CSE 392: Matrix and Tensor Algorithms for Data Spring 2024

Lecture 3 — 01/24/2024

Instructor: Shashanka Ubaru *Scribe: Sung Jung*

1 Least Squares Regression

Least squares regression is a foundational technique in predictive modeling and data fitting, where we seek to find a functional relation between some input features and targets with respect to a certain loss.

To be specific, we are given a data matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ with n samples $\{\mathbf{a}_i\}_{i=1}^n \in \mathbb{R}$ of d-dimensional features and a column vector $\mathbf{b} \in \mathbb{R}^n$ of targets. Then, for a loss function $l(.,.)$ and a function $f(.,\theta)$ that is parameterized with a set of parameters θ over a possible set Θ, we seek to solve

$$\theta^* = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} l(f(\mathbf{a}_i, \theta), b_i) \tag{1}$$

In linear regression, the parameterized function that we seek to solve is simply the inner product of the feature with a vector of parameters (or coefficients)

$$f(\mathbf{a}_i, \mathbf{x}) = \mathbf{a}_i^T \mathbf{x} \tag{2}$$

and the observed targets are the prediction from this function plus some noise (or error or residual):

$$b_i = f(\mathbf{a}_i, \mathbf{x}) + e_i = \mathbf{a}_i^T \mathbf{x} + e_i \tag{3}$$

We can also write it in matrix form that combines all n samples into one:

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e} \tag{4}$$

where $\mathbf{a}_i$ is the i-th row of $\mathbf{A}$, and b_i and e_i are the i-th elements of the vectors $\mathbf{b}$ and $\mathbf{e}$, respectively.

In the *least-squares* regression problem, the loss function is simply the ℓ_2 norm of the error vector $\mathbf{e}$:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^d} ||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 \tag{5}$$

1.1 Normal Equation

The vector $\mathbf{x}^*$ minimizes $||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2$ if and only if it is the solution to the **normal equation**

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \tag{6}$$

To prove this, consider any $\tilde{\mathbf{x}} = \mathbf{x}^* + \Delta\mathbf{x} \in \mathbb{R}^d$. Then,

$$\begin{aligned} ||\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}||_2^2 &= ||\mathbf{A}\mathbf{x}^* + \mathbf{A}\Delta\mathbf{x} - \mathbf{b}||_2^2 && (7)\\ &= ||\mathbf{A}\mathbf{x}^* - \mathbf{b}||_2^2 + 2(\mathbf{A}\Delta\mathbf{x})^T(\mathbf{A}\mathbf{x}^* - \mathbf{b}) + ||\mathbf{A}\Delta\mathbf{x}||_2^2 && (8)\\ &= ||\mathbf{A}\mathbf{x}^* - \mathbf{b}||_2^2 + 2(\Delta\mathbf{x})^T\mathbf{A}^T(\mathbf{A}\mathbf{x}^* - \mathbf{b}) + ||\mathbf{A}\Delta\mathbf{x}||_2^2 && (9)\end{aligned}$$

Now, if $\mathbf{A}^T\mathbf{A}\mathbf{x}^* = \mathbf{A}^T\mathbf{b}$, then the above becomes

$$\begin{aligned} ||\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}||_2^2 &= ||\mathbf{A}\mathbf{x}^* - \mathbf{b}||_2^2 + 2(\Delta\mathbf{x})^T\mathbf{A}^T(\mathbf{A}\mathbf{x}^* - \mathbf{b}) + ||\mathbf{A}\Delta\mathbf{x}||_2^2 && (10)\\ &= ||\mathbf{A}\mathbf{x}^* - \mathbf{b}||_2^2 + ||\mathbf{A}\Delta\mathbf{x}||_2^2 && (11)\\ &\geq ||\mathbf{A}\mathbf{x}^* - \mathbf{b}||_2^2 && (12)\end{aligned}$$

for any $\Delta\mathbf{x}$. The inequality is saturated when $\Delta\mathbf{x} = 0$, or, equivalently, when $\tilde{\mathbf{x}} = \mathbf{x}^*$. Hence, $\mathbf{x}^*$ minimizes the loss function.

