Lab 2 - Least Squares Regression

Main topics:

- Gaussian random variables and vectors
- Generalization vs training loss
- Least squares regression

In this lab you will see how to generate both gaussian random variables and gaussian random vectors.
Moreover you will impement the least squares solution for a simple 1-dimensional regression problem.
We are going to compare training and generalization error $\mathcal{L}_D(h)$ and training error $\mathcal{L}_S(h)$.

Let's generate gaussian random vectors using numpy

Generating gaussian random varaibles or vectors is extremely simple. To begin with remember that a gaussian distribution is parametrized by its mean and variance (covariance for multidimensional distribution). Therefore these 2 parameters uniquely characterize a gaussian distribution (once you fix them you will obtain the same distribution).

For the sake of simplicity let's start with a 1-d random variable. Let's say that we are willing to generate the outcomes $x_i \in \mathbb{R}\ i = 1, \dots, N$ of the gaussian random variable $X \sim \mathcal{N}(\mu, \sigma^2)$, assume we know both μ and σ.

There are plenty of ways to generate such outcomes! The easiest way is to exploit a property of gaussian distribution (which we have seen during the probability review): guassian distributions are closed under affine transformations! This means that if you apply an affine transformation (say $aX + b, a \in \mathbb{R}, b \in \mathbb{R}$) to a gaussian random variable the new transformed random variable will once again be a gaussian distribution. We transformed the random variable with the affine transformation, this means its parameters which uniquely describe it (mean and variance) are changed. Is it possible to find the new parameters once you know the starting parameters and the affine transformation?
You should remember the relation by heart, but for the sake of clarity let's recall them. Let $Y := aX + b$ be the new gaussian random variable. Therefore $\mathbb{E}[Y] = a\mathbb{E}[X] + b$, $Var[X] = \mathbb{E}[(aX + b - \mathbb{E}[aX + b])^2] = a^2\mathbb{E}[(X - \mathbb{E}[X])^2] = a^2 Var[X]$. Therefore we have $Y \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$. Boring eh?

Let's make a though experiment, say someone asked you to build a gaussian distribution with mean μ_{new} and variance σ^2_{new} starting from another gaussian (say $X \sim \mathcal{N}(\mu, \sigma^2)$), what would you do?
The easiest choice would be to apply an affine transformation to X, how to choose the proper affine transformation?
You only need to find a, b imposing the following: $\mu_{new} = a\mu + b$ and $\sigma^2_{new} = a^2\sigma^2$. $\implies a = \frac{\sigma_{new}}{\sigma}$ and $b = \mu_{new} - \frac{\sigma_{new}}{\sigma}\mu$.

In practice it is super easy to generate by means of a computer standard gaussian random variables (zero mean and unit variance). This means that you can create **any** guassian random variable starting from a standard one applying an affine transformation with a and b chosen as: $a = \sigma_{new}$ and $b = \mu_{new}$.

For now you have seen the go to strategy for generating gaussian scalar random variables! Can you generate in a similar way vector valued gaussian random variables?

How to generate gaussian random vectors of given mean and covariance?

Say you are given a mean vectors μ_{new} and a covariance matrix Σ_{new} (p.s.d.), how to generate a gaussian random vector with the proper mean and covariance?

Consider now a gaussian random vector $X \in \mathbb{R}^n$ (note we are using the same notation, capital letters, to denote random variables and vectors of random variables). Once again mean and covariance fully describe the gaussian random vector, from now on let $X \sim \mathcal{N}(\mu, \Sigma)$. Remember also in this case an affine transformation of a gaussian random vector is a gaussian random vector.

Therefore the same reasoning we made earlier is applicable here!

Let's compute mean and covariance of a gaussian random vector after the application of an affine transformation.

Remark: The affine transformation now is parametrized by the matrix $A \in \mathbb{R}^{n,n}$ and the vector $b \in \mathbb{R}^n$: $Y = AX + b$.

From the theory we know Y is a gaussian random variable. It is straightforward to compute its mean (use linearity of the expected value): $\mathbb{E}[Y] = A\mathbb{E}[X] + b$ (exactly as in the scalar case).

Now recall the definition of covariance for a vector valued random variable:
$Cov[Y, Y] = \mathbb{E}[(Y - \mathbb{E}[Y])(Y - \mathbb{E}[Y])^T]$, note what is happening here. We are multiplying $(Y - \mathbb{E}[Y]) \in \mathbb{R}^{n \times 1}$ (column ector) and $(Y - \mathbb{E}[Y]) \in \mathbb{R}^{1 \times n}$ (row vector), the result is a $n \times n$ matrix. That is why the covariance $Cov[Y, Y]$ is a $n \times n$ matrix!

