AMRC Exam Questions Answered

Florian D.

September 2023

1 Regression / Linear Methods

Question 1: Explain Ridge and Lasso Regression. What are the differences? How do they differ from OLS? Show all formulas, explain the λ parameter and also explain why Lasso sets coefficients exactly to zero and Ridge does not.

The ordinary least squares regression (OLS) is BLUE (best linear unbiased estimator) if we assume that the error terms ε are normally distributed with a mean of zero and equal variance σ^2. It has the following form:

$$y = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$$

The residual sum of squares is the minimization criterion:

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$

However it is not always the best idea to keep the model unbiased. We could reduce the sampling variance by increasing the bias. Ridge and Lasso penalize the parameter size.

$$\hat{\beta}_{Ridge} = \arg\max_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

We can see in the formula above that the size of the parameters β_j is penalized by taking their square. This leads to a smoother model and prevents overfitting. How much the model is penalized for large betas depends on the regularization parameter λ.

$$\hat{\beta}_{Lasso} = \arg\max_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

For Lasso we use the L1 norm instead of the L2 norm. This means that the acceptable region is (visually speaking) diamond-shaped, rather than a circle. Thus we end up at zero values much easier.

Question 2: Explain PCR and PLS.

PCR stands for Principal Component Regression. The idea is to reduce the dimensionality of the inputs by transforming the data into a new set of uncorrelated variables, which are ranked according to their variance size. These are the principal components. Here it is important to center and scale the data first. PCR is not scale-invariant.

$$Z = XV$$

Z are the principal components, which are the result of multiplying the inputs X by an eigenvector matrix V. The shape of X is (n x p), the shape of V is (p x p), which results in Z having the shape (n x p). It is important to mention that PCR does not take the latent variables y_i into account.

$$\begin{aligned} y &= X\beta + \varepsilon \\ &= XVV^\top \beta + \varepsilon \\ &= Z\theta + \tilde{\varepsilon} \end{aligned}$$

If we only use $q < p$ principal coponents:

$$y = Z_{1:q}\theta_{1:q} + \tilde{\varepsilon}$$

The estimator θ can be back-transformed to be interpreted in terms of the original variables:

$$\tilde{\hat{\beta}} = V_{1:q}\hat{\theta}_{1:q}$$

It is important to mention that $\tilde{\hat{\beta}}$ is no longer the original estimator unless the first q principal components explain all information of X.

PLS on the other hand stands for partial least squares. Here we also construct a set of linear combinations of the input for regression but we also use y in addition to X for this.

$$y_i = t_i^\top \gamma + \tilde{\varepsilon}_i$$

The formula above shows the form on a PLS regression model. Here instead of using the inputs X, we have a score matrix T, which has the shape (n x q), where q is less or equal to p. We don't know the content of T but we obtain each column of it sequentially by using the PLS criterion:

$$w_k = \arg\max_w \mathrm{Cov}(y, Xw)$$

We obtain the columns of the matrix W by applying this criterion. Then the score matrix T is obtained by:

$$T = XW$$

It is important to mention that we have the following constraints:

$$||w_k|| = 1 \text{ to make sure these are just directions}$$
$$Cov(X_{w_k}, X_{w_j}) = 0$$

to make sure that all directions are orthogonal. Our goal is to find the direction in which we can project X so that the covariance with the output y is maximized.

Question 3: Explain for multiple regression models the OLS solution, how to arrive at it and what to do in the case of near singularity of $X^\top X$

For the least squares regression model, we have the following solution for the estimator $\hat{\beta}$:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

For the prediction $\hat{y}$ we can write:

$$\begin{aligned} \hat{y} &= X\hat{\beta} \\ &= X(X^\top X)^{-1} X^\top y \end{aligned}$$

So $X(X^\top X)^{-1}X^\top$ is called the hat matrix because it puts a hat on the y.

