Prediction of double gene knockout measurements

Sofia Kyriazopoulou-Panagiotopoulou
sofiakp@stanford.edu

December 12, 2008

Abstract

One way to get an insight into the potential interaction between a pair of genes is to compare the phenotype of the double knockout organism to the phenotype we would expect if the genes were independent. However, since gene knockouts are expensive and time-consuming, we would like to be able to predict gene interactions. We used a dataset consisting of 352 yeast genes for which only a few double knockout measurements were known, and applied regression algorithms to predict the unknown measurements. In this article, we describe the algorithms we used and compare their results.

1 Introduction

One way to get an insight into the interaction of two genes is to knock them out and compare the phenotype of the double knockout organism to the phenotypes of the two single knockout organisms. This can be done by comparing the measurement of a reporter protein in the double knockout organism to what we would expect if the genes were independent, based on their single knockout measurements. For instance, if two genes are in the same pathway, then knocking out both may have a less severe phenotype than expected.

We consider the case where only a small subset (the query genes) of the genes we want to study have been knocked out in pairs with all the other genes, and for the rest of the genes (the array genes) we only have their double knockout measurements against the genes of the first group. We would therefore like to predict the unknown measurements.

For our experiments we used a dataset of 352 yeast genes. There are 96 query genes measured against almost all the genes of the dataset and 256 array genes measured only against query genes. Our goal is to predict the double knockout measurements of the array genes against other array genes. Since the expected double knockout measurements are known for all pairs, predicting the difference between observed and expected double knockout measurements is equivalent to predicting the true double knockout measurements. The data we used for predictions is in the form of a symmetric 352×352 matrix D. $D(i, j)$ is the difference between the true double knockout measurement of genes i and j and

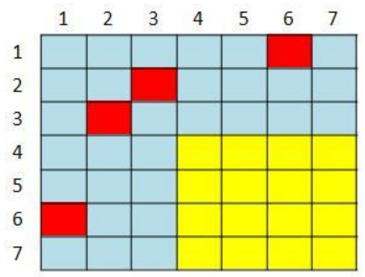

Figure 1: A simplified version of our problem. Genes 1 to 3 are query genes, and genes 4 to 7 are array genes. The entries in light blue are known. These are the differences between the observed and the expected double knockout measurements for the corresponding pairs of genes. All the other values are missing. We are interested in predicting the values in yellow.

their expected double knockout measurement. If there is no double knockout measurement available for the pair of genes (i, j), then $D(i, j)$ is treated as a missing value.

Figure 1 presents a simplified version of the problem. In this example genes 1 to 3 are query genes and genes 4 to 7 are array genes. The light blue entries are the differences between the observed and the expected double knockout measurements for the corresponding pairs of genes. These are the values we know. All the other entries are unknown. Notice that query genes are measured against most but not all the other genes: There might be a few missing values even for query genes. These may have been caused by errors in the experiments and are shown in red. Of course we could try to predict these values too (and in fact we do), but we are mostly interested in predicting the values shown in yellow, i.e. the measurements of the array genes against other array genes.

In section 2 we describe the cross validation scheme we used for evaluating our predictions. In section 3 we explain how we can make relatively "crude" predictions by clustering. Later, in sections 4 and 5 we see how these predictions can be refined using regularized linear regression and Gaussian Processes respectively. Finally, in section 6 we conclude and discuss the directions of future work.

2 Evaluating our predictions

We used a 10-fold cross validation scheme to evaluate our predictions. At each fold we selected a subset of the query genes and treated them as array genes, i.e. we treated their measurements against array genes as missing. Then we computed the root mean squared error (RMSE) of our predictions for these genes.

We assume that most of the pairs of genes are independent, so in most cases the difference between the true and expected double knockout measurement should be close to zero. Therefore, our baseline algorithm always predicts zero. The baseline RMSE is 0.3297.

3 Solution 1: k-means clustering

If we treat each row (or equivalently each column) of D as a training example with 352 features, then we can apply k-means to cluster the genes. Of course, the values of some of the features are missing for some examples. There are several ways to handle these missing values. For example, we could replace them with zero, based on our assumption that most pairs are independent, or with the row or column average. However, it turned out that the best approach in terms of performance was to completely ignore the missing values, i.e. to compute the distance between a gene and a centroid taking into account only the common non-missing features.

After clustering, we can use the values of the cluster centroids to predict the missing values of the genes in the cluster. Remember that D is symmetric, so to predict $D(i, j)$ we can use either the cluster of gene i or the cluster of gene j. To see why this is true, assume that we want to predict $D(4, 5)$ in the example of Figure 1. We could find the cluster of gene 4 and take the value of the 5^{th} feature of the centroid of that cluster. This would be the average measurement of the genes of the cluster against gene 5. Alternatively, we could find the cluster of gene 5 and take the value of the 4^{th} feature of the centroid of that cluster. This would be the average measurement of the genes of the cluster against gene 4. For that reason, our prediction for $D(i, j)$ is the average of the value of the i^{th} feature of the centroid of the cluster of gene j and the j^{th} feature of the centroid of the cluster of gene i.

When running k-means, we initialized the centroids 5 times and returned the assignment to clusters with the smallest sum of within-cluster distances. The 10-fold cross validation experiment was repeated 87 times. The mean of the RMSEs we obtained was 0.2564 with a standard deviation of 0.0048.