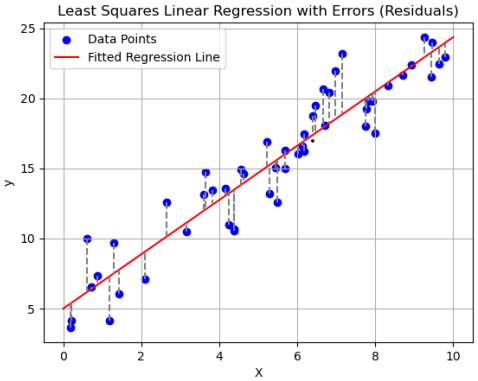
Conversely, let's suppose that $\mathbf{x}^*$ minimizes the loss function $||\mathbf{A}\mathbf{x}-\mathbf{b}||_2^2 = \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}-2\mathbf{x}^T\mathbf{A}^T\mathbf{b}+\mathbf{b}^T\mathbf{b}$. Since the loss function is quadratic with respect to $\mathbf{x}$ with a hessian of $2\mathbf{A}^T\mathbf{A}$ (a positive semidefinite matrix), we can find the global minimum by differentiating it and setting it equal to 0:

$$\begin{aligned} \frac{d}{d\mathbf{x}}(\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{b} + \mathbf{b}^T\mathbf{b}) &= 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b} && (13)\\ 2\mathbf{A}^T\mathbf{A}\mathbf{x}^* - 2\mathbf{A}^T\mathbf{b} &= 0 && (14)\\ \mathbf{A}^T\mathbf{A}\mathbf{x}^* &= \mathbf{A}^T\mathbf{b} && (15)\end{aligned}$$

So, $\mathbf{x}^*$ satisfies the normal equation.

1.1.1 Geometric Interpretation

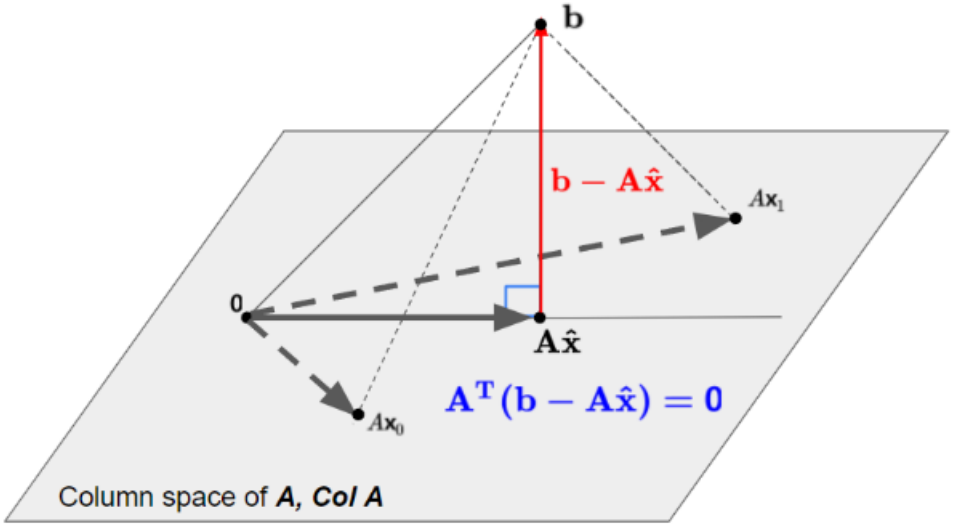
Conventionally, we visualize least-squares linear regression as a row picture, as written in Equation 3. Each observation is plotted on a graph at a point given by $(\mathbf{a}_i^T, b_i)$, and we try to find a line (hyperplane in higher dimensions) that fits all these points the best. For example, in the figure on the right, the $(\mathbf{a}_i^T, b_i)$ points are marked with blue markers, and the fitted regression line is drawn in red. The least-squares problem comes down to minimizing the sum of the squared lengths of the residuals, drawn with black dashed lines.



