PCA and LDA were implemented using NumPy [1]. Other algorithms were functions provided directly by Python packages, namely GMM from scikit-learn [2], SVM from LIBSVM [3] and CNN from Tensorflow [4].

For both LDA and CNN, which do not reference PCA like GMM and SVM, the full training dataset was used. In particular, density of visualized points is too low for LDA with only 500 points.

As the PIE images are labelled 1 to 68, I labelled my selfies as 69. For the scatter pots, colors are repeated as there is no convenient way to obtain a color directly from a number in Matplotlib, and the maximum number of colors in the colormaps seems to be 20. Selfies are distinguished with a red star.

1. PCA based data distribution visualizations

   Let the data matrix $\mathbf{X}$ be of $n \times p$ size.
   First, perform SVD:

   $$\mathbf{X} = \mathbf{USV}^\top \tag{1}$$

   The columns of $\mathbf{XV} = \mathbf{USV}^\top\mathbf{V} = \mathbf{US}$ are the principal components and the columns of $\mathbf{V}$ are the eigenfaces.
   For the principal components, only calculation for the highest dimension is necessary. For reduction to dimension k, the first k principal components can be obtained from the first k columns of $\mathbf{U}$ and the diagonal matrix formed by the first k values along the diagonal of $\mathbf{S}$.

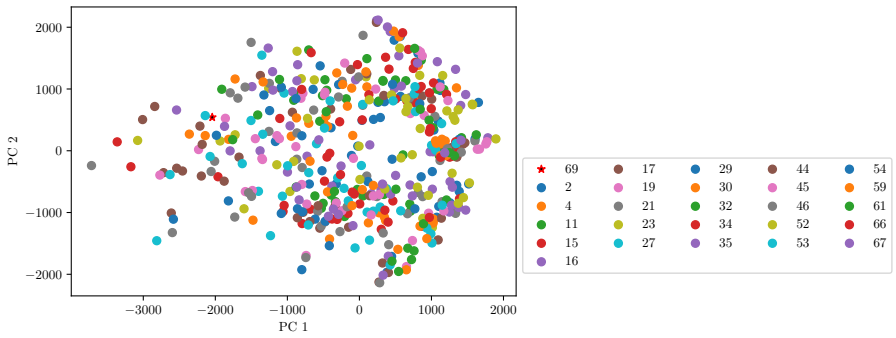


Figure 1: PCA projected data vector in 2D.

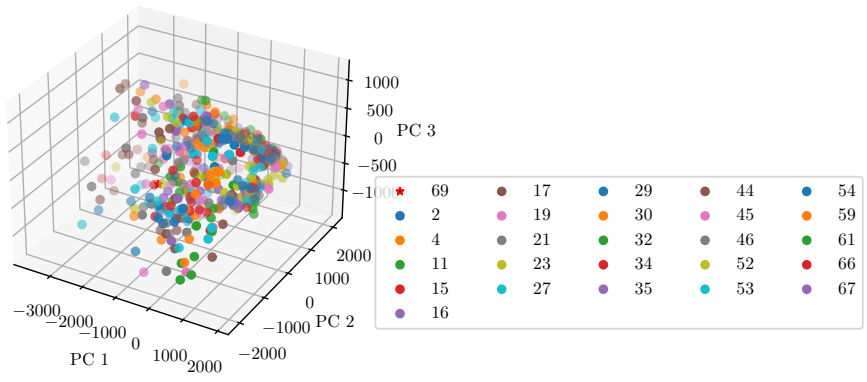
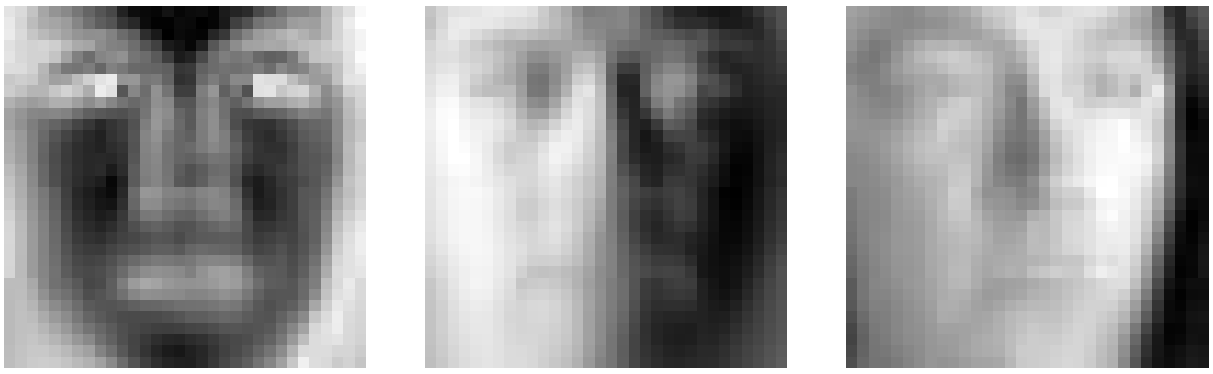


