Principal Component Analysis

Rein van den Boomgaard

February 28, 2017

1 Basic Idea

- The basic idea of PCA is to find a vector basis that is more appropriate for the dataset that is analyzed.

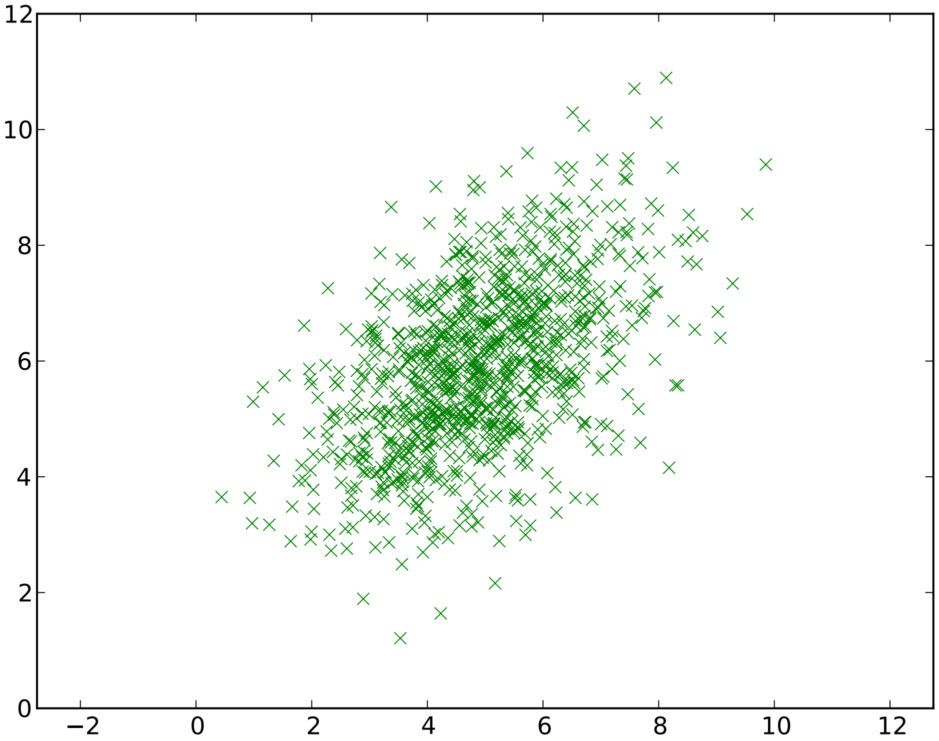

Figure 1: Clustering of data in 2D space.

- Looking at the scatter plot of the data set in figure 1 we see that the data is clustered around its mean μ and the cluster is ellipse shaped.
- The variance in the direction of the long axis of the ellipse is larger then in the perpendicular direction.
- It could even be the case that the variance in the perpendicular direction is zero: all points are on the line in the direction of the major axis.
- The axes of maximal en minimal variance are given by **principal component analysis**.

- Describing our data with respect to the new axes, we might even get rid of some dimensions (in the new axes) as they contain little variance (as in the example above) or relatively little variance.

2 Quadratic Forms

2.1 Definition

- Consider the function
$$q(\mathbf{x}) = \mathbf{x}^\mathsf{T} Q \mathbf{x}$$
where Q is a $(n \times n)$ symmetric matrix ($Q^\mathsf{T} = Q$). Such a function is called a **quadratic form**.

2.2 Level Lines

- What is the shape of the set of all $\mathbf{x}$ such that $q(\mathbf{x}) = \text{constant}$?
- Start easy with a 2 dimensional example:
$$q(x_1, x_2) = \begin{pmatrix} x_1 & x_1 \end{pmatrix} \begin{pmatrix} q_{11} & q_{12} \\ q_{12} & q_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = q_{11}x_1^2 + 2s_{12}x_1x_2 + s_{22}x_2^2$$
  - Observe that the function in x_1 and x_2 is a quadratic function, hence its name.
  - Observe that for a quadratic form with possible non symmetric matrix A the matrix $Q = \frac{1}{2}(A + A^\mathsf{T})$ leads to the same function. Therefore we may restrict quadratic forms to symmetric matrices.
- and make that even easier: $q_{12} = 0$ and $q_{11} = q_{22}$. What then is the shape of the set of points (x_1, x_2) such that $q(x_1, x_2) = 1$? We have:
$$q_{11}(x_1^2 + x_2^2) = 1$$
or equivalently:
$$x_1^2 + x_2^2 = \frac{1}{q_{11}}$$
This is evidently a circle with radius $\frac{1}{\sqrt{q_{11}}}$, but only in case $q_{11} > 0$.
- In case we leave $q_{12} = 0$ but allow different values for q_{11} and q_{22} we get:
$$\left(\frac{x_1}{\sqrt{q_{11}^{-1}}}\right)^2 + \left(\frac{x_2}{\sqrt{q_{22}^{-1}}}\right)^2 = 1$$
This is axes aligned ellipse.
- The general case turns out to be a rotated ellipse (again assuming positive coefficients), but the general case is easier proven using a bit of linear algebra.