Alternatively, we can view least-squares linear regression as a column picture. With $\mathbf{a}_{(i)}$ as the i-th column of $\mathbf{A}$ and x_i as the i-th element of the coefficient vector $\mathbf{x}$, we can rewrite Equation 4 as

$$\mathbf{b} = \sum_{i=1}^{d} \mathbf{a}_{(i)} x_i + \mathbf{e} \tag{16}$$

In this picture, we seek to find the best linear combination of the columns of $\mathbf{A}$ that best approximates $\mathbf{b} \in \mathbb{R}^n$. If $\mathbf{A}$ has $\geq n$ linearly independent columns (which implies that $d \geq n$), then these columns form a basis in $\mathbb{R}^n$, and there is an exact solution $\mathbf{x}$ that fits $\mathbf{b}$ with $\mathbf{e} = \mathbf{0}$. However, if $\mathbf{A}$ only has r independent columns (with $r \leq d \leq n$), then these columns span some r-dimensional subspace within $\mathbb{R}^n$, so a vector in this subspace may not be able to fit $\mathbf{b}$ with zero error. The "best" approximation of $\mathbf{b}$ that is in this subspace spanned by $\{\mathbf{a}_{(i)}\}$ would be the one that minimizes the length (ℓ_2 norm) of $\mathbf{e}$.



As seen above, this occurs when $\mathbf{e}$ is orthogonal to the column space of $\mathbf{A}$, or, in other words,

$$\mathbf{A}^T \mathbf{e}^* = 0 \tag{17}$$

Substituting for $\mathbf{e}^* = \mathbf{b} - \mathbf{A}\mathbf{x}^*$ and rearranging, we recover the normal equation from Equation 6.

$$\mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}^*) = 0 \tag{18}$$

$$\mathbf{A}^T \mathbf{A}\mathbf{x}^* = \mathbf{A}^T \mathbf{b} \tag{19}$$

1.2 Least Squares via Maximum Likelihood

We can also reach the least-squares problem in the previous section through the maximum likelihood principle, in which the parameters that maximize the likelihood of observing the given sample are chosen as the estimator of the true parameters. This method is outlined below.

In the same setup as before with a set of parameters $\theta \in \Theta$, the likelihood function is defined as

$$\mathcal{L}(\theta) = \mathcal{L}(\theta; \mathbf{y}) = g_n(\mathbf{y}; \theta) \tag{20}$$

where $g_n(\mathbf{y}; \theta)$ is the joint probability density of observing the data sample $\mathbf{y} = (y_1, ..., y_n)$ given the parameters θ.

If the n samples are independent and identically distributed (i.i.d) random variables, then the joint probability density can be written as a product of univariate density functions:

$$g_n(\mathbf{y}; \theta) = \prod_{i=1}^{n} g(y_i; \theta) \tag{21}$$

In maximum likelihood estimation, we seek to find the θ that maximizes the likelihood function over Θ:

$$\theta^* = \arg\max_{\theta \in \Theta} \mathcal{L}(\theta) = \arg\max_{\theta \in \Theta} \prod_{i=1}^{n} g(y_i; \theta) \tag{22}$$

Equivalently, we can maximize the logarithm of the likelihood function, since the logarithm is a monotonic function (so does not change the argmax of the maximum) and it is more convenient to represent the product as a sum:

$$\theta^* = \arg\max_{\theta \in \Theta} \log(\prod_{i=1}^{n} g(y_i; \theta)) = \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \log(g(y_i; \theta)) \tag{23}$$

As seen above, this method requires that we know or assume the true underlying distribution of the samples as a function of the parameters. For least squares, we assume that the sample noise is i.i.d for across the n data points, each one being normally distributed with mean 0 and variance σ^2:

$$e_i \sim \mathcal{N}(0, \sigma^2) \tag{24}$$

Since $b_i = \mathbf{a}_i^T \mathbf{x} + e_i$, this means

$$b_i \sim \mathcal{N}(\mathbf{a}_i^T \mathbf{x}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(b_i - \mathbf{a}_i^T \mathbf{x})^2) \tag{25}$$

and so

$$\log(g(b_i; \mathbf{x})) = \log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{1}{2\sigma^2}(b_i - \mathbf{a}_i^T\mathbf{x})^2 \tag{26}$$

With this, Equation 23 becomes

$$\mathbf{x}^* = \arg\max_{x\in\mathbb{R}^d}(n\log(\frac{1}{\sqrt{2\pi\sigma^2}}) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(b_i - \mathbf{a}_i^T\mathbf{x})^2) \tag{27}$$

$$= \arg\max_{x\in\mathbb{R}^d}(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(b_i - \mathbf{a}_i^T\mathbf{x})^2)) \tag{28}$$

$$= \arg\min_{x\in\mathbb{R}^d}\sum_{i=1}^{n}(b_i - \mathbf{a}_i^T\mathbf{x})^2 \tag{29}$$

which is equivalent to Equation 5.

