Dimensionality Reduction

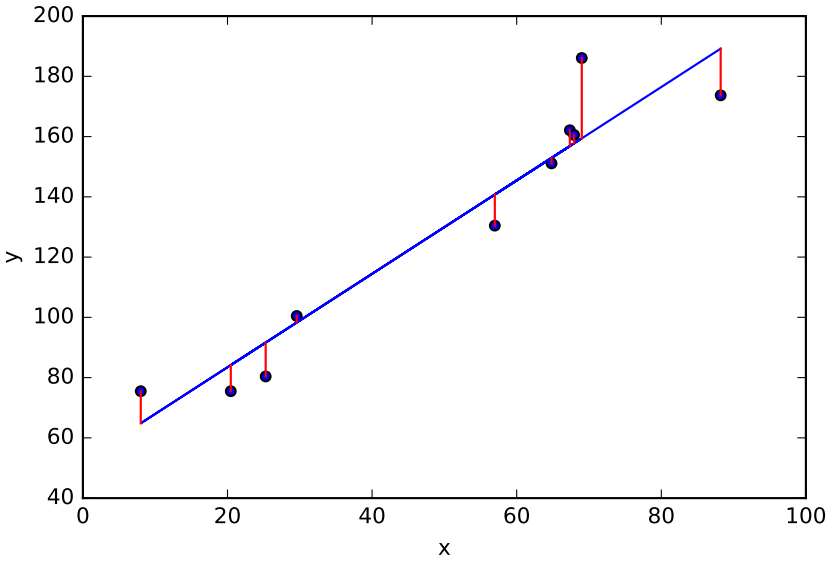
1 Introduction

Goals of Assignment Dimensionality reduction is a lossy compression technique that represents data X in a lower number of dimensions. The examples considered in this assignment are principal components analysis (PCA) and independent components analysis (ICA). You will get a high level understanding of what these methods are and apply them to the Olivetti faces data set and the MNIST digit data set. The goals of the assignment are to visualize the latent components, plot how the reconstruction error varies as a function of the number of top components, and to use the components in a larger classification scheme using logistic regression.

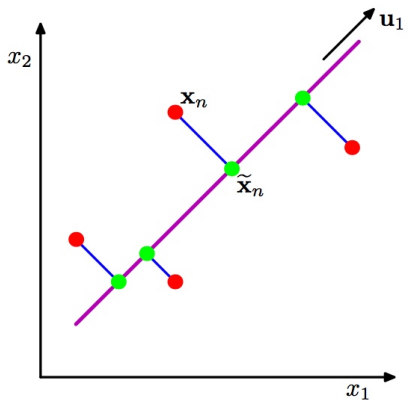
Software All the approaches you will use are implemented in the *scikit-learn* library. You can install *scikit-learn* with the command `conda install sklearn` or `pip install sklearn`, prepending the command with `sudo` if you need to.

2 Background

Principal components analysis (PCA) Recall how ordinary least squares for linear regression minimizes the sum of squared errors to the line $\mathbb{E}[Y|X] = X^\top \beta$, that is, it minimizes the sum of squared vertical distances for each observation Y_n to each prediction $X_n^\top \beta$, given independent variables $X_{1:N}$ and dependent variables $Y_{1:N}$ (see Fig. 1a). If we also wanted to include the possibility of error in the independent variables, we could instead look for a line that minimizes the sum of squared errors of data points *projected* to that line. This is what PCA does (see Fig. 1b). (Since we are now assuming statistical error in all the observed dimensions, in everything that follows, we refer to the observations as matrix X, which is the concatenation of independent and dependent variables of size N-by-D.)



(a) Ordinary least squares minimizes sum of squared errors to expected dependent variables.



(b) PCA on similar (but not identical) data as for Fig. 1a finds a line (or a hyperplane in higher dimensions) that minimizes the sum of squared errors to the projection of the data.

It can be shown that minimizing this error is equivalent to finding the top M eigenvectors of the covariance matrix of the data C. We can then form a *reconstruction* of the original data using PCA and the M vectors. We are free to chose M, knowing that if we use $M = D$ then there will be no error in the recovery.

Independent components analysis (ICA) ICA is based on similar assumptions to PCA, but assumes non-Gaussian distributed independent latent vectors.

3 Tasks

3.1 Task 1: Preparation and Fitting

- First, read the documentation for the `scikit-learn` implementations of PCA[1], ICA[2]. In particular, note how to use the `fit` function on data.
- Next, write a code cell to load the Olivetti faces data set[3] and the MNIST digits data set. The faces data can be loaded using the function `sklearn.datasets.fetch_olivetti_faces`. The MNIST digit data set can be loaded with these instructions[4]. Call the variables representing the data sets by suitable names like X_{faces} and X_{faces}.
- Fit PCA and ICA on the two data sets.

3.2 Task 2: Visualization

- Visualize all the components for the face and digit data for PCA and ICAHow would you describe the components visually (i.e., what aspects of the data is each component discovering)?
- Calculate the PCA and ICA image reconstructions of a small subset of faces and digits.

[1] `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html`

[3] `http://scikit-learn.org/stable/datasets/olivetti_faces.html`

[4] `http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html`

- Visualize your reconstructions using the same code that you used to view the discovered latent components.

3.3 Task 3: Reconstruction Error

- Let's now look at the quality of the reconstructed in terms of reconstruction error. Write a python function to find the root mean squared error (RMSE) between all the original images and their reconstructions for a given M, number of principal components.

- Plot the reconstruction error (y-axis) for PCA and ICA as a function of M. Is there a value of M that is small but for which there is (relatively) low reconstruction error? Select this value for PCA, ICA, for each data set in turn.

3.4 Task 4: Dimensionality Reduction in Classification

In this final task, let's use a *whitened* representation of the data using PCA and ICA as features for logistic regression.

- First, whiten the MNIST digit data sets using the `scikit-learn` function `whiten` (this works for both `PCA` and `ICA` objects).

- Second, using the whitened data for each of PCA and ICA, find the optimal coefficients in logistic regression for the digit label as the dependent variable.

- Third, for each of { PCA+classifiction, ICA+classification } plot the classification accuracy for the whole MNIST data set as a function of the number of components. This plot will have two lines: one for each method.

- (Fourth, optional: Repeat the classification accuracy plot when training on a randomly selected 70% of the data and testing on the remaining 30% of the data.)