2.3 Eigen Decomposition

- Consider the quadratic form: $q(\mathbf{x}) = \mathbf{x}^\mathsf{T} Q \mathbf{x}$, with $Q = Q^\mathsf{T}$
- For a symmetric matrix we have $Q = UDU^\mathsf{T}$ where U is the orthogonal matrix of the n (normalized) eigenvectors of Q and D is the diagonal matrix with the eigenvalues of Q.
  - Note that a symmetric matrix always has n eigenvectors that form an orthonormal basis.
  - The eigenvalues are assumed to be ordered $\lambda_1 \geq \lambda_2 \geq \cdots$.
  - The eigen decomposition $Q = UDU^\mathsf{T}$ can easily be understood starting with $Q\mathbf{u}_i = \lambda_i \mathbf{u}_i$. Collecting all eigenvectors in a matrix $U = (\mathbf{u}_1 \cdots \mathbf{u}_n)$ we can easily arrive at $Q = UDU^\mathsf{T}$.

2.4 Change of Basis

- The quadratic form $q(\mathbf{x}) = \mathbf{x}^\mathsf{T} Q \mathbf{x}$, with $Q = UDU^\mathsf{T}$ can be rewritten as:

  $$\begin{aligned} q(\mathbf{x}) &= \mathbf{x}^\mathsf{T} UDU^\mathsf{T} \mathbf{x} \\ &= (U^\mathsf{T}\mathbf{x})^\mathsf{T} D (U^\mathsf{T}\mathbf{x}) \end{aligned}$$

  In case we now take $\mathbf{y} = U^\mathsf{T}\mathbf{x}$ we are calculating the coordinates of the same vector but with respect to the basis given by the columnvectors of U. Note that U is an orthogonal matrix and thus in 2D amounts to a rotation. So we have in the new coordinate frame:

  $$q(\mathbf{y}) = \mathbf{y}^\mathsf{T} D \mathbf{y}$$

  Again a quadratic form but in fact a very simple one:

  $$q(\mathbf{y}) = \lambda_1 y_1^2 + \cdots + \lambda_n y_n^2$$

  In two dimensions $n = 2$ we recognize the ellipses (assuming the eigenvalues are positive). Thus with respect to the rotated basis the quadratic form has (hyper) ellipsoid sets where $q(\mathbf{x}) = \textit{constant}$.

3 Principal Components

3.1 Definition

- Consider a random variable $\mathbf{X}$ distribution with mean μ and covariance matrix Σ.
- We already have seen that the variance in a direction $\mathbf{r}$ (with $\|\mathbf{r}\| = 1$) is given by the quadratic form: $\operatorname{Var}(\mathbf{r}^\mathsf{T} X) = \mathbf{r}^\mathsf{T}\Sigma\mathbf{r}$.
- Using the fact that $\operatorname{Cov}(A\mathbf{X}) = A\operatorname{Cov}(\mathbf{X})A^\mathsf{T} = A\Sigma A^\mathsf{T}$ and that Σ is a symmetric matrix that can be decomposed as $\Sigma = U\Lambda U^\mathsf{T}$ where U contains the eigenvectors as columns and Λ is a diagonal matrix with the eigenvalues along the diagonal we see that