Going back to our example:
$$Cov[Y, Y] = \mathbb{E}[(Y - \mathbb{E}[Y])(Y - \mathbb{E}[Y])^T] = \mathbb{E}[(AX + b - A\mathbb{E}[X] - b)(AX + b - A\mathbb{E}[X] - b)^T] =$$
$$= A\mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^T]A^T = ACov[X, X]A^T \in \mathbb{R}^{n \times n}$$

Putting everything together we have: $Y \sim \mathcal{N}(A\mu + b, A\Sigma A^T)$.

How to generate a gaussian random vector of given mean (μ_{new}) and covariance (Σ_{new})?

For the sake of simplicity let's assume we start from a standard gaussian random vector $Z \sim \mathcal{N}(\mathbf{0}, I)$, where **0** is a vector of dimension n and $I \in \mathbb{R}^{\mathtt{m},\mathtt{m}}$.
If we apply the affine transformation to Z we get a gaussian with the following distribution $Y_{new} \sim \mathcal{N}(b, AIA^T)$ (where of course $AIA^T = AA^T$).

Once again we only need to impose the following equations to hold in order to find the proper affine transformation:
$\mu_{new} = b$ and $\Sigma_{new} = AA^T$. The first is trivial while the second one might be a little more complicated (we are asking to find a matrix A such that the moltiplication by its transpose gives exactly Σ_{new}).
Luckily for us we can consider A as the "square root" of Σ and find the proper factorization using very simple linear algebra technique: the spectral theorem (eigen values decomposition).

In the case of a real symmetric $n \times n$ matrix $\Sigma = \Sigma^T$ we all know we can write the eigendecomposition in the following way: $\Sigma = Q\Lambda Q^T$ where $Q \in \mathbb{R}^{n \times n}$ can be chosen orthonormal (and real) and Λ is real.
But we know Σ has another property: it is positive semidefinite! Which means that its eigenvalues

are grater or equals to zero ($\Lambda >= 0$).

These are all the ingredients we need to solve: $\Sigma = AA^T$. In fact if we write the eigenvalues decomposition in an equivalent way we can find the proper factorization:
$\Sigma = Q\Lambda Q^T = Q\sqrt{\Lambda}\sqrt{\Lambda}Q^T$, note $\sqrt{\Lambda}$ is nothing by the diagonal matrix containing the squares of the Λ matrix (we can take the square root only thanks to the fact $\Lambda >= 0$).

If we now define $A = Q\sqrt{\Lambda}$ we get $\Sigma = Q\sqrt{\Lambda}(Q\sqrt{\Lambda})^T$ and we have solved the problem.

Generalization vs training error

In this section we are going to compare the generalization error vs the training error in the most simple scenario you can imagine: scalar Least Squares using a linear model class.

Note: Since we want to make things as clear as possible we are going to be very pedantic and we are going to push much emphasis on the difference between a random variable and its realizations. We are going to use the underbar (strange right? this will be more eye catching for you) to denote a random variable. So you need to intend the following: $\underline{y}$ is a random variable (say a Gaussian, a Bernoulli, Chi-square, ...) and y is its realization, which is a number (say 1.2, 10.0, 99.032, ...).

To begin with we are going to setup up the notation and the main assumptions we need in order to perform all the necessary computations. We are going to compute integrals which might not have closed form solutions! That is why we need some assumptions.

We are going to compute the integral to get the generalization error for a simple model class: the scalar linear models.
$\mathcal{H}_{lin} := \{x \mapsto wx : w \in \mathbb{R}\}$
in the case of squared loss.

We are interested in comparing the following quantities:

$$L_{\mathcal{D}}(h) := \mathbb{E}_{\mathcal{D}}[l(\underline{y}, h(\underline{x}))] := \int l(y, h(x))p(x,y)dxdy \quad \text{This is a number}$$

$$L_{\mathcal{S}}(h) := \frac{1}{m}\sum_{i=1}^{m} l(y_i, h(x_i)) \quad \text{This is a number: a realization of the following random variable}$$

$$\underline{L_{\mathcal{S}}(h)} := \frac{1}{m}\sum_{i=1}^{m} l(\underline{y}_i, h(\underline{x}_i)) \quad \text{This is a random variable}$$

where l is a loss funtion (a deterministic function), $h \in \mathcal{H}$ and $\mathcal{D}$ is the joint distribution of $\underline{y}$ and $\underline{x}$ (outputs and inputs to our models).

In the following exercise we need to define what we mean by each ones of these quantities.

- 1 Let's start by defining what is the data generative process $\mathcal{D}$, you can think to it as it where the object Die in the previous lab (it describes the random variables which are generating all the data we see). We need to specify the joint distribution of $\underline{x}$ and $\underline{y}$: $p(x, y)$. Remember we can write the joint probability of two general random variables using conditional probabilities, therefore we have: $p(x, y) = p(y|x)p(x)$, don't forget $p(y|x)$ describes a random variable on $\underline{y}$ while x is to be considered as a fixed number (you can think of it as an outcome). For now everything is super general, let's introduce some assumption in order to make our lives easier. Let's assume $\underline{x} \sim \mathcal{N}(\mu_x, \sigma_x^2)$, and $p(y|x) \sim \mathcal{N}(ax, \sigma^2)$. Note here the mean of $\underline{y}$ is depending on the outcomes x. How to generate samples from $\underline{y}$? Take $\underline{x}$, sample it, get a number, say $x = 1.234$ and now generate a sample from a gaussian random variable whose mean is $a * 1.234$ and whose variance is σ^2.

