Notes on Least Squares and Nearest Neighbors

compiled from Elements of Statistical Learning by D. Gueorguiev, Oct 25, 2024

# Linear Models and Least Squares

The linear model is defined as:

Given a vector of inputs the output is predicted as:

(1)

where is the *intercept* or *bias*. Often, we include a unit constant ‘variable’ additionally in X and include the bias in the vector of the coefficients . Then we can rewrite (1) as :

(2)

where denotes a row vector or matrix transpose ( is a column vector).

Here we are modeling a single output so is a scalar; in general can be a -vector, in which case would be a matrix of coefficients. In the – dimensional input-output space, represents a hyperplane. If the constant is included in , then the hyperplane includes the origin and is a subspace; If not, it is an affine set cutting the -axis at the point . From now we assume that the intercept is included in .

Obviously, is a linear function over the -dimensional space and the gradient is a vector in input space that points in the steepest uphill direction.

Fitting the linear model to a set of training data

Popular method is least squares- we pick the coefficients to minimize the *residual sum of squares (RSS)* as:

(3)

is a quadratic function of the parameters, and hence its minimum always exists, but may not be unique. The solution can be written in matrix notation as:

(4)

where is an matrix with each row an input vector, and is an -vector of the outputs in the training set. Differentiating w.r.t. we get the *normal equations*:

(5)

If is non-singular, then the unique solution is given by:

(6)

and the fitted value at the -th input is .

## Insight in the Optimality of the Linear Regression with an Example

Let us look into a linear model in a classification context. The Figure 1 below shows a scatterplot of the training data.

The data is simulated. The output class variable has the values Blue or Orange and is represented as such on the scatterplot. There are 100 points in each of these two classes. The linear regression model was fit to these data, with the response coded as 0 for Blue and 1 for Orange. The fitted values are converted to a fitted class variable according to the rule

(7)

A graph of a linear regression

Description automatically generated with medium confidence

Figure 1: A classification example in two dimensions. The classes are coded as a binary variable (Blue = 0, Orange = 1), and then fit by linear regression.

The set of points in classified as Orange corresponds to depicted on the Figure 1 above. The two predicted classes are separated by the *decision boundary* , which is linear in this case. We notice that that there are misclassifications on both sides of the decision boundary.

Scenario 1: The training data in each class is generated from bivariate Gaussian distributions with uncorrelated components and different means.

Scenario 2: The training data in each class is generated from a mixture of 10 low-variance Gaussian distributions, with individual means themselves distributed as Gaussian.

*Note*: a mixture of Gaussians is usually described in terms of generative models. First, we generate a discrete variable that determines which of the component Gaussian to use, and then generate an observation from the chosen density.

With Scenario 1 it can be shown that a linear decision boundary is the best one can do, and that the Linear estimate is almost optimal. In the case of tightly clustered Gaussians (Scenario 2) the linear decision boundary is unlikely to be optimal. The optimal decision boundary is heavily nonlinear and disjoint and as such will be much more difficult to obtain.

# Nearest-Neighbor Methods

Nearest-neighbor methods use those observations in the training set closest in input space to to form . Specifically, the -nearest neighbor fit for is defined as follows:

(8)

where is the neighborhood of defined by the closest points in the training sample. Closeness implies a metric, which for the moment we assume is Euclidean distance.

In Figure 2 below, we use the same training data as in Figure 1, and use 15-nearest-neighbor averaging of the binary coded response as method of fitting. Assigning class Orange to if amounts to a majority vote in the neighborhood. The colored regions indicate all those points in input space classified as Blue or Orange by evaluating the procedure on a fine grid in input space.

A map of a neighborhood

Description automatically generated with medium confidence

Figure 2: The same classification example in two dimensions as in Figure 1. The classes are coded as a binary

Variable (Blue = 9, Orange = 1) and then fit by 15-nearest-neighbor averaging as in (8). Hence, the predicted class is chosen by majority vote amongst the 15-nearest neighbors.

