Epigenetic imputation using Exchangeable Convolutional Neural Networks and Hidden Markov Tensor Decomposition

Jeffrey P. Spence[1,*,†], Sanjit Singh Batra[2,*], Jonathan Fischer[2], Yun S. Song[2,3,4]

August 12, 2019

1 Graduate Group in Computational Biology, University of California, Berkeley, CA 94720
2 Computer Science Division, University of California, Berkeley, CA 94720
3 Department of Statistics, University of California, Berkeley, CA 94720
4 Chan Zuckerberg Biohub, San Francisco, CA 94158
∗ These authors contributed equally to this work
† Current address: Department of Genetics, Stanford University, Stanford, CA 94305

Release Statement: We agree to make our submission publicly available as part of the challenge archive.

Abstract

Imputing the results of unperformed epigenetic experiments is an area of active research. We introduce two new epigenetic imputation models, one based on exchangeable convolutional neural networks [1, 2, 3] and the other on a novel hidden Markov model approach to tensor decomposition. For the purposes of the ENCODE Imputation Challenge, we ensemble our predictions with those from a baseline imputation method, as well as Avocado [4] using an additional neural network.

1 Introduction

1.1 Problem Statement

All cells within a multicellular organism have the same genetic sequence up to a minuscule number of somatic mutations. Yet, a menagerie of cell types exist with diverse morphologies and functions. In order to understand how such differences arise and are maintained, a considerable number of experiments to assay aspects of the epigenome (e.g., DNA modifications such as CpG methylation, histone positions and modifications, etc...) have been developed. Two large consortia, ENCODE [5] and the NIH Roadmap Epigenomics consortium [6] have either performed an extensive number of assays in a small number of cell types (ENCODE) or a small number of assays across many cell types (Roadmap). Unfortunately, due to the large number of distinct cell types and epigenetic assays it is infeasible to perform every possible experiment, and as a result, computational prediction of the results of unperformed experiments is an attractive alternative [7, 8, 4].

Mathematically, this problem may be framed as a missing data problem. Let $X_{i,j,\ell}$ be the result (e.g., $-\log_{10} p$-value) of assay j in cell type i at locus ℓ. We may view this as an $I \times J \times L$ tensor,

where I is the number of distinct cell types, J is the number of distinct assays, and L is the number of loci. Due to the experimental design, we only observe some tubes of the tensor, which is to say that there exists some set of experiments $\Omega \subset [I] \times [J]$ such that for $(i, j) \in \Omega$, we observe all L values $X_{i,j,:}$ and for $(i, j) \notin \Omega$, all L values $X_{i,j,:}$ are unobserved. The goal then is to impute predicted values for these entries, $\hat{X}_{i,j,:}, \forall (i, j) \notin \Omega$ such that $\hat{X}_{i,j,:}$ is as close as possible by some metric to the true, unobserved values $X_{i,j,:}$.

Such imputation problems are considerably ill-posed: without making assumptions about the underlying data or data-generating process it is impossible to impute these unobserved entries. The key, then, is to create models flexible enough to impute potentially highly non-linear functions, while being constrained enough to keep the problem identifiable. We enforce two different sets of assumptions in our methods outlined below. One method relies on assuming that cell types are exchangeable. The other method relies on making linearity assumptions similar to a previous method [8]. Both methods explicitly enforce smoothness along the locus mode of the tensor, corresponding to the observation that adjacent loci likely have similar epigenetic values. This smoothness assumption contrasts with previous work [7, 8, 4].

1.2 Motivation and Intuition

We devised two distinct methods to approach the problem of imputing epigenetic marks genome-wide using observations from multiple assays and multiple cell types. The first is an exchangeable convolutional neural network (ExCNN) and the second is a form of tensor decomposition framed as a hidden Markov model (HMM).