Figure 2: PCA projected data vector in 3D.



(a) eigenface 0 (b) eigenface 1 (c) eigenface 2

Figure 3: Corresponding 3 eigenfaces used for PCA dimensionality reduction.

Test Images	Dimensionality	Accuracy
CMU PIE	40	49.02 %
	80	52.47 %
	200	55.22 %
Selfies	40	0 %
	80	0 %
	200	0 %

Table 1: KNN classification accuracy for dimensionalities 40, 80 and 200 after PCA with 500 samples, for PIE images and selfies.

2. PCA plus nearest neighbor classification results

   For classification with PCA, obtain **XV**, projection of the faces in the test data matrix **X** on the eigenface space. Each column of **V** and therefore each column of **XV** corresponds to a principal component.

   It is expected that classfication of selfies is unsuccessful. There are only three selfies in the test set and random sampling of 500 images for training included only one selfie which is an outlier in 3D as can be seen from the scatter plots. Moreover, my setup and lighting condition differs from those of the images in the PIE data set. illumination is a challenge for many computer vision problems.

3. LDA based data distribution visualization

   The Fisher-LDA optimization problem is

$$\max_{w} \frac{w^T S_b w}{w^T S_w w}$$

   wehre S_b and S_w are between- and within-class scatter matrices

$$S_b = \sum_{i=1}^{N} n_i(\mu_i - \mu)(\mu_i - \mu)^T \text{ and } S_w = \sum_{i=1}^{N} \sum_{x \in \mathcal{C}_i} (x - \mu_i)(x - \mu_i)^T$$

   The solutions are the eigenvectors of $S_w^{-1} S_b$, in descending order of the corresponding eigenvalues. Perform projection as with PCA.

   Unlike PCA, which does not consider the class labels, projected points belonging to the same class are grouped together.

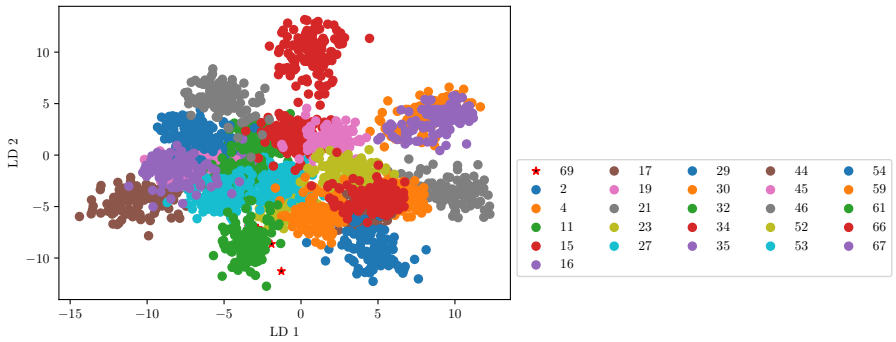


Figure 4: LDA projected data vector in 2D.

