

Notes for Linear Regression

Basic Remark: Regression means: predict some real value(s) given some other (usually real) values.

I'd like to make some things a bit more explicit than it is on the slides for linear regression.

To draw parallels with coin flipping: There we have observations of the form

"H T H H ...".

Now in linear regression, the observations are a bit more 'elaborate'. One observation is actually a pair (x, z) : For a given x (can be high dimensional), we observe one real number. We call this number also a *measurement*, as it nicely plays with the idea that we use a device to measure some interesting property (i.e. z) at some state (x). And we want to find out, how x and z are related to each other. Note, we are *not* interested in the x values on their own, only the relation (ie. the pair) is interesting! So we observe a sequence (or more general: a set) of pairs:

$$(x_1, z_1), (x_2, z_2), \dots, (x_n, z_n).$$

So far so good. Now, analogous to the coin: We think we have some way of generating these pairs (we *don't* try to generate the x 's!), ie. some model. This model itself has a probabilistic form, so it can tell us something about a given pair. This is illustrated with the graph on page 9: If we ask the model about possible pairs with $x = x_0$, it will tell us this: The mean value of all possible values z at x_0 is $y(x_0, w)$, a function denoted by the symbol on the actual input x_0 and some parameters w . Furthermore, the probability distribution for all possible z is a gaussian. So note again, this model does *not* care to talk about x_0 alone! It just assumes that the *input* value is given to it from somewhere. It's only interested in modeling the relation between z and x .

Now that we know how the model thinks about one pair, continue along the coin flip example: We want to talk about all observed pairs, specifically about the likelihood of these pairs, given our model:

$$p((x_1, z_1), (x_2, z_2), \dots, (x_n, z_n) | \text{my model})$$

Note how I condition on *my model*. This looks awkward, hence for the time being, I say that formally I can identify *my model* with some abstract parameters w . Good, in a way. Not so good, because, how should we go on about this huge probability expression over the pairs?

Independence assumption(s)! Those pairs have nothing to do with each other! Because the functional relationship we want to model between x and z is static. That is, for some x_p the value z_p we get from the measurement device only depends on *input* x_p ! It does not depend on e.g. if we measured briefly before on some 'location' x_q , or that this measurement resulted in z_q . Note that this independence assumption is again a very aggressive one and not necessarily true in reality. But it helps. And even better, we think all pairs behave the same and can be described by the same model. Note that these long sentences are expressed in the formula above by the conditioning on *my model*. All in all, the *i.i.d* assumption gives us:

$$p((x_1, z_1), (x_2, z_2), \dots, (x_n, z_n) | w) = \prod_i p(x_i, z_i | w)$$

Now for every pair the sum rule: $p(x_i, z_i | w) = p(z_i | x_i, w) p(x_i | w)$

Our model knows nothing about $p(x_i | w)$, so we *don't care*, i.e. leave it away! (Leaving it away means $p(x_i | w) \equiv 1$). We are already quite close to the right hand side of eq. 5 on page 10 of the slides.

Similar to the coin example, we have to plug in something specific for $p(z_i|x_i, w)$, otherwise we can't continue. Equation 4 states our assumptions in order to answer this question: The measurement z is produced by two sources: some function depending on x and w (ie. $y(x, w)$), and some unavoidable noise that happens to be zero mean gaussian (with some variance σ^2).

This means that z is itself a gaussian, with mean $y(x, w)$ and variance σ^2 (i.e. if you add some gaussian noise to a constant, you get a shifted version of the original gaussian).

Thus, $p(z_i|x_i, w) = \mathcal{N}(z_i|y(x_i, w), \sigma^2)$ where \mathcal{N} denotes *Gaussian pdf*. Plug in our specific assumption(s) about $y(x, w)$ (slide 7) and you get equation 5, right side. So now that we have a *handy* way to write out our original probability (the joint distribution over all pairs together), remember why we did that: This likelihood (its real value, a number) depends on some parameters w , and we want to find those that maximize it.

