MLPR Tutorial Sheet 1 (Answers)

You should attempt the tutorial questions before your tutorial.

I strongly recommend you discuss these questions—and the course in general—with your peers. You could try to meet up with people from your tutorial group (their email addresses are listed in the tutorial group assignments), or other people you have met in lectures.

As a student, I’d attempt tutorial questions by myself at first, but usually couldn’t do absolutely everything in a sensible amount of time. If you can’t do a part, skip it at first and move on! I’d then meet up with a friend from the class, and we’d pool our understanding. Finally we’d ask someone else, or ask questions in a tutorial, about the parts that neither of us could manage.

Your tutorial session won’t necessarily discuss every part. The point of tutorials isn’t to give you the answers: the answers will be made available after the week’s tutorials, and can be discussed further on Hypothesis. The tutorial sessions are mainly useful for giving you practice at explaining your thinking, and discussing particular points that the group might find helpful.

1. Linear Regression and linear transformations:

Alice fits a function \( f(x) = w^\top x \) to a training set of \( N \) datapoints \( \{x^{(n)}, y^{(n)}\} \) by least squares. The inputs \( x \) are \( D \)-dimensional column vectors.

Bob has heard that by transforming the inputs \( x \) with a vector-valued function \( \phi \), he can fit an alternative function, \( g(x) = v^\top \phi(x) \), with the same fitting code. He decides to use a linear transformation \( \phi(x) = Ax \), where \( A \) is an invertible matrix.

a) Show that Bob’s procedure will fit the same function as Alice’s original procedure.

NB You don’t have to do any extensive mathematical manipulation. You also don’t need a mathematical expression for the least squares weights.

Answer:
By fitting \( g(x) = v^\top Ax \), Bob can only set the function equal to linear combinations of the inputs \( (v^\top A)x = w^\top x \), where \( w = A^\top v \). Moreover, just like Alice, all linear combinations are available: any function Alice fits can be matched by setting \( v = A^{-\top}w \).

As both Alice and Bob are selecting the function that best matches the outputs from the same set of functions, and with the same cost function, they will select the same function.

b) Could Bob’s procedure be better than Alice’s if the matrix \( A \) is not invertible?

[If you need a hint, it may help to remind yourself of the discussion involving invertible matrices in the pre-test answers.]

Answer:
First we assume \( A \) is square and not invertible, (or rectangular with rank less than \( D \)), then multiple input points are transformed to the same vector: \( Ax = Ax' \).

It’s not possible for Bob to assign different function values to two such inputs, whereas Alice can. Bob can no longer fit all the same functions as Alice. Bob is still restricted to linear combinations of the features, but might not be able to
choose the linear combination of $x$ that minimizes the least-squares cost.

As a result, Bob’s training error might be worse than Alice’s, but can’t be better. If Alice’s fit is well-justified, Bob will also generalize more poorly due to underfitting. However, if $x$ is high-dimensional, or $N$ is small, Bob’s restricted regression model might avoid overfitting and could generalize better.

(For completeness: If $A$ is $K \times D$ with $K > D$, and has rank $D$, then no information is lost. The matrix is not invertible, but Bob will always get the same fit as Alice. There is not a unique way to set the $K$ parameters $v$ to represent this fit.)

c) Alice becomes worried about overfitting, adds a regularizer $\lambda w^\top w$ to the least-squares error function, and refits the model. Assuming $A$ is invertible, can Bob choose a regularizer so that he will still always obtain the same function as Alice?

**Answer:**
For any model fit $w$ that Alice considers, Bob can set $v = A^{-\top} w$ and represent the same function. To penalize every function in the same way as Alice, Bob could substitute $w = A^\top v$ into the penalty:

$$\lambda w^\top w = \lambda v^\top A A^\top v.$$ 

If we allow this form as a regularizer, then Bob can fit an equivalent model.

If we insist on a regularizer of the form $\lambda v^\top v$, then we’d need $A$ to be an orthogonal matrix where by definition, $AA^\top$ is the identity matrix. Otherwise I don’t see any way to ensure the optimal function is the same.

d) **Bonus part:** *Only do this part this week if you have time. Otherwise review it later.*

Suppose we wish to find the vector $v$ that minimizes the function

$$(y - \Phi v)^\top (y - \Phi v) + v^\top M v.$$ 

i) Show that $v^\top M v = v^\top (\frac{1}{2} M + \frac{1}{2} M^\top) v$, and hence that we can assume without loss of generality that $M$ is symmetric.

ii) Why would we usually choose $M$ to be positive semi-definite in a regularizer, meaning that $a^\top Ma \geq 0$ for all vectors $a$?

iii) Assume we can find a factorization $M = AA^\top$. Can we minimize the function above using a standard routine that can minimize $(z - X w)^\top (z - X w)$ with respect to $w$?

