Chapter 6:
Regression

Up until now we only had a look at classification problems. For our prediction function f we had that $f(x) \in \{-1, +1\}$, but real word problems are not only of the classification kind.
Suppose we are given houses as training data, and we want to predict the price of a house. A binary classification doesn't suffice we need a real valued prediction function.

The formal setting will be the following:

- Given training data $(\mathbf{x_i}, (y_i))$, with $\mathbf{x_i} \in \mathbb{R}^d, y_i \in \mathbb{R}$.
- Find $f : \mathbb{R}^d \to \mathbb{R}$ with $f(\mathbf{x}) \approx y$ for new data $\mathbf{x}, y$.

In the following we will look at an old method of solving these kind of problems, called Regression.

6.1 Linear Regression

The idea of linear regression is quite simple.
Suppose we are given some data points. A first idea would be to approximate these values linearly, by a hyperplane

$$\mathbf{w}^T\mathbf{x}.$$

For now we will intentionally ignore the displacement parameter $+b$ as we will see later that we don't need it.

The line would be good if it reduces the error. One way to measure the error of datapoint i would be the squared distance

$$(y_i - \mathbf{w}^T\mathbf{x_i})^2.$$

Observe that it is an intuitive choice of error, it is always positive and it is higher if the value predicted by the hyperplane and y_i are far apart. Finally we want to minimize the error for all datapoints, so we just sum over them, leading us to the following optimization problem.

Least square regression (Legendre)

$$\mathbf{w}_{\text{LS}} := \arg\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \mathbf{w}^\top \mathbf{x}_i\right)^2 = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \left\|\mathbf{y} - X^\top \mathbf{w}\right\|^2$$

We will also define a loss function, exhibiting our choice of error:

Definition : Least square loss
The function $\ell(t, y) := (t-y)^2$ is called least-squares loss. Our idea so far is good

but is prone to overfitting, because so far our model is only loss concentrated. To make our model less prone to overfitting we add our known regularizer, to end up with:

Ridge regression:

$$\mathbf{w}_{\mathrm{RR}} := \arg\min_{\mathbf{w}\in\mathbb{R}^d} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2 + C\left\|\mathbf{y} - X^\top \mathbf{w}\right\|^2}_{=:\mathcal{L}(\mathbf{w})}$$

We can now talk about how to find our optimum. One way would be to use SGD, but it turns out that our problem is much simpler.

Ridge regression is an unconstrained optimization problem. Thus we find the global optima by differentiation. Let us calculate:

$$\begin{aligned}
&\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) \\
&\quad = \nabla_{\mathbf{w}}\left(\frac{1}{2}\|\mathbf{w}\|^2 + C\left\|\mathbf{y} - X^\top \mathbf{w}\right\|^2\right) \\
&\quad = \nabla_{\mathbf{w}}\left(\frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\left(\mathbf{y} - X^\top \mathbf{w}\right)^\top\left(\mathbf{y} - X^\top \mathbf{w}\right)\right) \\
&\quad = \nabla_{\mathbf{w}}\left(\frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\left(\mathbf{y}^\top - \mathbf{w}^\top X\right)\left(\mathbf{y} - X^\top \mathbf{w}\right)\right) \\
&\quad = \nabla_{\mathbf{w}}\left(\frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\left(\mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top X^\top\mathbf{w} - \mathbf{w}^\top X\mathbf{y} + \mathbf{w}^\top XX^\top\mathbf{w}\right)\right) \\
&\quad = \nabla_{\mathbf{w}}\left(\frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\mathbf{y}^\top\mathbf{y} - 2C\mathbf{w}^\top X\mathbf{y} + C\mathbf{w}^\top XX^\top\mathbf{w}\right) \\
&\quad = \mathbf{w} - 2CX\mathbf{y} + 2CXX^\top\mathbf{w}
\end{aligned}$$

Finally we set $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = 0$ to find our optimum.

$$\begin{aligned}
\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = 0 \\
&\iff \mathbf{w} - 2CX\mathbf{y} + 2CXX^\top\mathbf{w} = 0 \\
&\iff -2CX\mathbf{y} + (I + 2CXX^\top)\mathbf{w} = 0 \\
&\iff (I + 2CCXX^\top)\mathbf{w} = 2CX\mathbf{y} \\
&\iff \mathbf{w} = (I + 2CXX^\top)^{-1}2CX\mathbf{y} \\
&\iff \mathbf{w} = 2C(I + 2CXX^\top)^{-1}Xy \\
&\iff \mathbf{w} = \left(\frac{1}{2C}I + XX^\top\right)^{-1}Xy
\end{aligned}$$