Why did I choose such a strange $p(x, y)$? You can prove such a joint distribution is derived from the following very simple model:
$\underline{y} := a\underline{x} + \underline{e}$
where $\underline{e} \sim \mathcal{N}(0, \sigma^2)$ and $\underline{x}$ is independent from $\underline{e}$.

TODO: Do the computations and show the distribution of $\underline{y}$ is $\mathcal{N}(a\mu_x, a^2\sigma_x^2 + \sigma^2)$.

- 2 We are choosing the squared loss, i.e. $l(b, c) := (b - c)^2$.
- 3 We choose our model class to be the set of scalar linear models defined by:
  $\mathcal{H}_{lin} := \{x \mapsto wx : w \in \mathbb{R}\}$

Generalization error computation: In this very simple case we can actually compute the generalizaiton error by hand (solving the integral in the definition). If you want you can compute the integral using the densities... but I prefer to save some time and use the properties of the expected value.

$$\begin{aligned}
L_{\mathcal{D}}(h) &= \mathbb{E}_{\underline{x},\underline{y}}[(\underline{y} - h(\underline{x}))^2] = \mathbb{E}_{\underline{x},\underline{y}}[(\underline{y} - w * \underline{x})^2] \qquad (1)\\
&= \mathbb{E}_{\underline{x},\underline{y}}[\underline{y}^2] - 2w\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}\underline{y}] + w^2\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}^2] \qquad (2)\\
&= \mathbb{E}_{\underline{y}}[\underline{y}^2] - 2w\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}\underline{y}] + w^2\mathbb{E}_{\underline{x}}[\underline{x}^2] \qquad (3)
\end{aligned}$$

Now recall the basic relation $var[\underline{y}] = \mathbb{E}[\underline{y}^2] - \mathbb{E}[\underline{y}]^2$, which allows us to write $\mathbb{E}[\underline{y}^2] = var[\underline{y}] + \mathbb{E}[\underline{y}]^2$ (we will use this formula both for $\underline{x}$ and $\underline{y}$)

$$\begin{aligned}
L_{\mathcal{D}}(h) &= \mathbb{E}_{\underline{y}}[\underline{y}^2] - 2w\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}\underline{y}] + w^2\mathbb{E}_{\underline{x}}[\underline{x}^2] \qquad (4)\\
&= a^2\sigma_x^2 + \sigma^2 + a^2\mu_x^2 - 2w\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}\underline{y}] + w^2(\sigma_x^2 + \mu_x^2) \qquad (5)
\end{aligned}$$

Our last "issue" is: $\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}\underline{y}]$, replacing the definition of $\underline{y}$ and using the fact that $\underline{x}$ and $\underline{e}$ are uncorrelated we have: $\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}\underline{y}] = \mathbb{E}_{\underline{x},\underline{y}}[\underline{x}(a\underline{x} + \underline{e})] = a\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}^2]$, therefore using again the basic relation we get:

$$\begin{aligned}
L_{\mathcal{D}}(h) &= a^2\sigma_x^2 + \sigma^2 + a^2\mu_x^2 - 2w\mathbb{E}_{\underline{x},\underline{y}}[\underline{x}\underline{y}] + w^2(\sigma_x^2 + \mu_x^2) \qquad (6)\\
&= a^2\sigma_x^2 + \sigma^2 + a^2\mu_x^2 - 2w\mathbb{E}_{\underline{x}}[\underline{x}^2] + w^2(\sigma_x^2 + \mu_x^2) \qquad (7)\\
&= a^2\sigma_x^2 + \sigma^2 + a^2\mu_x^2 - 2w(\sigma_x^2 + \mu_x^2) + w^2(\sigma_x^2 + \mu_x^2) \qquad (8)\\
&= \sigma^2 + a^2(\sigma_x^2 + \mu_x^2) - 2w(\sigma_x^2 + \mu_x^2) + w^2(\sigma_x^2 + \mu_x^2) \qquad (9)\\
&= \sigma^2 + (\sigma_x^2 + \mu_x^2)(a^2 - 2w + w^2) \qquad (10)\\
&= \sigma^2 + (\sigma_x^2 + \mu_x^2)(a - w)^2 \qquad (11)
\end{aligned}$$

Boring but easy!

In the following you will implement a function to compute $L_{\mathcal{D}}(h)$ (I am using the word "computing" not approximating here, so use the formula we just derived!!!) and a simple class to represent the joint probability of $\underline{x}$ and $\underline{y}$.