A problem that we have is that $X^\top X$ can become nearly singular when the input variables are highly correlated. This would mean the matrix would almost be not of full rank. And almost not invertible. Practically speaking, this makes the model extremely sensitive to changes in the input. The solution would be to use a regularized method like Ridge or Lasso.

2 Classification Methods

Question 4: Explain LDA, QDA and RDA. What is the idea of LDA? What is the Bayesian Theorem, what are its assumptions? Specify the φ formula. Also provide the formula for LDA and QDA. What are the components of LDA and how do you estimate them? Why can you estimate them?

LDA stands for Linear Discriminant Analysis. The goal here is to find a linear combination of the input which best separates the two classes. The following formula gives us the conditional probability that given an observation x, the random variable G is k:

$$P(G = k|x) = \frac{h_k(x)\pi_k}{\sum_{l=1}^{K} h_l(x)\pi_l}$$

In this formula above, $h_k(x)$ is the density function of x in class G = k. The probability of any observation belonging to class k is π_k.

Oftentimes it is assumed that $h_k(x)$ is the density of a multivariate normal distribution:

$$\varphi_k(x) = \frac{1}{\sqrt{(2\pi)^p|\Sigma_k|}} e^{-\frac{(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)}{2}}$$

LDA arises when we assume that all classess have a common covariance Σ_k.
Then the linear discriminant function is:

$$\delta_k(x) = x^\top \Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^\top \Sigma^{-1}\mu_k + \log\pi_k$$

The decision rule is then:

$$G(x) = \arg\max_k \delta_k(x)$$

In QDA, which is quadratic discriminant analysis, we do not assume equal covariance matrices. Then the discriminant function becomes:

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k) + \log\pi_k$$

Finallyu there is the special case of the regularized discriminant analysis (RDA). Here we have a parameter α, which provides are compromise between LDA and QDA.

$$\hat{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma}$$

If α is 0, we have LDA, if it is 1 we have QDA. Everything in between is a compromise.

Question 5: Explain logistic regression. Provide all formulas, including the log-likelihood function.

Logistic regression is a classification algorithm. Its output also contains an inference statistic, which provides information about which variables are well suitable for separating the groups. The model looks like this (for two classes):

$$\log \frac{P(G=1|x)}{P(G=2|x)} = \beta_0 + \beta_1^\top x = z$$

We can rewrite it like this:

$$\log \frac{P_1}{P_2} = z$$

$$\frac{P_1}{P_2} = \frac{P_1}{1-P_1} = e^z$$

$$P_1 = e^z - P_1 e^z \rightarrow P_1 + P_1 e^z = P_1(1+e^z) = e^z$$

$$P_1 = \frac{e^z}{1+e^z}$$

For P_2 we can say:

$$P_2 = 1 - P_1 = 1 - \frac{e^z}{1+e^z} = \frac{1}{1+e^z}$$

The log-likelihood function is:

$$l(\beta) = \sum_{i=1}^{n} \log\, p_{g_i}(x_i, \beta)$$

3 Basis Expansions / Splines

Question 6: Explain what splines are in general. What is the criterion to optimize and what are the solutions (provide formulas). How do you determine lambda and the degrees of freedom. What do degrees of freedom mean in this context? Why do we care about it? Talk about knots, penalization of curvature and how to minimize it.

In general splines are piecewise polynomials. A spline of order M with k knots ξ_i where i goes from 1 to k is a piecewise polynomial of order M, which has continuous derivatives up to order M-2 (needed for curvature later on). The general form of the basis functions is:

$$h_j(x) = x^{j-1},\ j = 1,...,M$$

$$h_{M+l}(x) = (x - \xi_l)_+^{M-1},\ l = 1,...,\text{k}$$

Here it is important to notice that M=1 means constant, M=2 means linear, M=3 means quadratic and M=4 means cubic splines.

For splines, we have to choose the order M of the splines, the number of knots $|\xi|$ and the placement of the knots.

The polynomials we use for the regions tend to be erratic near the lower and upper data range. This can result in poor approximations. In natural cubic splines we thus also add an extra condition that the function has to be linear beyound the boundary knots. By doing this we also win back 4

degrees of freedom (since cubic and now linear in both boundary regions).