**Answer:**

i) This is standard linear algebra manipulation (see background material). Expand out:

$$v^\top (\frac{1}{2} M + \frac{1}{2} M^\top) v = \frac{1}{2} v^\top M v + \frac{1}{2} v^\top M^\top v$$

The second term is a scalar, so we can take the transpose of it, and then
\[
\begin{align*}
\text{reorder:} \\
= \frac{1}{2} v^\top M v + \frac{1}{2} (v^\top M^\top v)^\top = \frac{1}{2} v^\top M v + \frac{1}{2} v^\top M v = v^\top M v.
\end{align*}
\]

As \((\frac{1}{2} M + \frac{1}{2} M^\top)\) is symmetric, and has the same effect as \(M\) in the cost function, we can always choose to specify the cost function with a symmetric matrix.

ii) Usually the purpose of the regularizer is to prevent large weights \(v\). If there is a direction \(a\) where \(a^\top M a < 0\) then we can make the second term as negative as we like by making \(v\) a large multiple of \(a\). Unless the term with \(\Phi\) and \(y\) prevents it, the minimum cost could be at infinite \(v\). The whole point of the regularizer is so we don’t need the data to constrain the parameters however. If we want to guarantee containing the parameters, we should have a strictly positive definite matrix.

iii) For information, we can always obtain the factorization \(M = AA^\top\). Because we can assume the matrix \(M\) is symmetric, the eigendecomposition of the matrix\(^1\) can be manipulated into this form. If the matrix is also positive definite, which it probably should be (why?\(^2\)), we can use a Cholesky decomposition.

We can then use the same data-augmentation trick as in the lecture notes. Add fictitious observations to the data:

\[
y' = \begin{bmatrix} y \\ 0 \end{bmatrix}, \quad \Phi' = \begin{bmatrix} \Phi \\ A^\top \end{bmatrix},
\]

and then set \(v\) to minimize \((y' - \Phi'v)^\top (y' - \Phi'v)\).

You might have given a different answer based on linearly transforming the data so that the regularizer takes the standard form.

**Intended lessons from Q1:**

You should know to immediately recognize that a sequence of linear transformations can be achieved with a single linear transformation. If we want a function to do anything else, we need to introduce non-linearities. However, linear transformations are an important building block, so you need to be able to manipulate and understand them.

Linear transformations, even simple ones such as scaling individual features, do change the effect of a regularization term. Therefore you may need to think about the scale of your data before setting a regularizer (or a grid of \(\lambda\) to explore). Alternatively always rescale your data so the numbers are zero mean and have unit variance. Then default choices for regularizers might work.

2. **Logistic Sigmoids:**

i) Sketch — with pen and paper — a contour plot of the sigmoidal function

\[
\phi(x) = \sigma(v^\top x + b),
\]

---


The square-root of the central diagonal matrix can be absorbed into the matrices either side.

2. If \(M\) is only positive semi-definite, then there are directions in which the weight vector can grow arbitrarily big without being penalized.

---

for \( v = [1 \ 2]^\top \) and \( b = 5 \), where \( \sigma(a) = 1/(1+\exp(-a)) \).

Indicate the precise location of the \( \phi = 0.5 \) contour on your sketch, and give at least a rough indication of some other contours. Also mark the vector \( v \) on your diagram, and indicate how its direction is related to the contours.

**Hints:** What happens to \( \phi \) as \( x \) moves orthogonal (perpendicular) to \( v \)? What happens to \( \phi \) as \( x \) moves parallel to \( v \)? To draw the \( \phi = 0.5 \) contour, it may help to identify special places where it is easy to show that \( \phi = 0.5 \).

**Answer:**

The contours are straight lines, parallel to each other and orthogonal (perpendicular) to \( v \). They are symmetrically spaced around the \( \phi = 0.5 \) contour, and bunched closer together near that contour.

Any two points where \( v^\top x + b = 0 \) can be used to identify the \( \phi = 0.5 \) contour.

**ii) If \( x \) and \( v \) were three-dimensional, what would the contours of \( \phi \) look like, and how would they relate to \( v \)? (A sketch is not expected.)**

**Answer:**

The contours of \( \phi \) are now the planes orthogonal (perpendicular) to \( v \). These planes are parallel to each other. As before, the function changes most quickly in the direction \( v \), and when we are close to the \( \phi = 0.5 \) contour plane.