Theorem: 6.1.1

$$\mathbf{w}_{\mathrm{RR}} = \left(XX^\top + \frac{1}{2C}I\right)^{-1}Xy$$

We have considered a linear model without bias b.
However, we can easily incorporate a bias into any linear learning machine (regression, SVM, etc.) by the following trick:
Let $\mathbf{x_1}, \dots \mathbf{x_n}$ be our training data.
Let

$$\tilde{\mathbf{x}}_i := \begin{pmatrix} \mathbf{x}_i \\ 1 \end{pmatrix}$$

and change our training data matrix into

$$\tilde{X} := (\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n) = \begin{pmatrix} X \\ \mathbf{1}^\top \end{pmatrix}$$

then we can conclude that every solution :

$$\begin{aligned} \mathbf{w}^* :=& \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{2}\|\tilde{\mathbf{w}}\|^2 + C\left\|\mathbf{y} - \tilde{X}^\top \tilde{\mathbf{w}}\right\|^2 \\ =& \arg\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2}\left(\|\mathbf{w}\|^2 + b^2\right) + C\left\|\mathbf{y} - X^\top \mathbf{w} - b\mathbf{1}\right\|^2 \end{aligned}$$

6.2 Leave one out cross validation (LOOCV)

Let us think about how we can choose our regularization constant C. An idea would be to separate our training data set, into a smaller training data set and a test set. We would then measure the average error and choose our best regularization constant. To make things fair we would also alternate through all possible test sets and training sets. Let us make this idea more formal in the following algorithm.

Algorithm k**-fold Cross Validation**

1: split data into $k \overset{\text{e.g.}}{=} 10$ equally-sized chunks (called "folds")
2: **for** $t \leftarrow 1$ to k and $C \overset{\text{e.g.}}{\in} \{0.01, 0.1, 1, 10, 100\}$ **do**
3: use ith fold as test set and union of all others as training set
4: train learner on training set (using C) and test on test set
5: **end for**
6: output learner with lowest average error

The meaning of error is clear in the classification problem, but what is an error in the regression setting? One way to define error in the regression setting would be:

The root mean square error:
Let $\{(\mathbf{x}_1, y_1), \dots (\mathbf{x}_n, y_n)\}$ be a test set, and let f be a learned regression function. The root mean squared error (RMSE) of f is:

$$\text{RSME}(f) := \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(f\left(\mathbf{x}_i\right) - y_i\right)^2}$$

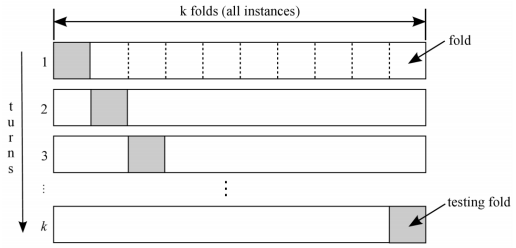

Figure 1: Visualization of the k-fold Algorithm

Sometimes our dataset is very small, in such a dataset itwould be intuitive to choose $k = n$.

Leave one out cross validation (LOOCV)
The special case of $k = n$ in the k-fold Cross Validation is called, leave one out cross validation.

We can use LOOCV also for deciding other parameters like kernel width or learning rate in ANN. Sadly the runtime of LOOCV is relatively high.
In Ridge Regression we loop over all data points. $O(n)$
Each time we train, with $n-1$ data points, so we need to invert a matrix. $O(d^3)$
Resulting in total runtime of $O(nd^3)$.