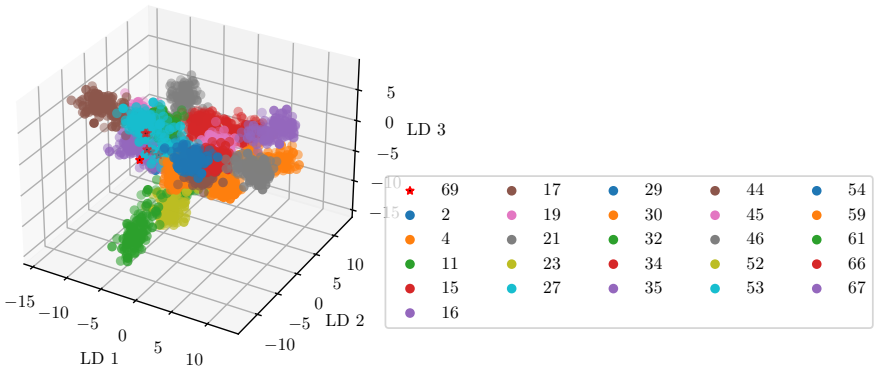


Figure 5: LDA projected data vector in 3D.

4. LDA plus nearest neighbor classification results

   For a fair comparison of PCA and LDA, PCA is rerun with the full training set.

   Clearly, LDA is able to achieve the same classification accuracy with significantly fewer dimensions. Exceeding PCA at 40 dimensions using only 9 dimensions.

   In general, PCA finds the directions of maximal variance while LDA finds a feature subspace that maximizes class separability. While PCA is suitable for feature extraction, LDA is better for classification, as it considers separately the within class and between class variances, minimizing the former while maximizing the latter.

   It is worth noting that *LDA* succeeds in identifying one out of three selfies with 9 dimensions. Given the difference between my selfies and the PIE images taken under more standardized conditions, LDA's consideration of within class variance allows it to be more successful at classiying my selfies, which would be outliers among the PIE images.

Test Images	Dimensionality	Accuracy
CMU PIE	2	47.69 %
	3	68.16 %
	9	94.20 %
Selfies	2	0 %
	3	0 %
	9	33.33 %

Table 2: KNN classification accuracy for dimensionalities 2, 3 and 9 after LDA, for PIE images and selfies.

Test Images	Dimensionality	Accuracy
CMU PIE	40	93.88 %
	80	96.00 %
	200	97.25 %
Selfies	40	0 %
	80	0 %
	200	0 %

Table 3: KNN classification accuracy for dimensionalities 40, 80 and 200 after PCA, for PIE images and selfies.

5. GMM for clustering

   For GMM, the clustering result appears to be heavily influenced by shadows or dark regions in general. Each cluster occupies a row above their associated scatter plot, with every 15th face shown, up to a limit of 10. Regardless of how or whether the input was processed, there is always one row that is of relatively uniform gray level (within each image), one row with dark region on the left, and one row with dark region on the right. The algorithm does not appear to be grouping people with similar facial features together, with a European and an Asian appearing together in every cluster.

   Consider the fact that we set the number of components to be 3. There are a lot of variations in facial features, but patches of black, regions of relatively darkness are much more uniform. Therefore, the algorithm is working exactly as intended and grouping the most similar images together, those with the same illumination.

   For clustering by facial features, we would need to set the number of components to be the number of distinct faces in the dataset. With 3 components, we are doing clustering by illumination.

   Clustering by illumination also means that PCA, which finds components reflecting the largest sources of variance works extremely well for reducing the number of dimensions. In this case, illumination is the largest source of variation.