2 Penalized Regression

2.1 Issue with normal equation

Typically, $d \leq n$ with $\text{rank}(\mathbf{A}) = d$. If so, then $(\mathbf{A}^T\mathbf{A})^{-1}$ exists, and the solution to Equation 6 is

$$\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{30}$$

Here, we show that $(\mathbf{A}^T\mathbf{A})^T$ could be highly *ill-conditioned*.

The **condition number** of a matrix is defined as

$$\kappa_2(\mathbf{A}) = ||\mathbf{A}||_2||\mathbf{A}^{-1}||_2 = \sigma_{max}/\sigma_{min} \tag{31}$$

Then,

$$\kappa_2(\mathbf{A}^T\mathbf{A}) = ||\mathbf{A}^T\mathbf{A}||_2||(\mathbf{A}^T\mathbf{A})^{-1}||_2 = (\sigma_{max}/\sigma_{min})^2 \tag{32}$$

Suppose that $\mathbf{A}$ a condition number $\kappa_2(\mathbf{A}) = 1/\epsilon$, for $0 < \epsilon \leq 1$. Then, $\kappa_2(\mathbf{A}^T\mathbf{A}) = \epsilon^{-2}$.

2.2 Ridge Regression

One way to mitigate this issue is through *ridge regression* (also known as *Tikhonov regularization*).

In ridge regression, the loss function includes a term that penalizes large values of the coefficient vector $\mathbf{x}$:

$$\mathbf{x}_{rr} = \arg\min_{x \in \mathbb{R}^d}(||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \lambda||\mathbf{x}||_2^2) \tag{33}$$

where the strength of the penalty is determined by the regularization parameter $\lambda > 0$.

Notice the following:

$$||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \lambda||\mathbf{x}||_2^2 = || \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} ||_2^2 = ||\mathbf{A}_1\mathbf{x} - \mathbf{b}_1||_2^2 \tag{34}$$

Adding the penalty term in the loss function is equivalent to augmenting the data set with target values of 0. Expressing Equation 33 in terms of $\mathbf{A_1}$ and $\mathbf{b_1}$, we can solve for $\mathbf{x}_{rr}$ like a regular least-squares problem with the solution given by Equation 30:

$$\mathbf{x}_{rr} = (\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_1^T\mathbf{b_1} \tag{35}$$

$$= (\begin{bmatrix} \mathbf{A}^T & \sqrt{\lambda}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix})^{-1} \begin{bmatrix} \mathbf{A}^T & \sqrt{\lambda}\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \tag{36}$$

$$= (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{b} \tag{37}$$

What would be the singular values of $\mathbf{A}_1^T\mathbf{A}_1$? With the singular value decomposition of $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, we can see that they are the singular values of $\mathbf{A}^T\mathbf{A}$ shifted by λ:

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$$

$$\mathbf{A}_1^T\mathbf{A}_1 = \mathbf{A}^T\mathbf{A} + \lambda\mathbf{I} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T + \lambda\mathbf{V}\mathbf{V}^T = \mathbf{V}(\mathbf{\Sigma}^2 + \lambda\mathbf{I})\mathbf{V}^T$$

So, the condition number of $\mathbf{A}_1^T\mathbf{A}_1$ is

$$\kappa_2(\mathbf{A}_1^T\mathbf{A}_1) = \frac{\sigma_{max}^2 + \lambda}{\sigma_{min}^2 + \lambda} = \frac{\sigma_{max}^2}{\sigma_{min}^2} \cdot \frac{1 + \frac{\lambda}{\sigma_{max}^2}}{1 + \frac{\lambda}{\sigma_{min}^2}} \leq \frac{\sigma_{max}^2}{\sigma_{min}^2} = \kappa_2(\mathbf{A}^T\mathbf{A}) \tag{38}$$

By decreasing the condition number of the matrix being inverted in Equation 30, we have made the solution more numerically stable to calculate.