The ExCNN is a deep convolutional neural network that uses both the epigenetic signal and the DNA sequence data. The convolutions exploit the inherent spatial structure of the data while the introduction of sequence ties different regions together and leverages information that may be encoded in specific motifs, GC content, or other genomic features. These modeling choices reflect the intuition that nearby assay values are informative for predicting one another and that sequence context may impact protein binding and chromatin structure.

A naive implementation of a neural network immediately runs into challenges. It has long been known that neural networks are universal approximators [9], which implies that if the network is sufficiently large it can approximate essentially any smooth function to a given degree of accuracy. In our setting, this result is problematic: if the network can learn essentially any mapping from the observed entries of X to the unobserved entries, then in the absence of further constraints the network may impute arbitrarily (but such networks may still implicitly regularize their learned mappings [10, 11]). To circumvent these issues of identifiability, the ExCNN shares information across cell types via its architecture, enforcing permutation equivariance along the cell type mode (discussed in more detail in Section 2.2). This treats the cell types as being exchangeable, which is a very weak assumption allowing the inferred mappings to be extremely flexible, while avoiding the aforementioned identifiability problem encountered by fully flexible networks. The exchangeability assumption explicitly encodes the intuition that if two cell types have the same values for the same set of assays, then they should have the same values for unperformed assays. Furthermore, enforcing permutation equivariance has been shown to greatly improve both the rate of training and the performance of neural networks in other settings [3].

The hidden Markov tensor decomposition (HMTD) is based on the assumption that nearby loci are likely to have similar epigenetic profiles, and that at a given locus, say ℓ, many cell types and assays will be similar, which is to say that $X_{:,:,\ell}$ should be an approximately low rank matrix. Specifically, HMTD assumes that each locus is characterized by a discrete latent variable conditional on which the observed data is a low rank matrix. Intuitively, the different values of the latent

variable correspond to different combinations of epigenetic states among the cell types (e.g., active promoter in neural-related cells). These latent variables are assumed to evolve along a chromosome according to a Markov chain. Imputation is then performed by computing the posterior mean over the hidden states. This offers a twist on the usual tensor decomposition framework by introducing discretized states and enforcing smoothness along the locus mode. This discretization supplies a robust regularization to signal components, while averaging over the posterior allows us to model uncertainty and interpolate between low-rank components. This method is motivated by the fact that different assays and cell types are likely to be related to one another, and multilinear models provide a robust and straightforward manner by which to model these associations [8]. Furthermore, we expect epigenetic signal to depend on the behavior of neighboring regions, and HMMs are often used to handle this fact in other genomic contexts.

As mentioned, both methods intentionally model the spatial aspect of the data. The ExCNN does so through convolutions while the HMTD relies upon the hidden state transition matrices. Locally, this explicit modeling of the spatial dimension improves predictions by introducing additional shrinkage for nearby regions. Both methods offer additional strategies to approach spatial structure than the multi-scale perspective of Avocado [4]. Each method also provides a unified imputation framework which is less sensitive to missingness patterns than ChromImpute, which relies on a nearest-neighbors-like search to determine which other cell types and assays to use to impute a given experiment [7].

2 Methods

2.1 Data pre-processing

For both methods, the original data ($-\log_{10} p$-values) were averaged in non-overlapping 25 bp windows. The data were then transformed by mapping $X_{i,j,\ell} \mapsto \log_{10}(X_{i,j,\ell} + 1)$ for all $(i, j) \in \Omega$ and all ℓ. These steps helped reduce the effect of outliers, reduce variance, and decrease memory and run-time requirements. Hyperparameter optimization was performed using the training/validation split, while all available data (both training and validation) were used to train our final models and impute the blinded data.

