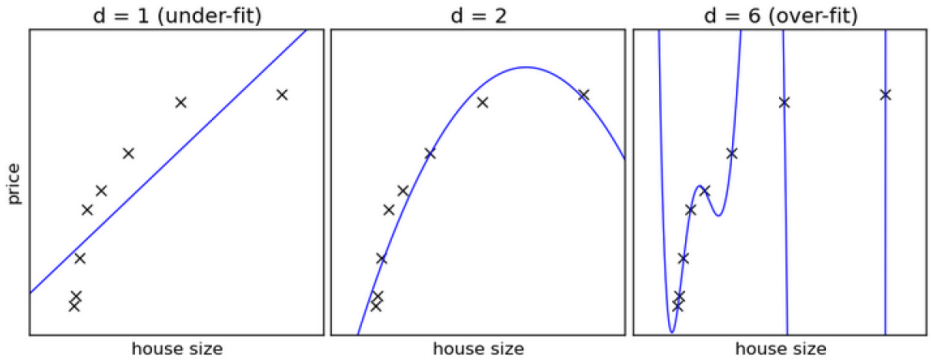

Figure 30: Overfitting and Underfitting with Polynomial Regression (coursera.org)

As also shown in Fig. 30($d = 2$) extrapolation is another problem for regression models. Outside of the features training data range the model output may not match the reality.

Bias-Variance Tradeoff/Dilemma

Bias and variance as already mentioned in section **??** describe the error of a model.

- **Bias:** represents how close the form of the model can get to the true relationship between features and outcome. *Too high bias:* the model fails to represent data often leads to underfitting
- **Variance:** represents modeling noise additional to the true relationship of data to the outcome. *Too high variance:* model fits data + noise often leads to overfitting.

A tradeoff has to be made between bias and variance, decreasing the bias leads to an increased variance and vice versa. For powerful models, it is common practice to increase the bias to decrease the variance. (restrict model power and allow certain errors in training to reduce overfitting)

Problem	Indicator	Counter measures
Underfitting	Training and test error are equally bad	Decrease bias, use more complex model
Overfitting	Training error is much better than application/test error	Decrease variance by restricting model complexity.

Regularization

In high variance scenarios, high model parameter values lead to overfitting. To prevent those high parameter values (e.g. high weight for some features) model parameters are only allowed to become large if there is a proportional reduction of the model error. Regularization adds a penalty for high parameter values. Examples of such penalized models are: ridge regression, least absolute shrinking and selection operator model (lasso) or elastic net.

Classification

Classification tries to assign a class to a given sample. (e.g. differentiate spam and non-spam messages, differentiate the contents of a given image) There are two different classes of classification problems:

- **Binary class problem**: Two different classes should be differentiated, mostly referred to as the positive (P) and negative (N) classes.
- **Multi-class problem**: Involves multiple classes (>2). This can be solved as multiple binary class problems. For each class that should be recognized a single model is created. These individual models use the samples of a single class as the positive class and the samples of all other classes as the negative samples.

Classification Metrics

The performance and error of classification models have to be calculated differently.

Confusion matrix

The confusion matrix is shown in fig. 31, shows the prediction results and the actual classes of a binary class problem. For multi-class problems a single class has to be chosen as the positive class and all remaining classes are considered the negative class.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Figure 31: Confusion Matrix, assume Yes is the positive class and No is the negative class.

The absolute values in the confusion matrix represent the four different classification results:

- True positives (TP): The prediction result and the actual class are both positive. (100)
- True negatives (TN): The prediction result and the actual class are both negative. (50)
- False negatives (FN): The prediction assigns the negative class besides the actual class being positive. (5)
- False positives (FP): The prediction assigns the positive class besides the actual class being negative. (10)

The confusion matrix can also be plotted as a graph using a heatmap and relative values. This is especially used for multi-class detection problems with an imbalanced dataset.