In general, for any number of dimensions, the function is constant within the hyper-planes orthogonal to \( v \).

**Intended lessons from Q2:**

There will be several important explanatory figures in the course, some of them two-dimensional contour plots. So you need to understand contour plots, and not everyone has seen them. The logistic sigmoid function also comes up a lot in machine learning.

**3. Radial Basis Functions (RBFs):**

In this question we form a linear regression model for one-dimensional inputs: \( f(x) = \)
where \( \phi(x; h) \) evaluates the input at 101 basis functions. The basis functions
\[
\phi_k(x) = e^{-(x-c_k)^2/h^2}
\]
share a common user-specified bandwidth \( h \), while the positions of the centers are set
to make the basis functions overlap: \( c_k = (k - 51)h/\sqrt{2} \), with \( k = 1 \ldots 101 \). The free
parameters of the model are the bandwidth \( h \) and weights \( w \).

The model is used to fit a dataset with \( N = 70 \) observations each with inputs \( x \in [-1, +1] \). Assume each of the observations has outputs \( y \in [-1, +1] \) also. The model
is fitted for any particular \( h \) by transforming the inputs using that bandwidth into a
feature matrix \( \Phi \), then minimizing the regularized least squares cost:
\[
C = (y - \Phi w)^\top (y - \Phi w) + 0.1 w^\top w.
\]

a) Explain why many of the weights will be close to zero when \( h = 0.2 \), and why
even more weights will probably be close to zero when \( h = 1.0 \).

**Answer:**
When \( h = 0.2 \) many of the basis functions are several bandwidths away from any
of the data’s input locations. These basis functions won’t have any noticeable
effect on the function values for \( x \in [-1, +1] \) unless they have huge weights,
which is vetoed by the regularization term. As the settings of the weights for
these basis function has almost no effect on the first term in the cost function,
they will be set close to zero to minimize the second term.

As the bandwidth increases, fewer of the centers are within a few bandwidths of
\( x \in [-1, +1] \). More basis functions are given near zero weight.

[A sketch may help.]

b) It is suggested that we could choose \( h \) by fitting \( w \) for each \( h \) in a grid of values,
and pick the \( h \) which led to a fit with the smallest cost \( C \). Explain whether this
suggestion is a good idea, or whether you would modify the procedure.

**Answer:**
Different settings of \( h \) give models that have effectively different numbers of basis
functions. While each model has 101 basis functions, many of them will have no
effect due to their position and the regularization. It is likely that models with
small \( h \), which bring in many active basis functions, will obtain the smallest cost.
However, these may not generalize well. Although there is some regularization,
making the basis functions narrow, and having more basis functions than data
points (\( N = 70 \)), is likely to lead to overfitting.

After fitting the weights for each \( h \) on a training set, each model should be
evaluated on a validation set to pick the best model.

**c)** Another data set with inputs \( x \in [-1, +1] \) arrives, but now you notice that all
of the observed outputs are larger, \( y \in [1000, 1010] \). What problem would we
encounter if we performed linear regression as above to this data? How could
this problem be fixed?

**Answer:**
The outputs can only be matched with large function values, which require large
weights. The regularization term may cause the fitted function to pass entirely beneath the observed data. We could turn off the regularization, but then we might fit wildly oscillating functions.

One solution is to simply rescale the observed $y$ values to lie within $[-1, +1]$ as before. The predictions would need to be scaled back to the original range.

Another solution is to work with the data at its native scale, but model the offset with a bias term. If we add a new basis function $\phi_{102}(x) = 1$, the value of the corresponding weight is likely to be around $w_{102} \approx 1005$. We should not include this parameter in the regularization, or it won’t be able to take on such a large value. (Alternatively we could choose $\phi_{102}(x)$ to be a larger constant.) Given this solution, the range of the outputs is 10 whereas it was 2 before, so if the regularization was sensible before, it should probably be decreased (by a factor of $5^2 = 25$), or re-estimated on a validation set.

**Intended lessons from Q3:**

We can’t fit the complexity of a model by minimizing training error, as the most flexible functions always get the lowest training error. We can use cross-validation, although that can be expensive. It is also not necessarily obvious which parameters control model complexity. The Bayesian methods that are covered later in the course are one alternative.

Once again we see that the scale of the data affects what regularization is sensible. Thinking about how to transform the regularization is fiddly, which is why transforming data to lie on a standard scale is so common.