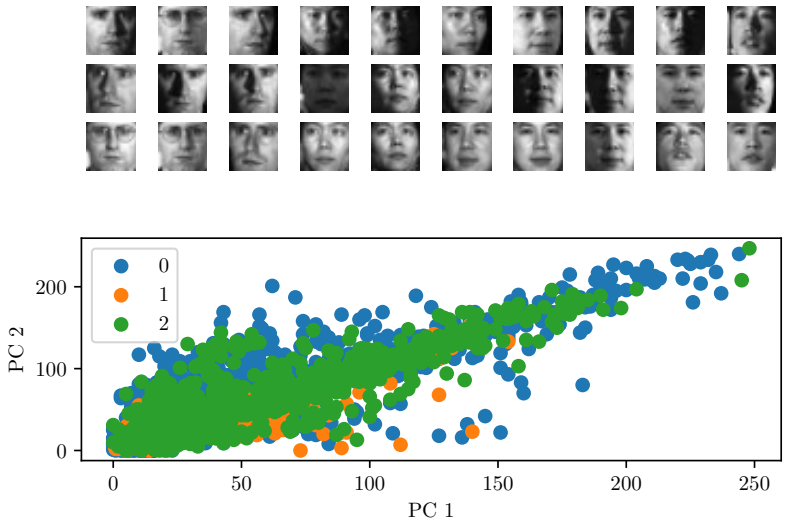


Figure 6: GMM clustering for raw face images (vectorized).

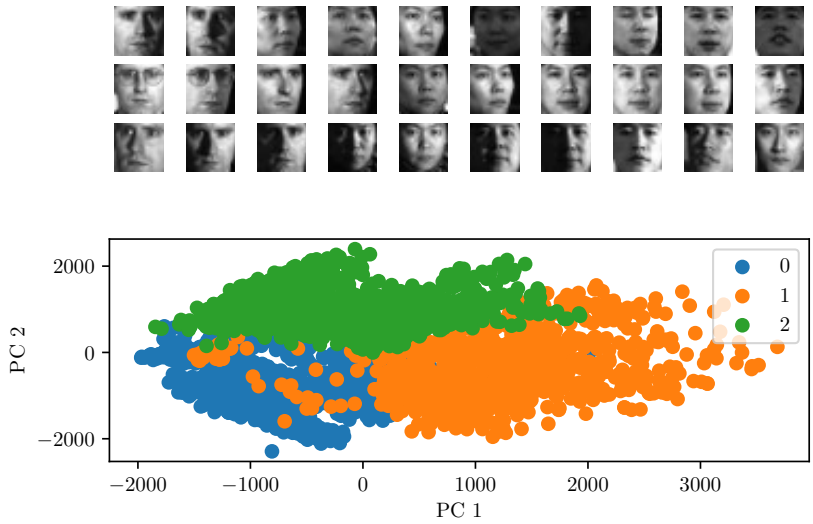


Figure 7: GMM clustering for face vectors after PCA pre-processing (with dimensionality of 200).

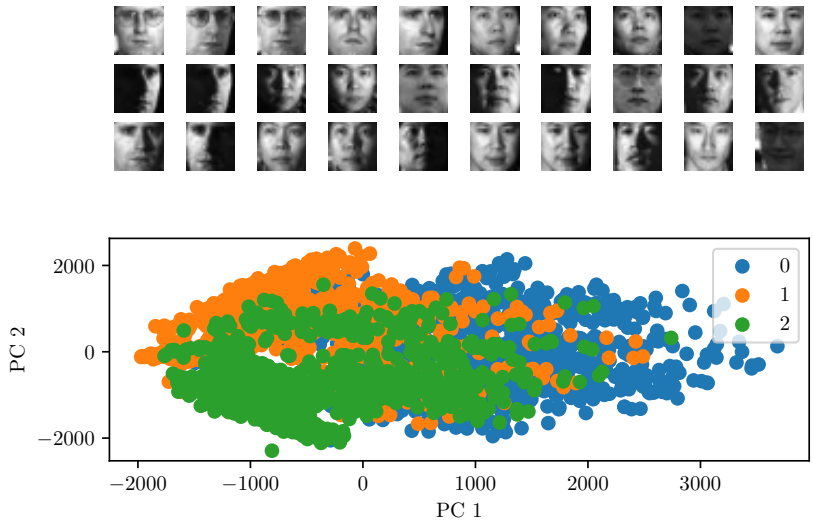


Figure 8: GMM clustering for face vectors after PCA pre-processing (with dimensionality of 80).