Multi-class As shown in fig. 33 multi-class classification problems can be displayed in a single confusion matrix. When calculating any other performance metric the values for TP, TN, FP and

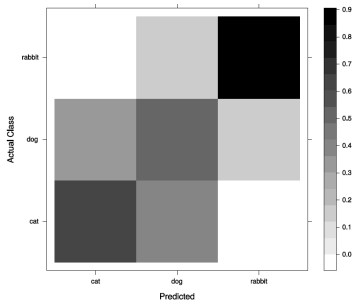

Figure 32: Relative confusion matrix plot

FN are needed. To get those values the confusion matrix has to be looked at from a single class's perspective.

In fig. 33 when determining the values for the class "dog":

- TP: all results where predicted and actual class match and are part of the specified class (3)
- TN: all results where neither the predicted nor the actual class are part of the specified class (5+0+0+11)
- FP: all results where the predicted class is the specified class but the actual class is any other class (3+2)
- FN: all results where the actual class is the specified class but the predicted class is any other class (2+1)

		Predicted		
		Cat	**Dog**	**Rabbit**
Actual class	**Cat**	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

Figure 33: Multi-class Confusion matrix

Accuracy

Single-value metrics and ratios are often more helpful than absolute numbers. The accuracy captures the percentage of correctly classified samples. It does not differentiate which errors occurred (FP or FN).

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

Other metrics

There are multiple other metrics available:

- True positive rate (TPR), sensitivity, recall $TPR = \frac{TP}{TP+FN}$ = *positive samples that are correctly classified as being positive*
- False negative rate (FNR) $FNR = 1 - TPR$ = *positive samples that are incorrectly classified as being negative*

- True negative rate (TNR), specificity $TNR = \frac{TN}{TN+FP}$ = *negative samples that are correctly classified as being negative*
- False positive rate (FPR), $FPR = 1 - TNR$ = *negative samples that are incorrectly classified as being positive*
- **TPR and TNR should always be stated together!**
- $Precision = \frac{TP}{TP+FP}$, Precision and recall state the same as TPR/TNR from different points of view.

Class probability thresholding

Most classifiers compute a probabiltiy/confidence metric in addition to the class, commonly a float value in the range $[0, 1]$. This probability can be used to derive which class a sample belongs to in a binary classification problem when assuming samples with $p(X) > 0.5$ are part of the positive class.

If another threshold is used besides 0.5 a tradeoff can be made between false positives and false negatives to fulfill certain classification needs.

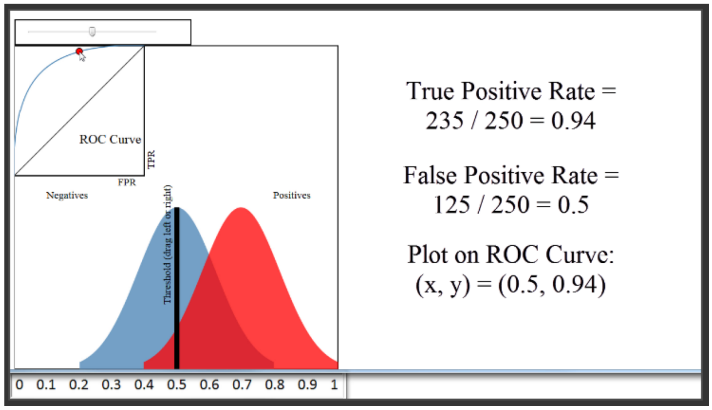

Figure 34: Probability thresholding

As shown in fig. 34 those probability thresholds and the TPR, FPR can be used to form a curve (ROC receiver operating characteristics). The area under the curve (AUC) indicates the separability of the two classes.
An AUC value of 1 indicates a clear separability of the who classes.

Uninformed classifier

When dealing with imbalanced datasets the accuracy metic can lead to wrong conclusions. If a model should be trained to classify credit card transactions (non-fraud and fraud) there will be a lot more non-fraudulent samples in the dataset. If there is 1 fraudulent in 10^6 transactions, an uninformed classifier always predicting non-fraud would lead to an accuracy of 0.999999, but result in 100% error in predicting fraudulent transactions. If there is a high class imbalance bad performance in the smaller class can be hidden in good performance in the bigger class.