2.2 Exchangeable convolutional neural network

The architecture for our ExCNN is shown in Figure 1. Briefly, we consider a window composed of 100 bins each of 25 bp. We begin with a one-hot encoding of the DNA sequence for this region (i.e., a 4×2500 matrix). We then perform 16 same-padded 1D convolutions with patch width 4 and 64 filters, resulting in a 64×2500 matrix. We then apply a valid-padded 1D convolution with patch width 25 and stride 25 with 64 filters resulting in a 64×100 matrix. This matrix is then transposed and tiled to result in a $51 \times 100 \times 64$ tensor, which is concatenated to the epigenomic data which we transpose into a $51 \times 100 \times 35$ tensor (cell-type by locus by assay; there are 51 cell types and 35 assay types), resulting in a $51 \times 100 \times 99$ tensor. We encode missing data as -1. Up to this point, the network essentially acts as a sequence extractor, and then appends the extracted sequence features to the epigenetic assays for each cell type for each position. Since the same sequence information is appended to each cell type the network is, thus far, permutation equivariant. To continue to enforce this invariance we then stack 8 permutation-equivariant layers. Each permutation-equivariant layer consists of a permutation-equivariant part and a permutation-invariant part. The equivariant part is constructed by performing a valid-padded 2d convolution with patch size 1×4 with 64 filters – because the patch size is 1 along the cell type mode, this convolution only ever considers information

from a single cell type at a time, maintaining permutation equivariance. The permutation-invariant part is constructed by performing a valid-padded 2d convolution, again with patch size 1×4 and 64 filters, but this is then collapsed along the cell type mode by taking the max along the cell type dimension. This collapsed tensor is then tiled, and the output of the permutation-equivariant and permutation-invariant parts are concatenated, resulting in a $51 \times w \times 128$, where $w \leq 100$ because of the valid padding. Finally, we perform a valid-padded 2d convolution with linear activations with 35 filters and patch size $1 \times w'$, where w' is the length of the tensor output by the penultimate layer. This results in a matrix of shape 51×35 corresponding to the imputed assay $\times$ cell type matrix at the central locus. Throughout our network, all convolutions are followed by ReLU activations except where noted. We implemented our model in Keras [12] using a TensorFlow backend [13]. Architecture hyperparameters were tuned by hand using the training/validation split.

The ExCNN was trained using the observed epigenetic tracks (preprocessed as described in Section 2.1) and raw sequence data (one-hot encoded). We trained the model using the `adam` optimizer [14]. For each training data point, we choose ℓ uniformly at random and then for each $i, j \in [I] \times [J]$ we would set the entire tube $X_{i,j,\ell:(\ell+100)}$ to be missing in the input, to mimic the test-time scenario of trying to impute unknown values. Let this masked version of the input be denoted $\tilde{X}$, and denote the above neural network with weights w by Φ_w. Recall that the output of our neural network, $\Phi_w(X)$ is a cell type by assay matrix Our loss function for a given datapoint is then

$$||\Phi_w(\tilde{X}_{:,:,\ell:(\ell+100)}) - X_{:,:,(\ell+50)}||_F^2$$

where $||\cdot||_F$ is the Frobenius norm, and so we essentially minimize squared error.

Because our model consists solely of convolutions we were able to speed up training and testing by sharing computations across adjacent loci. We thus used data points of 1000 bins rather than 100 bins and used the following loss function:

$$\sum_{\ell'=0}^{900} ||\Phi_w(\tilde{X}_{:,:,(\ell+\ell'):(\ell+\ell'+100)}) - X_{:,:,(\ell+\ell'+50)}||_F^2,$$

allowing us to see more data at a lower computation cost. We used minibatches of size 4, and trained on $25,000$ minibatches, which corresponds to the network seeing about 2.5Gb of data. Training was performed on a single Nvidia TITAN X (Pascal) GPU, and took less than six hours. Imputation was performed using the same hardware and took less than **(FIXME: how long? also, double check training details)**.