With smoothing splines we can avoid to choose the number and placement of knots. We do this by controlling the complexity of the fit through regularization. We minimize the following residual sum of squares:

$$RSS(f, \lambda) = \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 + \lambda \int f''(t)^2 dt$$

The first term measures the closeness to the data and the second term penalizes curvature of the function. We basically sum up the squared curvatures of the function at all points. By setting λ to 0, f can be any function that interpolates the data. By setting λ to ∞ we get the simple least square fit, where no second derivative can be tolerated.

The solution to this minimization problem is a unique minimizer, which is a natural cubic spline with knots at the values x_i with i going from 1 to n. It seems like this solution is over-parameterized, since n knots correspond to n degrees of freedom. However, these degrees of freedom are reduced by the penalty term, since the spline coefficients are shrunk towards the linear fit.

Since the solution is a natural cubic spline, we can write it as:

$$f(x) = \sum_{j=1}^{n} N_j(x)\theta_j$$

where N_j is the j-th spline basis function.

So the criterion is reduced to:

$$RSS(\theta, \lambda) = (y - N\theta)^\top (y - N\theta) + \lambda \theta^\top \Omega_N \theta$$

with $\{N\}_{ij} = N_j(x_i)$ and $\{\Omega_N\}_{jk} = \int N_j''(t) N_k''(t) dt$

The solution is:

$$\hat{\theta} = (N^\top N + \lambda \Omega_N)^{-1} N^\top y$$

This is a generalized form of the ridge regression. Choosing λ can be done using cross-validation.

4 Generalized Additive Models (GAMs)

Question 7: What are GAMs? What does the model look like? What is the minimization criterion? How do we find a solution?

Instead of having a weighted sum (linear combination) of regressor variables in GAMs we have a weighted sum (linear combination) of transformed regressor variables. The general form of a GAM looks like this:

$$g[\mu(x)] = \alpha + f_1(x_1) + ... + f_p(x_p)$$

In general, we have a conditional expectation of the target value y, given the corresponding data. This expectation is $\mu(x)$. We take this expectation and feed it into a so-called link function g on the left side. Examples are the identity link, the logit function or just the log.

We can also write the model like this:

$$y_i = \alpha + \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i$$

To find the functions f_j, we minimize the penalized residual sum of squares criterion:

$$\text{PRSS}(\alpha, f_1, ..., f_p) = \sum_{i=1}^{n} \{y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij})\}^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt$$

Here t_j is the domain along which the smoothing is applied. Independent of the choice of the smoothing parameters λ_j an additive model with cubic splines minimizes the PRSS. The solution can be found using the backfitting algorithm.

The general idea is to start with an initial guess for each component (smoothed function) of the model. Then we pick one of the components and keep all others fixed. We fit this component to the residuals from the other components. We update our guess based on the information that the other components have not explained yet. We repeat this process for all components. We do all of this until the estimates stop changing (or change very little). Once, converged, our GAM is estimated.

5 Tree-Based Methods

Question 8: What are regression and classification trees? Explain the general idea and provide the criterion to be minimized for both cases. Also explain the different measures of node impurity. Furthermore, explain how trees are split (splitting criteria and formula), how to grow a tree, when to stop and how to prevent overfitting?

In general, tree-based methods partition the feature space into rectangular regions. The goal is for them to be as homogeneous as possible. In the context of the lecture, we limited the decision rules and thus the splits to binary splits, meaning we always only have two branches. The split is based on a split variable (which feature to look at) and a split point (what is the cutoff value for the split).