Figure 3 below shows the results for 1-nearest-neighbor classification: is assigned the value of the closest point to in the training data. In this case the regions of classification correspond to a *Voronoi tessellation* of the training data. Each point has an associated tile bounding the region for which it is the closest input point. For all points in the tile, . The decision boundary is even more irregular than before.

A map of a city

Description automatically generated with medium confidence

Figure 3: The same classification example as in Figures 1 and 2. The classes are coded as a binary variable (Blue = 0, Orange = 1) and then predicted by 1-nearest-neighbor classification.

If we compare Figure 1 with Figure 2 we notice that the 15-nearest neighbor classifier misclassifies less training observations compared to the linear regression. From Figure 3 we notice that the -nearest neighbor algorithm does not misclassify any of the training data points.

Thus we conjecture that the error on the training data for the -nearest neighbor classifiers should be increasing function of and will be for .

It appears that the -nearest neighbor fits a single parameter (the number of neighbors) compared to the parameters in the case of linear regression. However, what is important is the *effective* number of parameters which for the -nearest neighbors is which generally is bigger than and decreases with increasing . Intuitively, this is true because if the neighborhoods were nonoverlapping , there would be unique neighborhoods and we would fit one parameter (a mean) in each neighborhood.

**Definition** *stable fit*

The linear decision boundary from the least squares linear regression is stable fit – that is, small (infinitesimal) variations in the training data points will lead to small (infinitesimal) variations in the boundary line location. The stable fit is rendered through the rigidity of the decision boundary which is constraint to be a straight line. This classifier will fit with **high bias** (due to rigidity) and **low variance** (due to stability)

On contrary, the k-nearest-neighbor do not rely on stringent assumptions about the decision boundary which would limit the minimization of the training error. However, the contour of the decision boundary with the k-nearest-neighbor is very sensitive on the specific positions of the input points and is inherently unstable. This classifier will fit with **low bias** (due to flexibility of the decision boundary) and **high variance** (due to sensitivity to the positions of the input data points).

We already discussed Scenario 1 and Scenario 2 earlier and mentioned that linear regression is more appropriate for Scenario 1 while the nearest neighbor classifiers are more suitable for Scenario 2.

//TODO: formulate and prove a theorem capturing the infinitesimal behavior of each of these models using appropriate delta-epsilon definition. Ideally, the proof should be for the general dimensional problem.

**Note**: The data for the Figures 1,2, and 3 is simulated by model similar to both Scenario 1 and Scenario 2 but closer to the latter:

1) means are generated from a bivariate Gaussian distribution and labeled all data points from this class with BLUE. Similarly, more means are drawn from and the corresponding class is labeled ORANGE.

2) For each class there are generated observations as follows: for each observation we pick an at random with probability , and then generate a , thus leading to a mixture of Gaussian clusters for each class.

Figure 4 below shows the results of classifying 10,000 new observations generated from the model. We compare the results for least squares regression and those for -nearest neighbors for a range of values of .

A graph with numbers and lines

Description automatically generated

Figure 4: Misclassification curves for the simulation example used in Figure 1, 2, and 3.

A single training sample of size 200 was used, and a text sample of size 10,000. The results for

For linear regression are the bigger orange and blue squares at three degrees of freedom.

The purple line is the optimal Bayes error rate.

**Note 2**: A large subset of the most popular classification techniques today are variants of these two simple procedures. The following list describes enhancement of these two simple procedures:

* Kernel methods ([2], [3]) use weights that decrease smoothly to zero with distance from the target point, rather than the effective 0/1 weights used by -nearest neighbors.
* In high dimensional spaces the *distance kernels methods* ([4]) are modified kernel methods which use *kernel functions* computing the distances between the data points without explicitly computing the coordinates in that space. This is achieved via the so-called *kernel trick* ([5]) where only the products between the mapped data points are needed, making the computation significantly faster.
* Local regression fits linear models by locally weighted least squares, rather than fitting constants locally
* Linear models fit to a basis expansion of the original inputs allow arbitrary complex models
* Projection pursuit and neural network models consist of sums of non-linearly transformed linear models

