Lab 2

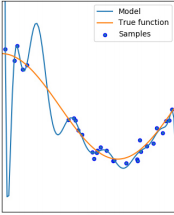
# Task 1

1. **When can you use linear regression?**  
   You can use linear regression to make a forecast of upcoming data points or get a mean value of the already existing data. To get a good result, the data must be correlated in a linear pattern. Linear regression assumes the following:

* The relationship between X and Y is linear
* Y is distributed normally at each value of X
* The variance of Y at every value of X is the same (homogeneity of variances)
* The observations are independent

1. **How can you generalize linear regression models to account for more complex relationships among the data?**You can draw from an array of different distributions to find the “best” fit model. (GLM – Generalized Linear Models: Poisson regression, normal regression and binomial regression)
2. **What are the basis functions?**The basis functions are the functions that are used to project our one-dimensional values into higher dimensions to find more complex relationships between x1 and x2.
3. **How many basis functions can you use in the same regression model?**We did not find a good answer for this question, but it should be as many as the number of components.
4. **Can overfitting be a problem? And if so, what can you do about it?**

Yes, overfitting can create misleading data (even outside the min/max boundaries). An example in the beginning of the graph in the figure below. If you test your model with different datasets, use cross-validation or/and stop early in the fitting, you can prevent it.



# Task 2

1. **Why choosing a good value for k is important in KNN?**Because a “wrong” k can produce a result which is not desired. For example, a high k is not good if you desire a high-speed algorithm with high variance. A too low number of k is very sensitive to noise.
2. **How can you decide a good value for k?**Look at the dataset and use the elbow method until it converges to your desired result.
3. **Can you use KNN to classify non-linearly separable data?**Yes
4. **Is KNN sensible to the number of features in the dataset?**Yes, it does not scale well.  
    Many features = probably large number k = probably large dataset = slow
5. **Can you use KNN for a regression problem?**Yes
6. **What are the Pros and Cons of KNN?** *Pros*   
   - Faster training phase compared to other classification algorithms.   
   - KNN can be useful in case of nonlinear data.   
   - Can be used with regression problems. (The predicted value of the new data point is computed by calculating the average of the k closest neighbors values.)  
    *Cons*   
   - The testing phase of KNN is slower and costlier in terms of time and memory.  
   - It requires large memory for storing the entire training dataset.  
   - KNN requires the rescaling of the data set in particular if the Euclidean distance measure is used. The Euclidean distance is sensitive to magnitudes; the features with high magnitudes will weigh more than the features with low magnitudes.   
   - KNN is not suitable for large dimensional data sets.

# Task 3

1. **What is the basic idea/intuition of SVM?**A supervised algorithm for classification or regression. Locate the support vectors, i.e. the outlying datapoints for each cluster and then draw a line which maximizes the margin to the vectors.
2. **What can you do if the dataset is not linearly separable?**You can use the kernel method to find a nonlinear decision boundary.
3. **Explain the concept of Soften Margins.**You allow some points to be positioned inside the margin to reduce the impact of noise. (The margin line will be drawn inside the cluster).
4. **What are the pros and cons of SVM?***Pros*

* Their dependence on relatively few support vectors means that they are very compact models and take up very little memory.
* Once the model is trained, the prediction phase is very fast.
* Because they are affected only by points near the margin, they work well with high dimensional data; even data with more dimensions than samples - which is a challenge for other algorithms.
* Their integration with kernel methods makes them very versatile, able to adapt to many kinds of data.

*Cons*

* The scaling with the number of samples N is at worst O(N3), or O(N2) for efficient implementations. For large numbers of training samples this computational cost can be prohibitive.
* The results are strongly dependent on a suitable choice for the softening parameter C. This must be carefully chosen via cross-validation, which can be expensive as datasets grow.
* The results do not have a direct probabilistic interpretation. Although this can be estimated via an internal cross-validation, but this extra estimation is costly.