2.3 LASSO Regression

Another popular penalized regression is the LASSO regression, which stands for *Least Absolute Shrinkage and Selection Operator*. Unlike ridge regression, which adds an ℓ_2 penalty to the coefficients, LASSO adds an ℓ_1 penalty:

$$\mathbf{x}_{lasso} = \arg\min_{x \in \mathbb{R}^d}(||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2 + \lambda||\mathbf{x}||_1) \tag{39}$$

This problem is still convex, but it is non-smooth, and its solution cannot be found using the augmented linear model method we used above for ridge regression. However, many efficient optimization algorithms have been proposed, such as the Fast Iterative Shrinkage Algorithm (FISTA) and the Alternating Direction Method of Multipliers (ADMM).

One interesting aspect of the LASSO regression is that unlike ridge regression, which tends to shrink all parameter estimates towards 0, LASSO regression is highly likely to produce a *sparse solution* - where some estimates are exactly 0. Consequently, LASSO regression can be used for variable selection - when only a few parameters are desired to be kept in the model.

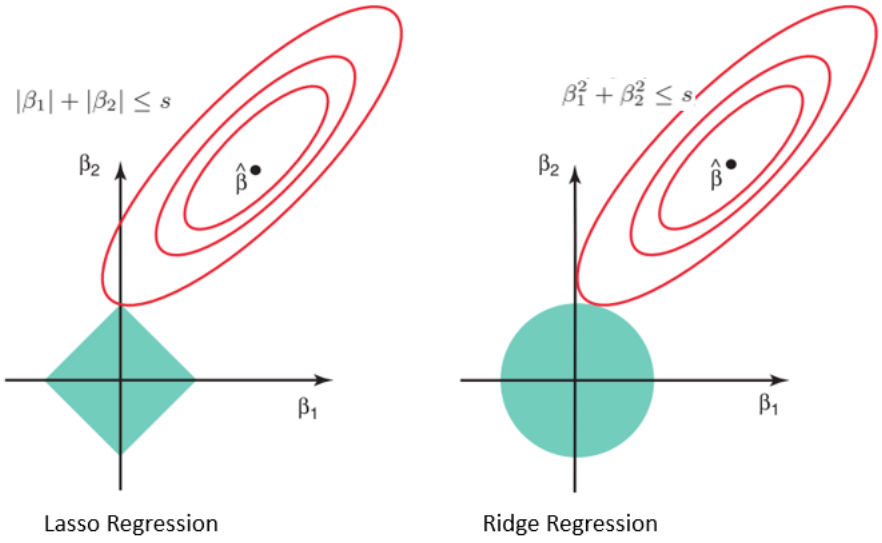
The following visualization reveals why LASSO regression is likely to yield a sparse solution. Firstly, there is a one-to-one correspondence between the Lagrangian form of the LASSO loss function (Equation 39) and the constrained optimization form:

$$\mathbf{x}_{lasso} = \underset{x \in \mathbb{R}^d}{\arg\min} \, ||\mathbf{Ax} - \mathbf{b}||_2^2 \text{ , subject to } ||\mathbf{x}||_1 \leq s \tag{40}$$

where λ and s are related given $\mathbf{A}$ and $\mathbf{b}$ (see here). A similar thing can be said for the ridge regression loss function:

$$\mathbf{x}_{rr} = \underset{x \in \mathbb{R}^d}{\arg\min} \, ||\mathbf{Ax} - \mathbf{b}||_2^2 \text{ , subject to } ||\mathbf{x}||_2^2 \leq s \tag{41}$$

If we plot the contours of $||\mathbf{Ax} - \mathbf{b}||_2^2$ (red) with the region of the feature space that satisfies the constraints (cyan), we get the following (β has replaced $\mathbf{x}$ in the figure below):



The solution to the penalized regression problem is the point in the feature space that is on the lowest possible contour (red) that "hits" the shaded region (cyan). In LASSO, this region is shaped like a diamond (polyhedron in higher dimensions) because the constraint is an ℓ_1 norm, and it is very easy for the contour to hit a corner of this diamond (edge of a polyhedron in higher dimensions).

As a result, the figure above has shrunk β_1 to exactly 0, selecting only β_2 as the nonzero parameter. In contrast, since the cyan region of ridge regression is circular (ℓ_2 norm), it is almost impossible for the contour to meet the region where $\beta_i = 0$. As a result, it only brings all the parameters closer to the origin but not completely to 0.