Our tree partitions our feature space into M Regions $R_1, ..., R_M$ and we model the response for each region as a constant c_m (meaning every input combination that lies in this region will produce the same result). This can be written as:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

Here I is the indicator function, which returns 1 if x is part of R_m and 0 otherwise. If our minimization criterion is the sum of squares, the best choice of $\hat{c}_m$ is just the average of the y_i values within each region.

$$\hat{c}_m = \text{ave}(y_i | x_i \in R_m)$$

Finding the best partition for this problem is computationally infeasible, thus we approximate a solution. For this we consider a split variable x_j and a split point s. A split based on these is defined by two half planes:

$$R_1(j, s) = \{x | x_j \leq s\};\ R_2(j, s) = \{x | x_j > s\}$$

We then define the following minimization criterion and search for the splitting variable x_j and the split point s that solve it:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

Put into words, we are basically looking for a splitting variable and a split point so that the sum of the errors of the two resulting regions are minimized. The inner part of this problem is easily solved because:

$$\hat{c}_1 = ave(y_i|x_i \in R_1(j,s) \text{ and } \hat{c}_2 = ave(y_i|x_i \in R_2(j,s)$$

Because this is so trivial, we can scan through all the inputs to determine the best pair (j, s).

To avoid overfitting here, we use a tuning parameter, which tries to regulate the model's complexity. First we grow a large tree T_0. We stop growing this tree when a minimum node size is reached. Then we prune this tree using cost complexity pruning. By doing this, we get a sub-tree T.

For the cost complexity criterion we index the leaf nodes by m representing the Region R_m. $|T|$ denotes the number of leaf nodes in our sub-tree T and

$$\hat{c}_m = \frac{1}{n_m}\sum_{x_i\in R_m} y_i$$

(this is just the average of the region since n_m is the number of observations in R_m)

as well as

$$Q_m(T) = \frac{1}{n_m}\sum_{x_i\in R_m}(y_i-\hat{c}_m)^2$$

(which just denotes the impurity of the node).

With this, we can now define the cost complexity criterion as

$$c_\alpha(T) = \sum_{m=1}^{|T|} n_m Q_m(T) + \alpha|T|$$

We want to minimize this criterion, which means that we want to reduce the impurity as well as the size of the tree as much as possible. To find the optimal sub-tree T_α for a given α, we eliminate the internal node which yields the smallest increase of $\sum_m n_m Q_m(T)$ per node. We do this until no node is left. This sequence of sub-trees must include T_α.

For classification, we partition the x-variables into K classes based on their y-values, which range from 1 to K. In a node m representing a region R_m with n_m observations, we define the proportion of the class k in this node like this:

$$\hat{p}_{mk} = \frac{1}{n_m}\sum_{x_i\in R_m} I(y_i = k)$$

Every node m is assigned a class like this:

$$k(m) = \arg\max_k \hat{p}_{mk}$$

In other words, we assign the most common class in a node to this node.

Three example measures that can be used for the impurity measure $Q_m(T)$ are:

- Misclassification error: $\frac{1}{n_m} \sum_{x_i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_m k$
- Gini Index: $\Sigma_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
- Cross-entropy: $\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

Question 9: What are Random Forests? What is the basic idea behind them? Explain the algorithm step-by-step.

Decision trees are very sensitive to small changes in the data. A small change in the data can lead to a completely different tree. What random forests do is they train many trees in an ensemble. These trees are generated randomly by using only bootstrapped samples and by only using a random selection of variables to be available for splitting at each knot.

The algorithm works as follows:

- Step 1: Bootstrap Sampling (random sample with replacement of size n)
- Step 2: Grow tree based on this first sample
  - At each knot choose a random selection of p variables to be available for splitting
  - For classification choose $\sqrt{p}$ variables and for regression choose $\frac{p}{3}$ variables
- Step 3: Use the OOB data (not used for training, i.e. not part of the bootstrapped sample) to compute feature importance
  - Compute the impurity of the whole tree T_b as $\pi_b = \sum_{m=1}^{|T_b|} Q_m(T_b)$ (sum of impurity of all nodes)
  - Shuffle each variable x_j randomly and compute how impure the tree is after this as π_{b_j}
  - The feature importance is $\delta_{b_j} = \pi_b - \pi_{b_j}$
- Step 4: Repeat steps 1-3 for B number of trees and combute for each the feature importances δ_{b_j} for all b and all j
- Step 5: Compute the overall feature importance score for the j-th variable as follows:

$$\hat{\theta}_j = \frac{1}{B} \sum_{b=1}^{B} \delta_{b_j}$$

To make decisions with the random forest we just take the average of all tree responses for regression and the majority for classification.