Turns out, we can be faster.
Let us look at the particular setting of LOOCV.
Recall:

$$\mathbf{w}_{\text{RR}} = (\underbrace{XX^\top}_{=\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top} + \frac{1}{2C} l)^{-1} \underbrace{Xy}_{=\sum_{i=1}^n \mathbf{x}_i y_i}$$

Let $\mathbf{w_i}$ be the RR solution when ith data point is left out at training.
We have

$$\mathbf{w}_i = \left(XX^\top - \mathbf{x}_i \mathbf{x}_i^\top + \frac{1}{2C} I \right)^{-1} (Xy - \mathbf{x}_i y_i)$$

The error in LOOCV is defined as :

$$\text{RSME}_{\text{loocv}} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}_i^\top \mathbf{w}_i - y_i \right)^2}$$

We will now see, that we actually don't need to invert n times. We will use the following theorem.
Theorem: Sherman-Morrison Formula
Let $A \in \mathbb{R}^{d \times d}$ be an invertible matrix, and let $\mathbf{u} \in \mathbb{R}^d$. If $\mathbf{u}^\top A^{-1} \mathbf{u} \neq 1$, then:

$$\left(A - \mathbf{u}\mathbf{u}^\top \right)^{-1} = A^{-1} + \frac{A^{-1} \mathbf{u}\mathbf{u}^\top A^{-1}}{1 - \mathbf{u}^\top A^{-1} \mathbf{u}}$$

First let us rewrite:

$$\mathbf{w}_i = \left(\underbrace{XX^\top + \frac{1}{2C} I}_{:=A} - \mathbf{x}_i \mathbf{x}_i^\top \right)^{-1} (Xy - \mathbf{x}_i y_i)$$

By using the theorem, we get :

$$\begin{aligned} \text{RSME}_{\text{loocv}} &= \sqrt{\sum_{i=1}^n \left(\mathbf{x}_i^\top \mathbf{w}_i - y_i \right)^2} \\ &= \sqrt{\sum_{i=1}^n \left(\mathbf{x}_i^\top \left(A^{-1} + \frac{A^{-1}\mathbf{x}_i \mathbf{x}_i^\top A^{-1}}{1 - \mathbf{x}_i^\top A^{-1} \mathbf{x}_i} \right) (Xy - \mathbf{x}_i y_i) - y_i \right)^2} \end{aligned}$$

This shows we only need $O(d^3)$ iterations, to compute LOOCV error.But we can further simplify the equation.

Theorem: 6.2.1

The LOOCV-RMSE of ridge regression can be computed in $O\left(d^3\right)$ through:

$$\text{RSME}_{\text{loocv}} = \sqrt{\sum_{i=1}^n \left(\frac{\mathbf{x}_i^\top \mathbf{w}_{\text{RR}} - y_i}{1 - \mathbf{x}_i^\top A^{-1} \mathbf{x}_i} \right)^2}$$

where $A := XX^\top + \frac{1}{2C}\text{I}$.

Proof:

Recalling $\mathbf{w}_{\text{RR}} = A^{-1}Xy$ and denoting $\beta_i := \mathbf{x}_i^\top A^{-1} \mathbf{x}_i$, it is:

$$\begin{aligned} \text{RSME}_{\text{loocv}} &= \\ &= \sqrt{\sum_{i=1}^n \left(\mathbf{x}_i^\top \left(A^{-1} + \frac{A^{-1}\mathbf{x}_i \mathbf{x}_i^\top A^{-1}}{1 - \mathbf{x}_i^\top A^{-1} \mathbf{x}_i} \right) (Xy - \mathbf{x}_i y_i) - y_i \right)^2} \\ &= \sqrt{\sum_{i=1}^n \left(\mathbf{x}_i^\top \mathbf{w}_{RR} + \frac{\beta_i \mathbf{x}_i^\top \mathbf{w}_{R\text{R}}}{1 - \beta_i} - \beta_i y_i - \frac{\beta_i^2}{1 - \beta_i} y_i - y_i \right)^2} \\ &= \sqrt{\sum_{i=1}^n \left(\left(1 + \frac{\beta_i}{1 - \beta_i} \right) \mathbf{x}_i^\top \mathbf{w}_{\text{RR}} - \left(\beta_i + \frac{\beta_i^2}{1 - \beta_i} + 1 \right) y_i \right)^2} \\ &= \sqrt{\sum_{i=1}^n \left(\frac{\mathbf{x}_i^\top \mathbf{w}_{\text{RR}} - y_i}{1 - \beta_i} \right)^2} = \sqrt{\sum_{i=1}^n \left(\frac{\mathbf{x}_i^\top \mathbf{w}_{\text{RR}} - y_i}{1 - \mathbf{x}_i^\top A^{-1} \mathbf{x}_i} \right)^2} \quad \square \end{aligned}$$

6.3 Non-linear Regression

As we can suspect, a linear function is not always the best prediction function. We will try now to employ non-linearity into regression.