  $$\operatorname{Cov}(U^\mathsf{T}\mathbf{X}) = \Lambda$$

- The eigenvectors of U are called the principal components. And when representing the vectors with respect to this basis, i.e. $\mathbf{Y} = U^\mathsf{T}\mathbf{X}$, the covariance matrix is diagonal meaning that the elements of the vectors are uncorrelated and the variances of the elements simply add up.
- The eigenvalues collected in matrix Λ are assumed to be ordered. So most of the variance is due to the first element in vector $\mathbf{Y} = U^\mathsf{T}\mathbf{X}$. Then the second element etc.
- When we plot the eigenvalues from high to low we get the **scree diagram**. This diagram gives us a hint about how many dimensions are really important for this random (vector) variable.
  - In case $\sum_{i=1}^{k} \lambda_i > 0.99 \sum_{i=1}^{n} \lambda_i$ the first k elements of a vector expressed with respect to the eigenvector basis account for more then 99 percent of the variance.
  - In case $k << n$ we could decide that instead of dealing with vectors in $n$-dimensional space we can reduce the dimensionality of the problem by considering only k dimensions. Note that we cannot just drop $n - k$ dimensions in the original vector $\mathbf{X}$: first we have to transform it to the PCA basis and then we can drop the last $n - k$ dimensions.

3.2 PCA in Practice

- In practice we only have N samples from an \$n\$-dimensional random vector: $\mathbf{x}_1, \ldots, \mathbf{x}_2$.
- From these samples (assuming they do represent the population) we can estimate the expectation and covariance matrix:

  $$\hat{\mu} = \bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

  and

  $$\hat{\Sigma} = S = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\mathsf{T}$$

- We often 'forget' that we are dealing with estimates and write Σ instead of S and μ instead of $\bar{\mathbf{x}}$.
- In our Python programs we often collect all samples in a matrix of shape $n \times N$, the columns are the samples. We often call this matrix X which might be confusing as we are used to denote rv's with capital letters. But then: remember that in practice all our knownledge of the rv *is* the collection of samples from it.
- Python Code for calculation of the principle components is shown below.

```
from pylab import loadtxt, mean, tile, transpose, dot, \
     eig, argsort, plot, savefig, bar

def sortedeig(M):
   d, U = eig(M)
   si = argsort(d)[-1::-1]
   d = d[si]
   U = U[:,si]
   return (d,U)

X = loadtxt('data/data3d.txt',delimiter=',')
n,N = X.shape

xbar = mean(X,1)
Xzm = X - tile(xbar.reshape(n,1), N)
S = dot(Xzm,transpose(Xzm))/(N-1)

d,U = sortedeig(S)
bar(range(3),d)
savefig('figures/pcareduce.pdf')

x = X[:,42]                 # an 'arbitrary' vector from distribution
xzm = x - xbar;             # subtract mean
yzm = dot(transpose(U),xzm) # coordinates with respect to eigenvector basis

yzm_k = yzm[:-1];           # get rid of last element
xzm_k = dot(U[:,:-1],yzm_k) # recontruct xzm as linear combination of eigenvectors
x_k = xzm_k + xbar          # add the mean

print(x)                    # these two should be really close...
print(x_k)
```

```
None
```