3 Kernel Methods

3.1 Feature Maps

Linear regression fits a linear function to the data. However, oftentimes, the true functional relationship may be nonlinear, so we seek to increase the complexity of the model. For example, consider fitting a cubic function:

$$b = x_0 + ax_1 + a^2x_2 + a^3x_3$$

Although this function is nonlinear with respect to the one-dimensional input a, we can view it as a linear function over a set of higher-dimensional features that are created from the lower-dimensional input. For instance, let the map $\phi : \mathbb{R} \to \mathbb{R}^4$ be defined as:

$$\phi(a) = [1, a, a^2, a^3] \tag{42}$$

If $\mathbf{x} = [x_0, x_1, x_2, x_3]^T$, then

$$b = \phi(a)^T\mathbf{x} \tag{43}$$

The function ϕ is called the **feature map**.

3.2 Kenalization & Kernel Ridge Regression

Consider a ridge regression using such a feature map. With the transformation $\mathbf{a}_i \in \mathbb{R}^d \to \Phi_i = \phi(\mathbf{a}_i) \in \mathbb{R}^m$ and the matrix $\Phi \in \mathbb{R}^{n\times m}$ whose i-th row is Φ_i, the solution to the ridge regression is

$$\mathbf{x}_{kr} = (\Phi^T\Phi + \lambda\mathbf{I}_m)^{-1}\Phi^T\mathbf{b} \tag{44}$$

where $\mathbf{I}_m$ is an $(m \times m)$ identity matrix. Here, we must invert an $(m \times m)$-dimensional matrix, but m (the dimension of the feature map) may be extremely large - maybe even infinite! To make this feasible, we use the following identity, given that the inverses of $\mathbf{P}$ and $\mathbf{R}$ exist:

$$(\mathbf{P}^{-1} + \mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^T(\mathbf{B}\mathbf{P}\mathbf{B}^T + \mathbf{R})^{-1} \tag{45}$$

Refer here for the proof.

If we take $\mathbf{P} = \frac{1}{\lambda}\mathbf{I}_m$, $\mathbf{B} = \Phi$, and $\mathbf{R} = \mathbf{I}_n$, then Equation 45 becomes

$$(\lambda \mathbf{I}_m + \Phi^T\Phi)^{-1}\Phi^T\mathbf{b} = \frac{1}{\lambda}\Phi^T(\frac{1}{\lambda}\Phi\Phi^T + \mathbf{I}_n)^{-1}\mathbf{b} \tag{46}$$

$$= \Phi^T(\Phi\Phi^T + \lambda\mathbf{I}_n)^{-1}\mathbf{b} \tag{47}$$

$$= \Phi^T(\mathbf{K} + \lambda\mathbf{I}_n)^{-1}\mathbf{b} \tag{48}$$

Now, the matrix that must be inverted is $(n \times n)$, where n is the number of data points. If the feature mapping is very high dimensional, the new $(n \times n)$ matrix may be cheaper to invert.

In the process, we have introduced the kernel (or Gram) matrix $\mathbf{K}$, where $K_{ij} = \Phi_i^T\Phi_j = \phi(\mathbf{a}_i)^T\phi(\mathbf{a}_j)$. We will also use the notation $K(\mathbf{a}_i, \mathbf{a}_j) \equiv \phi(\mathbf{a}_i)^T\phi(\mathbf{a}_j)$, called the **kernel**.

A key property of a kernel is stated by the **Mercer Theorem**:

Theorem 1. *Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be given. Then, for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{\mathbf{a}_1, ..., \mathbf{a}_n\}$, $(n < \infty)$, the corresponding kernel matrix is symmetric and positive semi-definite (PSD).*