6 Support Vector Machines (SVMs)

Question 10: What are SVMs? How do they work? What is the criterion for the linearly separable case and for the non-separable case? Explain the optimization formula and how you get there for both. Explain L_p and L_d with constraints.

Support Vector Machines aim to separate groups of data using hyperplanes. In the lectore we focused on the two class case (K=2). The classification returns a sign (positive or negative) for the group membership.

In general a hyperplane $\mathcal{L}$ is defined by:

$$f(x) = \beta_0 + \beta^\top x = 0$$

As a distance measure for any point x to $\mathcal{L}$ we can use the following:

$$\beta^{*\top}(x - x_0) = \frac{1}{||\beta||}(\beta^\top x + \beta_0) = \frac{1}{||f'(x)||} f(x)$$

Here x_0 is a point on $\mathcal{L}$ and x is any point. By subtracting them, we get a distance vector and then we project it onto the direction of β^*.

If all points are completely separable by a hyperplane we can always find a function $f : \mathbb{R}^p \to \mathbb{R}$ so that $g_i f(x_i) > 0$ for all i. In other words all instances will be classified correctly. The product of the class $g_i \in \{-1, 1\}$ and the function result will always be positive.

We define the distance M as the minimal distance of a point to the hyperplane taken over all observations. So the distance to the closest point from the hyperplane.

$$M = \min_{i=1,\ldots,n} g_i f(x_i)$$

Our goal is to find a hyperplane that maximizes the margin, which is $2M$ wide (one M on each side). So we end up with the following maximization problem:

$$\max_{\beta,\beta_0,||\beta||=1} M$$

so that

$$g_i(x_i^\top \beta + \beta_0) \geq M$$

Put into words, we are just aiming so maximize the margin under the condition that all points are on or beyond the margin. Nothing is within it. We can re-write this further as a minimization problem to remove the constraint that β has to be normed.

$$\min_{\beta,\beta_0} ||\beta||$$

so that

$$g_i(x_i^\top \beta + \beta_0) \geq 1$$

In our distance measure before we had $||\beta||$ in the denominator. It has an inversely proportional relationship to the margin. The smaller β, the larger the margin. Keeping $g_i(x_i^\top \beta + \beta_0)$ greater or equal to one ensures that the instances are all on the correct side of the hyperplane.

This minimization problem is a convex optimisation problem. This means that we are looking for the minimum of a function that looks like a bowl. We cannot get stuck in a local minimum. We can use the Lagrange method to solve it. For this we have to minimize the Lagrange Primal Function:

$$L_p = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{n} \alpha_i [g_i(x_i^\top \beta + \beta_0) - 1]$$

Here we minimize $\frac{1}{2}||\beta||^2$ due to mathematical convenience. Derivation is easier like this. The second part is the inequality condition which we bring into the objective function using the Lagrange multipliers α_i.

Setting the derivatives of this function with respect to β and β_0 to zero, results in the Lagrange dual function:

$$L_d = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j x_i^\top x_j$$

This now results in the final maximization problem:

$$\max_{\alpha_i} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j x_i^\top x_j \right]$$

so that $\alpha_i \geq 0$ for all i.

To be optimal, we also need to satisfy the Karush-Kuhn-Tucker conditions:

$$\sum_{i=1}^{n} \alpha_i g_i x_i = \beta$$

$$\sum_{i=1}^{n} \alpha_i g_i = 0$$

$$\alpha_i [g_i (x_i^\top \beta + \beta_0) - 1] = 0 \quad \text{for all i}$$

In the non-separable case, where the data points of the two classes overlap, we introduce a so-called slack variable $\xi_i \geq 0$ for each side condition. Its value is the violation of the corresponding side condition. When it is positive it means that the point x is on the wrong side of the margin by the amount $M\xi_i$. For correctly classified points x_j the slack variables ξ_j are zero.