6. SVM classification results with different parameter values

   From Tab. 4, we can see that SVM classification achieves much higher accuracy than that of KNN in Tab. 1. More principle components improves the classification accuracy, with 200 principle components, the classfication accuracy is the same as using raw images.

   However, the parameter C appears to have no impact on the classfication accuracy. There is an explanation online for this [5]. Of the two criteria, the use of a linear SVM is stated in the instructions for this assignment.

   According to Rasmussen & Williams [6], "In a feature space of dimension N, if $N > n$ then there will always be a separating hyperplane. However this hyperplane may not give rise to good generalization performance, especially if some of the labels are incorrect."

   For PCA, 500 data points were sampled, less than 1024. The data is linearly separable, meeting the other criterion.

   Therefore, "if C values change within a reasonable range, the optimal hyperplane will just randomly shift by a small amount within the margin (the gap formed by the support vectors). Intuitively, suppose the margin on training data is small, and/or there is no test data points within the margin too, the shifting of the optimal hyperplane within the margin will not affect classification error of the test set."

Test Images	C	Accuracy
Raw	0.01	98.4351 %
	0.1	98.4351 %
	1	98.4351 %
PCA (200)	0.01	98.4351 %
	0.1	98.4351 %
	1	98.4351 %
PCA (80)	0.01	97.9656 %
	0.1	97.9656 %
	1	97.9656 %

Table 4: SVM classification accuracy for dimensionalities 40, 80 and 200 after PCA.

7. CNN classification results with different network architectures

   For CNN, the accuracy ranges from 95.00 %to 97.00 % with shuffling of the order of images.

   Following the updated instructions received via email, my model includes two convolutional layers and two fully connected layers. The updated number of nodes is "20-50-500-26". There is an additional scaling layer at the start to scale the range of each pixel from the range 0–255 to 0–1. Although often recommended, scaling does not seem to improve the classfication accuracy of this model.

   For previous parts of this assignment, labels were taken directly from the folder names in the provided PIE dataset. However, for CNN, one hot encoding is necessary because there is no natural ordinal relationship for the order of PIE subjects. For classification of two or more labels using one-hot encoded representation, categorical crossentropy is the option in Keras. The final layer uses a softmax function to convert logits to probabilities. Otherwise, ReLU is used to introduce nonlinearity.

   The latest optimizer offered by Keras, the AMSGrad variant of ADAM was used for best performance. As ADAM is an adaptive algorithm, no tuning of the learning rate was necessary or offered any improvement to classfication accuracy.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 32, 32, 1)         0

 conv2d (Conv2D)             (None, 28, 28, 20)        520

 max_pooling2d (MaxPooling2D  (None, 14, 14, 20)       0
 )

 conv2d_1 (Conv2D)           (None, 10, 10, 50)        25050

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 50)         0
 2D)