Proof. $\Rightarrow$ Let the kernel matrix $\mathbf{K}$ be defined as $K_{ij} = \phi(\mathbf{a}_i)^T\phi(\mathbf{a}_j)$. If K is a valid kernel, then $K(\mathbf{a}_i, \mathbf{a}_j) = \phi(\mathbf{a}_i)^T\phi(\mathbf{a}_j) = \phi(\mathbf{a}_j)^T\phi(\mathbf{a}_i) = K(\mathbf{a}_j, \mathbf{a}_i)$. Furthermore, for any vector $\mathbf{z}$, we have:

$$\mathbf{z}^T\mathbf{K}\mathbf{z} = \sum_i\sum_j z_i K_{ij} z_j = \sum_i\sum_j z_i\phi(\mathbf{a}_i)^T\phi(\mathbf{a}_j)z_j \tag{49}$$

$$= \sum_i\sum_j\sum_k z_i\phi_k(\mathbf{a}_i)\phi_k(\mathbf{a}_j)z_j = \sum_k\sum_i z_i\phi_k(\mathbf{a}_i)\sum_j z_j\phi_k(\mathbf{a}_j) \tag{50}$$

$$= \sum_k(\sum_i z_i\phi_k(\mathbf{a}_i))^2 \geq 0 \tag{51}$$

So, $\mathbf{K}$ is PSD.

$\Leftarrow$ Let a matrix $\mathbf{K}$ be PSD. Then, we can compute its eigendecomposition as $\mathbf{K} = \mathbf{U}\Lambda\mathbf{U}^T$ with Λ being a diagonal matrix of the eigenvalues of $\mathbf{K}$, which are all ≥ 0.

Let $\Lambda^{1/2}$ be a diagonal matrix of the square roots of the eigenvalues of $\mathbf{K}$. Then,

$$\mathbf{K} = \mathbf{U}\Lambda^{1/2}(\Lambda^{1/2})^T U^T = (\mathbf{U}\Lambda^{1/2})(\mathbf{U}\Lambda^{1/2})^T \tag{52}$$

$$K_{ij} = (\mathbf{U}_{i*}\Lambda^{1/2})(\mathbf{U}_{j*}\Lambda^{1/2})^T \tag{53}$$

Define $\phi(\mathbf{a}_i) \equiv (\mathbf{U}_{i*}\Lambda^{1/2})^T$. Then, the above can be rewritten as $K_{ij} = \phi(\mathbf{a}_i)^T\phi(\mathbf{a}_j)$

■

The kernel is an inner product between two feature maps. Therefore, intuitively, when $\phi(\mathbf{a}_i)$ and $\phi(\mathbf{a}_j)$ are close to each other, the kernel $K(\mathbf{a}_i, \mathbf{a}_j)$ should be large. Conversely, if they are far apart,

it should be small. Thus, the kernel can be thought of as a similarity measure of the features. One popular kernel is the **Gaussian Kernel**, which is defined as:

$$K(\mathbf{a}_i, \mathbf{a}_j) = \exp(-\frac{||\mathbf{a}_i - \mathbf{a}_j||}{2\sigma^2}) \tag{54}$$

and it corresponds to an infinite dimensional feature map ϕ.

If the Gaussian kernel corresponds to an infinite dimensional feature map, can we even use it in kernel ridge regression? Equation 48 shows that we still need to have Φ^T, which would be $(\infty \times n)$.

Luckily, not quite. Given a *new* data point $\mathbf{a}$, the prediction would be

$$\hat{b} = \phi(\mathbf{a})^T \mathbf{x}_{kr} = \phi(\mathbf{a})^T \Phi^T (\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{b} = \kappa(\mathbf{a})(\mathbf{K} + \lambda \mathbf{I}_n)^{-1} \mathbf{b} \tag{55}$$

where

$$\kappa(\mathbf{a}) = \phi(\mathbf{a})^T \Phi^T = [\phi(\mathbf{a})^T \phi(\mathbf{a}_1), ..., \phi(\mathbf{a})^T \phi(\mathbf{a}_n)] = [K(\mathbf{a}, \mathbf{a}_1), ..., K(\mathbf{a}, \mathbf{a}_n)]$$

In other words, we do not need to explicitly define or compute the feature map ϕ; we can only work with the kernel function $K(.,.)$. If the input enters an algorithm only in the form of inner products, then we can replace the inner product with some kernel - this is called the **kernel trick**.

3.2.1 Aside

More generally to kernel ridge regression, it can be shown that for any matrix $\mathbf{A}$,

$$(\mathbf{A}^T \mathbf{A})^\dagger \mathbf{A}^T = \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^\dagger$$

where $\mathbf{X}^\dagger$ is the Moore-Penrose pseudoinverse of $\mathbf{X}$. The proof involves expressing $\mathbf{A}$ as its singular value decomposition to compute transposes, pseudoinverses, and matrix multiplication.