4 Solution 2: Regularized linear regression

Consider the simplified example of Figure 1 and assume we want to predict the unknown measurements for gene 7, i.e. we want to predict the last 4 values of column 7. We can use the first three rows of D, for which (almost) all the values are known, as the training data for a linear regression model. Our dataset consists therefore of three training examples with 6 features each. The corresponding outputs are the three first values of column 7. Using the parameters learned by linear regression, we can predict the values of column 7 that correspond to the sets of features of the last 4 rows of D.

However, not all the features of the last 4 rows are known. For that reason, we first apply k-means and replace missing values with the corresponding predictions. This implies that we follow a two-step prediction procedure, where we first find some "crude" predictions using k-means and then refine them using regression.

Note that the number of features used for linear regression is much larger than the number of training examples. Going back to the original problem, there are 96 query genes which will be our training examples, each having 351 features. As a result, linear regression doesn't work, unless we use some form of regularization, such as L2 and L1.

L2 regularized linear regression minimizes $\|X\theta - \vec{y}\|^2 + k\|\theta\|^2$, where X is the matrix of the training data with one training example per row, θ is the column vector of the parameters of linear regression, $\vec{y}$ is the column vector of the target values, and k is a parameter of the model. The closed form solution for θ is then $(X^T X + kI)^{-1} X^T \vec{y}$, where I is the identity matrix.

L1 regularized linear regression minimizes $\|X\theta - \vec{y}\|^2 + k\|\theta\|_1$. There is no closed form solution to this minimization problem, so we used an existing MATLAB implementation [3].

For both L2 and L1 regularization, the parameter k was chosen by comparing different values using the cross-validation scheme described in section 2. Since we apply k-means before clustering, there is a random component in our predictions, so again we ran the cross-validation several times and computed the mean and standard deviation of the RMSEs obtained. The mean RMSE for L2 regression was 0.2268 with a standard deviation of 0.0019, while for L1 regression the mean was 0.2235 and the standard deviation 0.0029.

5 Solution 3: Gaussian Processes

We briefly explain how Gaussian Processes can be applied in the case we study using the example of Figure 1 and following the same naming conventions as in [2].

Assume that we want to predict the missing values of gene 7 and that $y^{(i)} = f(x^{(i)}) + \epsilon^{(i)}$, $i = 1, 2, \ldots, 3$, where $\{(x^{(i)}, y^{(i)})\}$ are our training examples, as in the previous section, and $\epsilon^{(i)} \sim N(0, \sigma^2)$ are i.i.d. "noise" random variables. Assume also that X is the 3×6 matrix of training examples as explained before, X_* is the 4×6 matrix of test examples (with missing values replaced as in the previous section), $\vec{y}$ is the vector of known outputs for the examples of X, and $\vec{y}_*$ are the target values we want to predict. Then, assuming a zero-mean prior over functions $f(\cdot) \sim GP(0, k(\cdot, \cdot))$, where k is a kernel function, it can be shown ([1], [2]) that $\vec{y}_* | \vec{y}, X, X_* \sim N(\mu^*, \Sigma^*)$, where

$$\mu^* = K(X_*, X)(K(X, X) + \sigma^2 I)^{-1} \vec{y}$$

$$\Sigma^* = K(X_*, X_*) + \sigma^2 I - K(X_*, X)(K(X, X) + \sigma^2 I)^{-1} K(X, X_*)$$

$K(X, X) \in \mathbf{R}^{3\times 3}$, such that $(K(X, X))_{ij} = (k(x^{(i)}, x^{(j)}))$, and $K(X, X_*)$, $K(X_*, X)$, $K(X_*, X_*)$ are defined similarly.

We used a linear kernel $k(x, x') = x \cdot x'$, and chose the noise variance by cross validation. The mean RMSE for $\sigma^2 = 10$, found after 42 repeats of the experiment, was 0.2292, with a standard deviation of 0.0023. It is worth noting that we experimented with other kernels too. The predictions of the squared exponential kernel were always very close to zero, so its performance was almost the same as the baseline performance, and the polynomial kernels of order higher than 1 performed worse than the linear.

6 Conclusions and future work

Table 1 summarizes the results of the previous sections.

	k-means	L2 regression	L1 regression	GP
mean RMSE	0.2564	0.2268	0.2235	0.2292
standard deviation	0.0048	0.0019	0.0029	0.0023
iterations	87	87	30	42

Table 1: Mean and standard deviation of the RMSEs for the algorithms of the previous sections. The last row contains the number of iterations that were used to obtain the statistics in each case. The baseline RMSE is 0.3297.

Although L1 linear regression has the best performance, we believe that Gaussian Processes are a promising approach due to the flexibility introduced by the kernel function. Evaluating more kernels is an important direction of future work. We are currently working on using L1 linear regression to learn a Bayesian network of the genes and to create a covariance matrix that could be used as a kernel. Additionally, we are studying ways of incorporating the single knockout measurements into our predictions.

Acknowledgements

I am deeply grateful to Ph.D. student Alexis Battle for her invaluable guidance and advice throughout this project.

References

[1] C. E. Rasmussen, C. Williams: Gaussian Processes for Machine Learning, MIT Press, 2006, `http://www.gaussianprocess.org/`

[2] Chuong B. Do, Honglak Lee: Gaussian Processes, 2008

[3] P. Carbonetto: MATLAB implementation for L1-regression, `http://www.cs.ubc.ca/~pcarbo/`

[4] U. N. Lerner: Hybrid Bayesian Networks for Reasoning about Complex Systems, Ph.D. Thesis, 2002