We can formulate the following minimization problem:

$$\min_{\beta, \beta_0} \left[\frac{1}{2} ||\beta||^2 + C \sum_{i=1}^{n} \xi_i \right]$$

so that

$$\begin{cases} g_i (x_i^T \beta + \beta_0) \geq 1 - \xi_i \quad \text{for all i} \\ \xi_i \geq 0 \quad \text{for all i} \end{cases} \tag{1}$$

We again maximize the margin as before by minimizing $\frac{1}{2}||\beta||^2$. Then we also penalize misclassified observations by adding $C \sum_{i=1}^{n} \xi_i$ to the problem. Here C is the cost parameter. For this problem we again have a Lagrange primal function:

$$L_p = \frac{1}{2} ||\beta||^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i [g_i (x_i^T \beta + \beta_0) - (1 - \xi_i)] - \sum_{i=1}^{n} \lambda_i \xi_i$$

Again we put our minimization problem first and then we add the side conditions with Lagrange multipliers α_i and λ_i. After taking the derivatives with respect to β, β_0 and ξ_i, we get the following Lagrange dual function:

$$L_d = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j x_i^\top x_j$$

This leads to the maximization problem:

$$\max_{\alpha_i} \left[\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j x_i^\top x_j \right]$$

so that $0 \leq \alpha_i \leq C$ for all i.

To be optimal, we also need to satisfy the Karush-Kuhn-Tucker conditions:

$$\sum_{i=1}^{n} \alpha_i g_i x_i = \beta$$

$$\sum_{i=1}^{n} \alpha_i g_i = 0$$

$$C = \alpha_i + \lambda_i$$

$$\lambda_i \xi_i = 0$$

$$\alpha_i [g_i (x_i^\top \beta + \beta_0) - (1 - \xi_i)] = 0$$

$$g_i (x_i^\top \beta + \beta_0) - (1 - \xi_i) \geq 0$$

for all i.

Question 11: Explain the kernel trick and kernel functions.

We can use transformations of our original data and then fit a linear model on these transformed inputs. Let $h_m(x)$ be the m^{th} transformation of x for M transformations. A linear basis expansion of x is defined as:

$$H(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$$

Instead of using the function f as before, we re-define it as follows:

$$f(x) = h(x)^T \beta + \beta_0$$

This is still a linear model in the variables $h(x)$ but these variables themselves can be non-linear transformations. The Lagrangian dual function can be written as:

$$L_d = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j g_i g_j \langle h(x_i), h(x_j) \rangle$$

Here $\langle h(x_i), h(x_j) \rangle$ denotes the inner product. It is important to notice that the transformations $h(x)$ only occur as an inner product. This means that there is no need to actually evaluate them. It is enough if we know a symmetric positive (semi-) definite function K, the so-called kernel function:

$$K(u, v) = \langle h(u), h(v) \rangle$$

The kernel function K computes the inner product in the transformed feature space without us having to first compute the transformations. The following kernel functions are important:

Linear kernel:

$$K(u, v) = \langle u, v \rangle = u^\top v$$

Polynomial kernel (degree d):

$$K(u,v) = (c_0 + \gamma\langle u, v\rangle)^d \quad \text{for constant } c_0 \text{ and with } \gamma > 0$$

Radial Basis kernel (RBF):

$$K(u,v) = e^{-\gamma||u-v||^2} \quad \text{with } \gamma > 0$$

Sigmoid kernel:

$$K(u,v) = \tanh(\gamma\langle u, v\rangle + c_0) \quad \text{for constant } c_0 \text{ and with } \gamma > 0$$

This is what the kernel trick is about. We don't choose transformations $h(x)$, which we then use in a linear model. Instead we work with the kernel functions, which directly compute the inner product in the transformed feature space.