 flatten (Flatten)           (None, 1250)              0

 dense (Dense)               (None, 500)               625500

 dense_1 (Dense)             (None, 26)                13026

=================================================================
Total params: 664,096
Trainable params: 664,096
Non-trainable params: 0
_________________________________________________________________
```

Listing 1: Summary of CNN.

   For simplicity, I arbitrarily set the number epochs to 100 with batch size of 128.

   It takes about 55 epochs to reach 100 % accuracy on the training set and for the training loss to fall from an initial value of about 3.25 to 0.

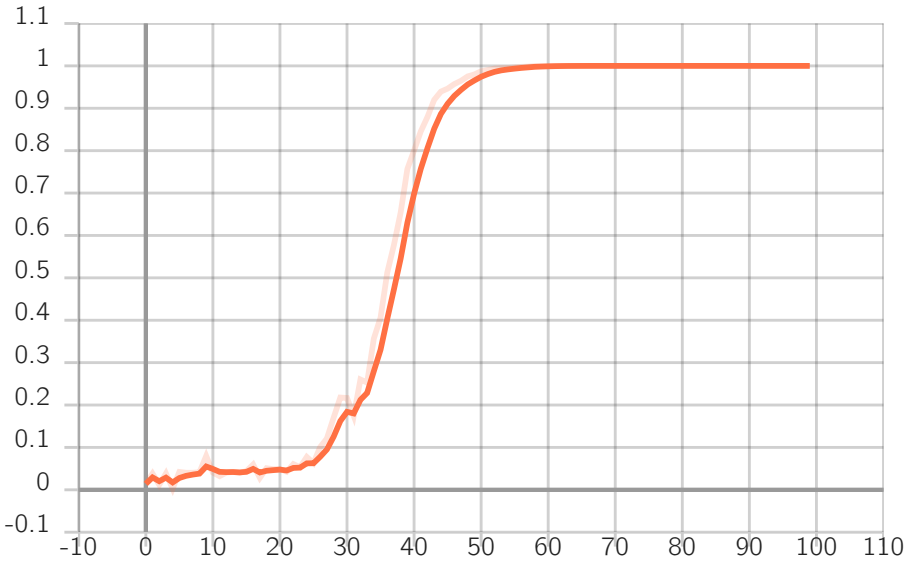


Figure 9: Epoch accuracy.

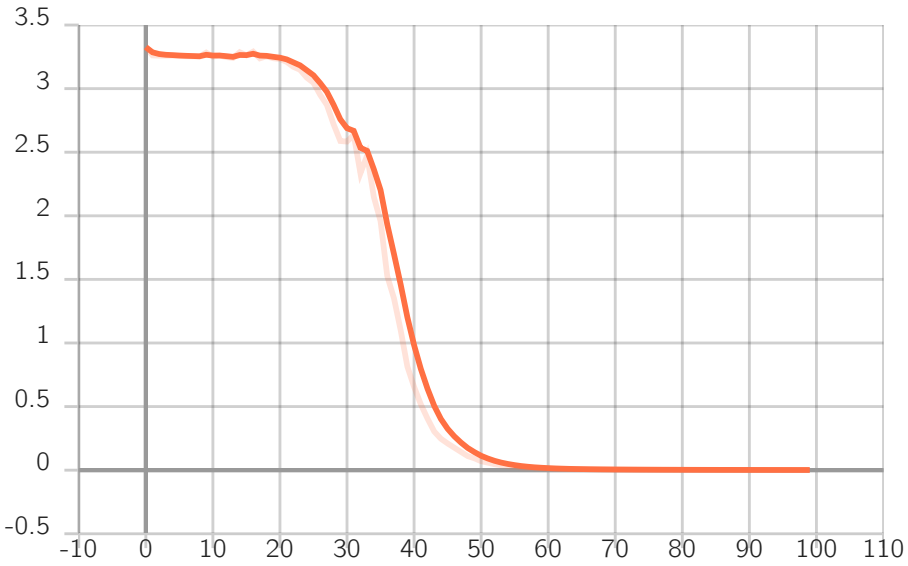


Figure 10: Epoch loss.

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, p. 357–362, 2020.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow, Large-scale machine learning on heterogeneous systems," 11 2015.

[5] luz (https://stats.stackexchange.com/users/133311/luz), "What is the influence of c in svms with linear kernel?" Cross Validated, uRL:https://stats.stackexchange.com/q/238209 (version: 2016-10-04). [Online]. Available: https://stats.stackexchange.com/q/238209

[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.