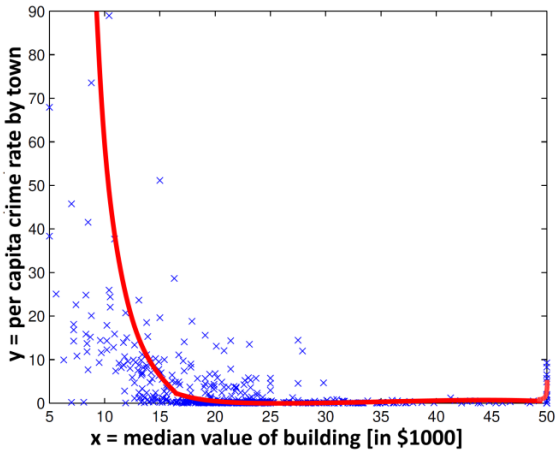

Figure 2: The red non-linear function, approximates this dataset much better than any linear function would.

But firstly let us try a quick fix, with which we could still use linear regression. Consider applying a log transformation on the label $y = log(y)$. We have a look

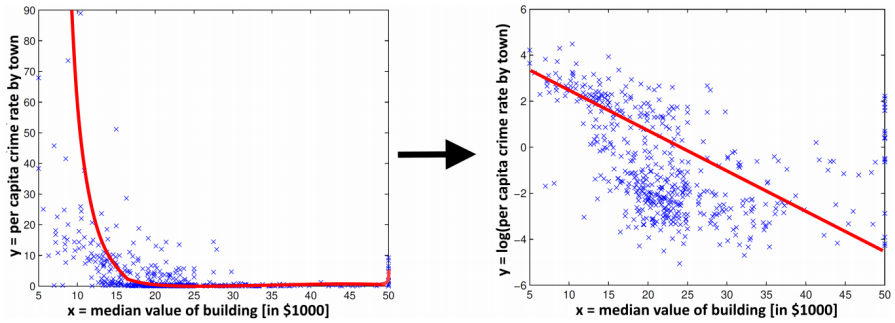

Figure 3: After the log transformation, a linear regression line is a reasonable approach.

at a generalization of the above method.

Box Cox Transformation
The Box-Cox transformation (or 'power transformation') with parameter λ is:

$$x_{\text{new}} \leftarrow \begin{cases} \log(x_{\text{old}}) & \text{if } \lambda = 0 \\ \frac{x_{01}^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0 \end{cases}$$

The parameter λ has intuitive interpretations, as follows:

$\lambda > 1:$	data is stretched (e.g $\cdot\ \lambda = 2:$ quadratically)
$0 < \lambda < 1:$	data is concentrated (e.g. $\lambda = 0.5:$ square root)
$\lambda = 0:$	log transform
$\lambda < 0:$	analogously, with the order of data reversed

Generally before using any other method, one would try out some transforma-

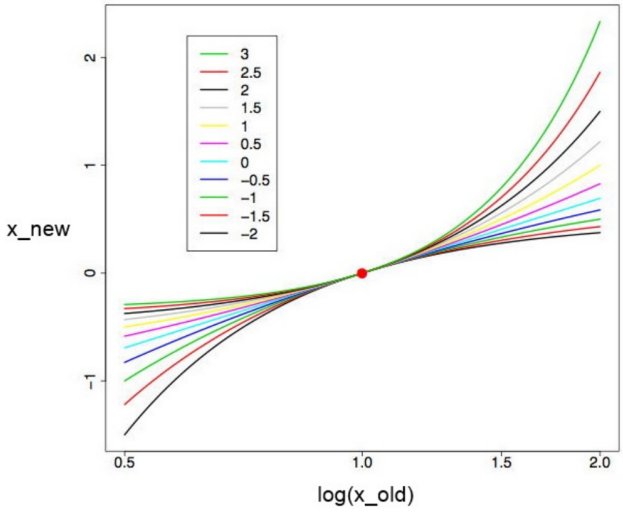

Figure 4: The parameters λ and their curves.

tions on the dataset. If this still doesn't help we can use kernel ridge regression.