2.3 HMM Tensor Decomposition

The HMTD is framed as an HMM with K hidden states and is parameterized by a set of cell type loadings $\mathbf{u}_1, \ldots, \mathbf{u}_N \in \mathbb{R}^{51}$, a set of assay loadings $\mathbf{v}_1, \ldots, \mathbf{v}_M \in \mathbb{R}^{35}$, variance terms $\sigma^2 \in \mathbb{R}_{++}^{51 \times 35}$, hidden state weights $\phi \in \mathbb{R}^{N \times M \times K}$, transition matrix $\mathbf{Q} \in \mathbb{R}^{K \times K}$, and initial probabilities $\eta \in \mathbf{\Delta}^{K-1}$. Let $Z_\ell \in [K]$ denote the hidden state at locus ℓ. The generative model of the data is then

$$\begin{aligned} Z_1 &\sim \text{Categorical}(\eta) \\ Z_{\ell+1}|Z_\ell &\sim \text{Categorical}(\mathbf{Q}_{Z_\ell,:}) \\ X_{i,j,\ell}|Z_\ell &\sim \mathcal{N}\left(\sum_{n=1}^{N}\sum_{m=1}^{M} u_{i,n}v_{j,m}\phi_{n,m,Z_\ell}, \sigma_{i,j}^2\right). \end{aligned}$$

In essence, for a given hidden state Z_ℓ, the observed matrix of epigenetic values at that locus is $\sum_{n=1}^{N}\sum_{m=1}^{M} \phi_{n,m,Z_\ell}\mathbf{u}_n\mathbf{v}_m^T$ with independent, mean zero noise added to each entry of the matrix.

Thus, under this model, if N and M are small, the HMTD generates a sequence of low rank matrices along the genome.

In order to learn the parameters of the HMTD $(\mathbf{u}_1, \ldots, \mathbf{v}_1, \ldots, \sigma^2, \phi, \mathbf{Q}, \eta)$, which we will collectively denote by Θ, we would like to appeal to standard algorithms for HMMs [15], but unfortunately such methods do not scale to such large datasets and cannot handle continuous observations. To circumvent this issue, we compute approximate stochastic gradients of the log-likelihood of the data and use the `adam` optimizer [14]. That is, we would like to be able to compute:

$$\nabla_\Theta \log \mathbb{P}(X|\Theta),$$

but this is infeasible due to the large size of X. As such, we note that by the chain rule of probability

$$\begin{aligned}\log \mathbb{P}(X|\Theta) &= \sum_{\ell=0}^{L-1} \log \mathbb{P}(X_{:,:,\ell+1}|X_{:,:,0:\ell}, \Theta) \\ &\approx \sum_{\ell=0}^{L-1} \log \mathbb{P}(X_{:,:,\ell+1}|X_{:,:,(\ell-\delta):\ell}, \Theta)\end{aligned}$$

where the approximation is somewhat justified by the Markov property of the hidden states. Intuitively, the hidden state at a given locus should not depend too much on the data at extremely distant loci. The above approximation suggests the following stochastic estimator of the gradient

$$\nabla_\Theta \log \mathbb{P}(X_{:,:,\ell+1}|X_{:,:,(\ell-\delta):\ell}, \Theta),$$

with ℓ chosen uniformly at random. In practice, we consider the likelihood of multiple adjacent sites, using the following stochastic gradient

$$\nabla_\Theta \log \mathbb{P}(X_{:,:,(\ell+1):(\ell+\epsilon)}|X_{:,:,(\ell-\delta):\ell}, \Theta),$$

with $\delta = 100$ and $\epsilon = 500$. These gradients are computed using automatic differentiation of the standard HMM forward algorithm [15], which we implemented in TensorFlow [13]. We trained our HMM using data from across the whole genome (as preprocessed in Section 2.1) for 2500 minibatches using minibatches of size 8, corresponding to seeing about 250 Mb of data.

We performed a grid search to find suitable hyperparameters, choosing $N = M = 5$ and $K = 100$.