This uninformed classifier always predicts the class including more samples can help as a baseline to compare models.

Logistic Regression

Logistic regression uses a linear regression model to solve a binary classification problem. The hyperplane (multi-dimensional linear function) is used to separate the classes (see fig. 35). It is used for binary

classification (yes/no, true/false) which is mostly represented by the values 0 and 1. To add a probability to its output, logistic regression results in values in the $[0, 1]$ range. (the closer to 1 the more probable the sample belongs to the "positive class") The distance to the separator determines the probability - the higher the distance, the higher the probability the classification result is correct. Logistic regression uses the sigmoid function (see fig. 36) to map the distance to the probability range ($[0, 1]$).

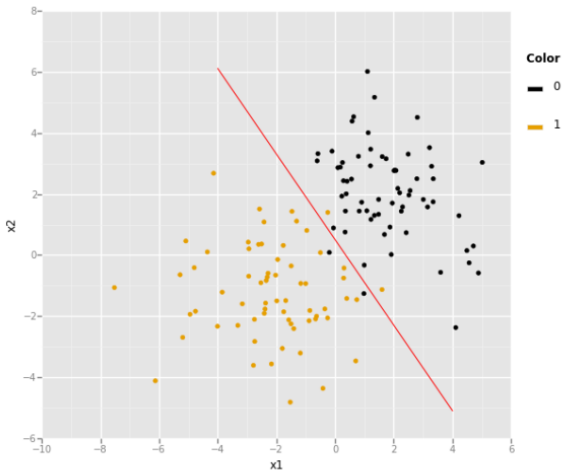

Figure 35: Logistic Regression

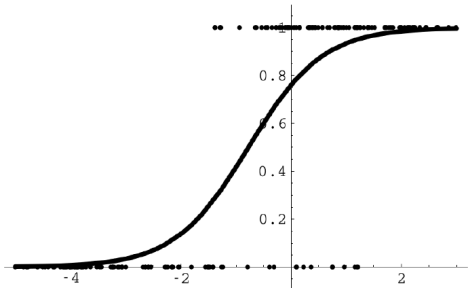

Figure 36: Sigmoid Function

As already shown in section for linear regression problems, logistic regression can also solve non-linear problems using basis-functions.

K-Nearest Neighbor (KNN)

KNN is primarily a simple classification algorithm. The simple idea is to select the class of a new/unknown sample based on the class of the K nearest neighbors. For two features this can be easily visualized. Depending on the number of neighbors (K) considered and the distance measures used the results may vary. When considering multiple neighbors usually odd numbers are chosen and the majority vote principle is applied. E.g. with $K = 3$ the class with at least 2 neighbors being part of the class is chosen as the classification result. Commonly used distance measures are Eucledian and Manhatten distance.

For the new (orange) sample in Fig. 37 with K=3 the closest neighbors are 2 green and one red. The result for the new sample would be the green class.

As shown in Fig. 38 depending on K the number of samples changes the overall robustness against

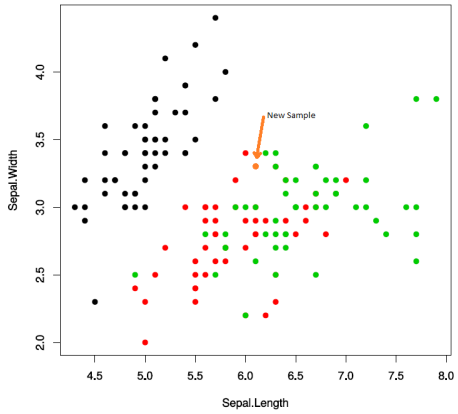

Figure 37: KNN with new Sample

outliers. There are some green outliers in the training data which lead to more green classified results compared to K=7.