Definition: Kernel Ridge Regression
Let k be a kernel with associated feature map $\phi : \mathbb{R}^d \to \mathcal{H}$, i.e., $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle$. Then kernel ridge regression (KRR) is defined as:

$$\mathbf{w}_{KRR} := \arg\min_{\mathbf{w} \in \mathcal{H}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \|y_i - \langle \mathbf{w}, \phi(x_i) \rangle\|^2$$

Let us try to use the kernel trick onto KRR.

Recall the representer theorem statement, the optimal solution is in the span of the dataset:

$$\exists \boldsymbol{\alpha} \in \mathbb{R}^n : \quad \mathbf{w}_{KRR} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) = \phi(X)\boldsymbol{\alpha}$$

with

$$\phi(X) := (\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n))$$

Recall, the definition of the Kernel matrix K, we have :

$$\begin{aligned}
&\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}\|^2 + C\left\|\mathbf{y} - \phi(X)^\top \mathbf{w}\right\|^2 \\
&= \min_{\alpha\in\mathbb{R}^n} \frac{1}{2} \qquad \underbrace{\|\phi(X)\alpha\|^2}_{\boldsymbol{\alpha}^\top \underbrace{\phi(X)^\top \phi(X)}_{=K} \boldsymbol{\alpha}} \qquad + C\|\mathbf{y} - \underbrace{\phi(X)^\top \phi(X)}_{=K}\alpha\|^2 \\
&= \min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} + C\|\mathbf{y} - K\boldsymbol{\alpha}\|^2
\end{aligned}$$

By differentiation we can also find the optimal solution, leading us to the following theorem.

Theorem 6. 3.1
The solution of KRR is given by

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \quad \text{with} \quad \alpha = \left(K + \frac{1}{2C} I_{n\times n}\right)^{-1} y$$

It can thus be computed in $O\left(n^3\right)$. Usually we use KRR together with a Gaussian kernel, but sometimes it can make sense to use a linear kernel:
By definition linear kernel regression is the same as ridge regression.
If $n < d$ then it makes sense to use a linear kernel instead of ridge regression, as the matrix we need to invert is of size $n \times n$ as seen above, and the matrix inversion is in $O(n^3)$ time instead of $O(d^3)$.

We also use regression in deep learning, for tasks like deciding the price of a house, given a picture . We only need to change the loss to least squares.
Deep Regression
Predict $f(\mathbf{x}) = \mathbf{w}_*^\top \phi_{W_*}(\mathbf{x})$, where:

$$(\mathbf{w}_*, W_*) := \min_{\mathbf{w},W} \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{l=1}^{L} \|W_l\|_{\text{Fro}}^2 + C\sum_{i=1}^{n} \left(y_i - \mathbf{w}^\top \phi_W(\mathbf{x}_i)\right)^2$$

6.4 Unifying View

Recall our unifying equation:

$$\min_{[W,]b,\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \ell\left(y_i\left(\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle + b\right)\right) \left[+\frac{1}{2}\sum_{l=1}^{L} \|W_l\|_{\text{Fro}}^2\right] \tag{UE}$$

We would like to also regression into this equation, making it further more general. In order to do so, we define new notation for our loss function:

$$l(t, y) := \begin{cases} (t-y)^2 & \text{for regression} \\ \ell(yt) & \text{for classification} \end{cases}$$

We can now get all of our known loss functions, by plugging in:

- Use $l(t, y) := \max(0, 1 - yt)$ for SVM ("hinge loss")
- Use $l(t, y) := \ln(1 + \exp(-yt))$ for LR and ANN("logistic loss")
- Use $l(t, y) := (t - y)^2$ for regression

Similiarly, by plugging the following kernel function in each loss functions from above, into UE, we get respectively:

- $\phi :=$ id for linear SVM, linear LR, and RR
- $\phi := \phi_k$ for kernel SVM, kernel LR, and KRR
- $\phi := \phi_W$ for ANN and DR

Support Vector Regression (SVR)
Consider the new loss function called ϵ **insensitive loss**

$$\ell(t, y) := \max(0, |y - t| - \epsilon)$$

We can now define the support vector regression, whose aim it is to find a line predicting $f(x_i)$, such that the predicted value doesn't deviate more than ϵ to y_i. This error is captured by the distance of the prediction line and y_i.

$$\mathbf{w}^*_{\text{SVR}} := \arg\min_{\mathbf{w} \in \mathcal{H}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \max\left(0, |y_i - \langle \mathbf{w}, \phi\left(\mathbf{x}_i\right)\rangle| - \epsilon\right)$$