To impute we perform the Forward-Backward algorithm [15] to obtain a marginal posterior over the K hidden states at each locus. We then impute using the posterior mean:

$$\hat{X}_{i,j,\ell} = \sum_{k=1}^{K} \mathbb{P}(Z_\ell = k|X) \sum_{n=1}^{N} \sum_{m=1}^{M} \mathbf{u}_{n,i} \mathbf{v}_{m,j} \phi_{n,m,k}.$$

In order to reduce memory usage, we divided the genome up into chunks of size 250 kb, which overlapped by 25 kb, and ran the Forward-Backward algorithm independently in each chunk.

2.4 Ensembling

Ensembling of the four predictive models–ExCNN, HMTD, Avocado, and baseline–was performed to improve overall performance. We implemented a feed-forward, two-layer neural network with 32

units in the first hidden layer and 16 in the second. The input was the imputed values of the four methods at a given locus for a given track, and it was trained on MSE between the output and the true value of that track at that locus (as preprocessed in Section 2.1). The same network was used for all tracks, and was trained by **(FIXME: minibatch size, etc...)**, choosing tracks uniformly at random.

3 Discussion

We presented two novel methods for performing tensor imputation. One, the ExCNN makes a simple, flexible exchangeability assumption and can learn highly non-linear imputations while taking sequence information into account. The other, the HMTD is a more parametric generative modeling approach. Overall, we found that the ExCNN generally outperformed Avocado, and both outperformed the HMTD (see Figure 2). Yet, the parameters of the HMTD are more readily interpretable (e.g., cell-type and assay loadings for the different hidden states and the rate of transitioning between different hidden states) and may be of biological interest. Visualizing and interpreting these parameters is a future research direction. Meanwhile, the ExCNN makes explicit use of the DNA sequence, and it would be of interest to see what signals it learns, and if these signals correspond to known motifs.

We also found great utility in ensembling the various methods (see Figure 2), with the ensembled prediction generally outperforming any of the constituent methods.

References

[1] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc., 2017.

[2] Siamak Ravanbakhsh, Jeff Schneider, and Barnabás Póczos. Equivariance through parameter-sharing. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2892–2901. JMLR.org, 2017.

[3] Jeffrey Chan, Valerio Perrone, Jeffrey Spence, Paul Jenkins, Sara Mathieson, and Yun Song. A likelihood-free inference framework for population genetic data using exchangeable neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8594–8605. Curran Associates, Inc., 2018.

[4] Jacob Schreiber, Timothy Durham, Jeffrey Bilmes, and William Stafford Noble. Multi-scale deep tensor factorization learns a latent representation of the human epigenome. *bioRxiv*, page 364976, 01 2019.

[5] ENCODE Project Consortium. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489(7414):57–74, 09 2012.

[6] Roadmap Epigenomics Consortium et al. Integrative analysis of 111 reference human epigenomes. *Nature*, 518:317 – – 330, 02 2015.

[7] Jason Ernst and Manolis Kellis. Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues. *Nature Biotechnology*, 33:364 – 376, 02 2015.

[8] Timothy J. Durham, Maxwell W. Libbrecht, J. Jeffry Howbert, Jeff Bilmes, and William Stafford Noble. Predictd parallel epigenomics data imputation with cloud-based tensor decomposition. *Nature Communications*, 9(1):1402, 2018.

[9] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[10] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *The Fourth International Conference on Learning Representations*, 2016.

[11] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[12] François Chollet et al. Keras. `https://keras.io`, 2015.

[13] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *The Third International Conference for Learning Representations*, 2015.

[15] Lawrence R. Rabiner and Biing Hwang Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3:4–16, 1986.

4 Authors Statement

JPS, SSB, JF, and YSS conceived the study. JF explored data pre-processing. JPS and SSB implemented the Ex-CNNs and HMM TD. SSB and JF performed ensembling. JF, JPS, and SSB drafted this summary and YSS proofread it.

5 Acknowledgments

(FIXME: Insert funding)

List of Figures

Figure 1: **(FIXME: Make an architecture figure.)**

Figure 2: **(FIXME: Include a benchmarking figure with ensemble, Avocado, HMTD, ExCNN, and baseline)**