Really do the couple of steps required to get from there to the final form in equation 6. Notice there that $\ln p(z|w, \beta)$ basically has the form $-E_D(w)$ (the minus sign is important! The underbrace at eq. 6 does *not* include the minus in front of the β [btw. the β is uninteresting, left out here]). We want to find a w that maximizes that (after all, we do maximum likelihood estimation), which is the same as *minimizing* the value of $E_D(w)$ (i.e. the minus sign thrown away!). Still, maximizing/minimizing is all about finding extrema, so the procedure is as usual.

You should be able to write down the result in eq. 8, starting from $E_D(w)$ as given in equation 6. As a hint, consider the scalar product $a^T a$ for some N -dimensional vector a : it is a sum over squared values, the entries of a . Now use this information and work backwards from the sum form in $E_D(w)$. This is an important trick! Writing things down as pure Vector/Matrix operations opens up the door for all possible Linear Algebra tricks/methods/rules.

Please, spend also some time thinking specifically about the design matrix. It depends only on the x 's! E.g. What form would it have if our model is the one on page 3? How many rows/columns for those 10 observed blue dots? What about the model on page 4? How does the *first* column of the design matrix *always* look like, according to our slides?

Btw. a comment on the idea of basis functions and design matrix. Consider a two dimensional x i.e. (x_1, x_2) , and polynomial basis functions, up to degree 2. What would this specifically mean? Well, for some x we would compute x_1, x_2, x_1^2, x_2^2 and $x_1 x_2$. So the design matrix will have 6 (don't forget the bias!) columns.

What if we use some other basis functions. E.g, we use *three* gaussians. Note, that these are supposed to be used as basis functions, so they are not representing any probability density (ie. they are unnormalized). One of these gaussians will have the form $\exp(-0.5(x - p)^2/\sigma^2)$, p specifies where this gaussian is centered, and σ how wide it is. These two parameters are chosen *before* we do any learning, so they are not adaptable through eg. MLE. So then you plug in some x into those three gaussians. If x happens to be one of the centers that you specified, than this gaussian will result in a 1 for this x . So here the designmatrix will have 4 entries (columns) for every observed data point.

A remark on semi positive definite. For checking on the type of extrema value we always look at the second derivative. This is simple for one dimensional functions. For multi variate functions this involves a square form (the second derivative is in general a matrix), in our case

$$x^T \Phi^T \Phi x$$

Do the calculation for the second derivative on your own! One gets a minimum if the second derivative is positive definite. If we set $b = \Phi x$ we see that this square form is simply $b^T b$, the squared length of b . Unless b is not the zero vector, it will always have length greater than 0. Nice! The problem here

now is that Φx can be 0, i.e. the matrix Φ can transform some of the x to the zero vector. (This is related to the next point, rank of a matrix.) So, strict positive definiteness is in general not ensured (but we assume so!).

A remark on the rank of a matrix. Let's assume some matrix A , having N rows and d columns, with $N > d$. The rank in general means the maximum number of linearly independent rows/columns (same number for rows and columns!). So A having full rank means that it will have rank d . Now look at $A^T A$. This is a square matrix of form $d \times d$. Annoyingly, even if A has full rank, it is not necessarily that $A^T A$ has full rank. But for sure, if A has *not* full rank, then $A^T A$ has rank smaller than d because the rank of the product of any two matrices is smaller than or equal to the minimum rank of the two matrices. So if $\text{rank}(A) \leq d - 1$ then $\text{rank}(A^T A) \leq d - 1$ (note: $\text{rank}(A) = \text{rank}(A^T)$).

A remark on *ill conditioned*. In general matrices encode linear transformations. For square matrices this is usually described through saying that a matrix does rotations and/or shearing. See the link to the SVD tutorial for a generalization to any rectangular matrix. "Ill-conditioned" hereby means then (informally!) that one 'aspect' of this transformation dominates all the others, e.g. everything is stretched (relatively) heavily in one particular direction. This means, in particular, that if you have two points that are very close to each other transformed by such a matrix, they are 'thrown' apart rather heavily. Mathematically, this is no big thing! But when implementing things on a computer, we only have finite precision to encode stuff. So we inevitably do 'wrong' calculations, e.g. numerical errors accumulate along our computation. Such an ill conditioned transformation then can exaggerate these numerical errors, possibly in such a way that these errors entirely dominate the result. Even more so, as the transformation itself must be encoded on the computer and introduces error itself by being applied.