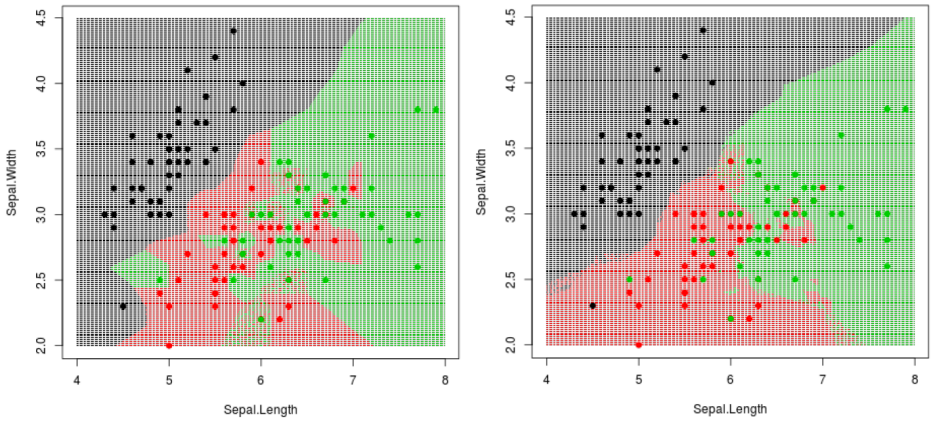

Figure 38: K=1 and K=7

Regression

KNN can also be used for Regression problems. It uses the nearest neighbors to predict a numeric value. This can be achieved using different combination approaches:

- unweighted: calculating the mean value of the neighbor's values.
- weighted: uses the distance to weigh the output value. Different distance measures can be used for weighing the results.

Limitations

KNN is so-called instance-based learning because it uses the data directly instead of deriving an explicit model. It is easy to add new samples because there is no retraining required. The storage requirements are huge because all samples for "training" are needed for each classification, which leads to slow classification speeds. This is also called lazy learning - the training is delayed until the result of the task is requested.

The distance as a measure of relationship leads to a high importance of feature scaling. Thus: *It is important to normalize the values to make sure each feature is weighted equally. Different scaling leads to different distances and therefore different results.*

Decision Trees

Combining multiple if-else (conjunction or node) statements to solve a binary classification problem is the main idea behind decision trees. Those conditions form tree-shaped structures with the leaves (terminal node) representing the prediction result.

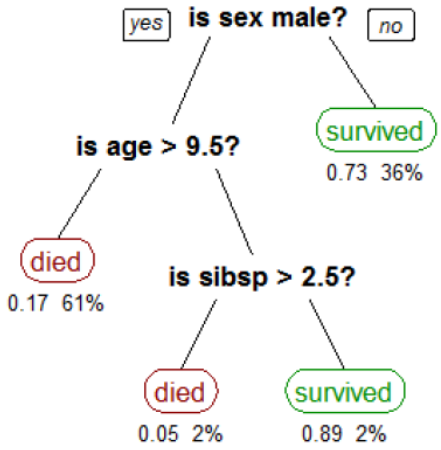

Figure 39: Decision tree example (Titanic Survivours)

Trees split the training samples into "more homogeneous" groups - this leads to less variance in the output variable. Those if-else statements are easy to understand (white-box) and can solve non-linear problems. Besides those advantages, the decision tree is prone to overfit easily, especially for deeper trees. Trees are also called weak learners, due to the low learning ability of simple trees resulting in a model performing only slightly better than random guessing.

Decision trees can also be used for regression.

Approaches

To train generalizing trees a few different approaches can be used. The easiest approach is to train a tree and then reduce its complexity by pruning/cutting it using regularization.

More complex options use multiple trees combined using ensembling techniques to form a stronger model. When combing the different trees the majority vote principle is used for classification problems and the mean value for regression problems. Bagging and boosting are the two most important ensembling techniques explained in the following sections. Those approaches can also be applied for other weak learner model types.

Bagging Bagging splits the training data into N equal-size partitions and trains an individual model for each partition. The resulting trees look different from each other. The combination of those different models leads to decreased variance but increased performance needs.