# A Bit of Statistical Decision Theory

Let denote a real valued random vector, and a real valued random output variable, with joint distribution . We seek a function for predicting given values of the input . This approach requires a *loss function* for penalizing errors in prediction. The most common loss function is the *squared error loss* . This leads us to a criterion for choosing ,

(9)

We condition on which allows us to write:

(10)

We rewrite (10) as :

Substituting (10) in (9) yields

# References

[1] [The Elements of Statistical Learning; Data Mining, Inference, Prediction, Trevor Hastie, Robert Tibshirani, Jerome Friedman, Second Edition, 2017](https://github.com/dimitarpg13/statistical_learning_and_kernel_methods/blob/main/literature/books/EelementsOfStatisticalLearning_print12.pdf)

[2] [Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Bernhard Schoelkopf, Alexander J. Smola, MIT, 2002](https://github.com/dimitarpg13/statistical_learning_and_kernel_methods/blob/main/literature/books/scholkopf2002learning_with_kernels.pdf)

[3] Kernel Methods for Pattern Analysis, John Shawe-Taylor, Nello Cristiannini, 2004

[4] [Gentle Introduction to the Kernel Distance, JR Phillips et al, 2011](https://github.com/dimitarpg13/statistical_learning_and_kernel_methods/blob/main/literature/articles/kernel_methods/A_Gentle_Introduction_to_the_Kernel_Distance_Phillips_2011.pdf)

[5] [The Kernel Trick for Distances, B. Schoelkopf, Microsoft Research, NIPS 2000](https://github.com/dimitarpg13/statistical_learning_and_kernel_methods/blob/main/literature/articles/kernel_methods/NIPS-2000-the-kernel-trick-for-distances-Schoelkopf.pdf)

[6] [Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels, S. Jayasumana et al, 2015](https://github.com/dimitarpg13/statistical_learning_and_kernel_methods/blob/main/literature/articles/kernel_methods/Kernel_Methods_on_Riemannian_Manifolds_with_Gaussian_RBF_Kernels_Jayasumana_2015.pdf)

# Appendix

## Affine Space

### Informal discussion

Affine space is what is left from vector space after one has forgotten which point is the origin.

Imagine that there are two observers – observer *A* and observer *B*. Observer *A* knows the real origin, but Observer *B* believes that another point is the origin. Two vectors , and , are to be added. Observer *B* believes he has computed the distance but observer A knows better – B has actually computed .

Similarly, observers *A* and *B* may evaluate any linear combination of and or of any finite set of vectors and will generally get different answers. However, if the sum of the coefficients is unit then the observers will arrive at the same answer. Moreover, for any finite number of observers if the sum of the coefficients is unity each of them will come to the same answer.

Origin according to observer B

Origin according to observer A

*according to origin A*

*according to origin B*

according to origin A

*according to origin A*

Figure A.1: origins and vector computations from the perspectives of observer A (red) and observer B (blue)

In the example on Figure A.1 if observer *A* travels to and observer *B* travels using the same route then observer *A* will observe from his frame of reference the following path of observer *B* :

So under the condition all observers who have different frames of reference will describe the same point with the same linear combination despite the different origins. While only observer *A* on the earlier example knows the *linear structure* both observers *A* and *B* know the *affine structure* that is, the values of affine combinations defined as linear combinations in which the sum of the coefficients is unity.

**Definition** *Affine Space*

A set together with a vector space , and a transitive and free action of the additive group of on the set . The elements of the affine space are called points. The vector space is said to be associated with the affine space and its elements are called *translations,* or *free vectors*.

The action in affine space will be denoted as addition and it is a mapping having the following properties:

1) *Right identity*:

, where is the zero vector in

2) *Associativity*:

, (here the last is the addition in )

3) *Free and transitive action*:

, is a bijection

The first two properties are simply defining properties of a (right) group action. The third property characterizes free and transitive actions, and onto character coming from transitivity, and then the injective character follows from the action being free. Forth property follows from 1) and 2)

4) *Existence of one-to-one translations*

the mapping is a bijection

3) is often used in the following equivalent form :

5) *Subtraction*

denoted with such that