Feature Bagging

Another form of bagging is feature bagging used by the random forest model type. The individual models are trained based on a subset of the features. A random selection of features is chosen and for each random subset, a tree is trained. This results in independent trees focusing on different features.

Boosting

Boosting assigns the same weight to each training sample at the beginning of the training. After a model is trained the weights are altered:

- Samples incorrectly classified - increase weight
- Samples correctly classified - decrease weight

The resulting models are in turn weighted by their individual performance and combined to form a strong model. This approach assigns a higher weight to difficult-to-predict samples, which leads to each model focusing on different parts of the data.

Boosting leads to an increased prediction performance while reducing the variance. However due to the greediness of the weighing approach, there is still the possibility of overfitting. This approach is used by different model types e.g. AdaBoost.

Choosing the Right Model (Type)

When comparing multiple different model types or models commonly K-fold CV is used. As already described in section CV gives a realistic performance estimate without using the held-back test set. With increasing model complexity the training performance increases (see fig. 40). Because of the increased variance, the chance of overfitting increases with increasing model complexity and the test performance decreases.

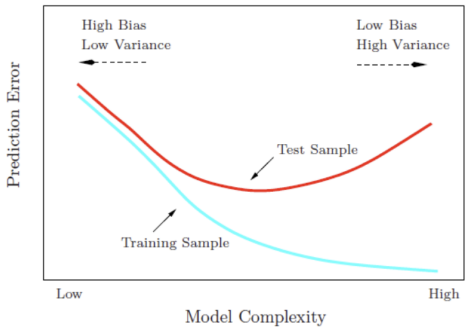

Figure 40: Model complexity graph

To evaluate the effect of an increased model complexity (e.g. higher polynomials used) for each complexity (hyperparameter) configuration a series of models is trained using K-Fold CV. For each configuration, the performance estimate is calculated (e.g. R2-Score, TPR). As shown in fig 41 the best-performing model may not be the least complex model.

To increase the trustworthiness of the resulting model a model with acceptable performance and decreased complexity is preferred. To be able to systematically determine the best suitable model for reaching those goals the “1-standard error (SE)” rule is introduced. The standard error is defined as follows:

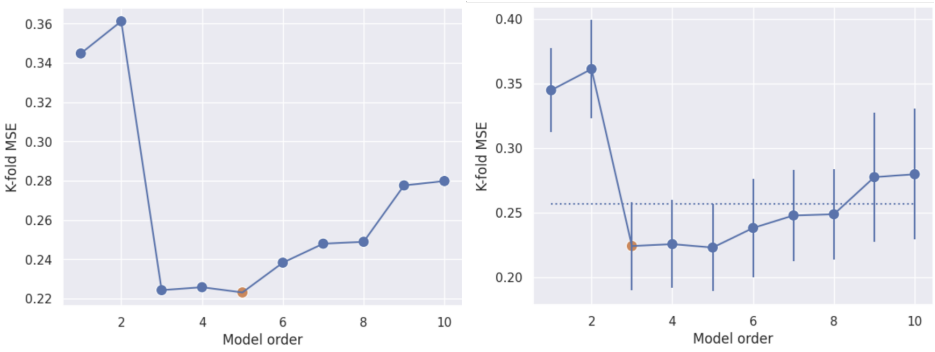

Figure 41: Model selection using K-Fold CV

$$\frac{\sigma_{\text{Error}}}{\sqrt{K-1}}$$

For each model candidate, the standard deviation of the Error (or other performance metric) is calculated over the K folds. The SE from the best-performing model is added (in other cases subtracted - if higher is better) to the best performance metric. The model with the lowest complexity, still better than the calculated value is selected.

Model Types

This selection approach can also be used when comparing different model types. With comparable performance, the least complex model type is preferred.

Model types are commonly classified as being white-box or black-box. White-box models are easy to understand for humans. Black-box models in comparison are hard to understand, and this may lead to unforseable prediction results.

Advantages of low complexity models

- better understandable for humans
- usually faster prediction speeds