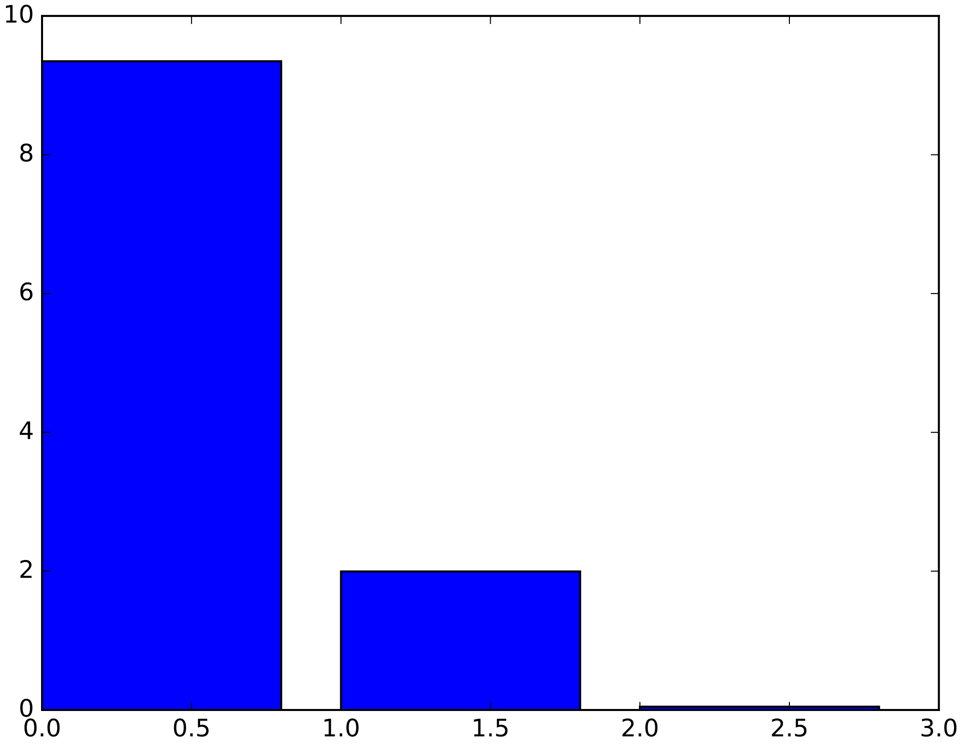

Figure 2: Eigenvalues for 'data3d.txt'

3.3 Reducing the dimensionality

- Let S be the covariance matrix of a zero mean set, and let $S = UDU^\intercal$ be the eigenvalue decomposition of S.
- A vector $\mathbf{x}$ with respect to the standard basis can be represented with respect to the eigenvector basis with the coordinate transform $\mathbf{y} = U^\intercal\mathbf{x}$.
- Assume that most information is supposed to be in the first k elements of the vector $\mathbf{y}$ with coordinates with respect to the eigenvector basis.
- Then we can truncate $\mathbf{y}$ by chopping off the last $n-k$ elements from $\mathbf{y}$ leading to vector $\mathbf{y}_k$.
- From this reduced dimension version of $\mathbf{y}$ we may reconstruct $\mathbf{x}$ anyway: $\hat{\mathbf{x}} = U_k\mathbf{y}_k$ where U_k is a matrix with k column vectors being the first k columnvectors of U.

4 Eigenvalues or Singular values

- Let X be a $n \times N$ data matrix whose columns are the samples from a distribution. Assume the data set has zero mean.
- The covariance matrix is estimated as:

$$S = XX^\intercal/(N-1)$$

- In case we write $X' = X/\sqrt{N-1}$ we have that $S = X'X'^{\mathsf{T}}$
- Let $X' = UDV^{\mathsf{T}}$ be the singular value decomposition of X'.
- This leads to $S = UDV^{\mathsf{T}}(UDV^{\mathsf{T}})^{\mathsf{T}} = UDV^{\mathsf{T}}VDU^{\mathsf{T}} = UD^2U^{\mathsf{T}}$, showing that U is the matrix with eigenvectors of S and D^2 is the matrix of eigenvalues of S.
- In practice it is advisable to use the Singular Value Decomposition instead of the eigen decomposition.

5 Exercises

5.1 Exercise: Incremental Calculation of the Covariance Matrix

Given n samples of a $d$-dimensional multivariate rv, we want to estimate the covariance matrix. Let $\mathbf{x}_i$ for $i = 1, \ldots, n$ be the vector samples. The estimator for the covariance matrix is defined as:

$$S = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^{\mathsf{T}}$$

where $\mathbf{m}$ is the estimator for the expectation (mean) of the distribution:

$$\mathbf{m} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

.

Prove that the expression for the covariance matrix can be rewritten as:

$$S = \frac{(\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^{\mathsf{T}}) - n\,\mathbf{m}\mathbf{m}^{\mathsf{T}}}{n-1}$$

Show that this expression can be used to calculate S *without* storing all examples $\mathbf{x}_i$ in memory.

5.2 LabExercise: Opponent Color Model

Color is a very complex phenomenon. For a physicist, color is accounted for by the distribution of electromagnetic wavelengths of light over the visible range from about 400 to 700 nm. For a biologist color is accounted for by the reponse of the three types of light receptors that are found in the human eye on the retina to measure the incoming light. Indeed what enters our visual brain is not the entire spectrum but the responses of three different sensors in our eye. The eye projects the infinite dimensional spectral space to a three dimensional color space. But why only three? In this exercise we will show that from a statistical point of view the 'choice' for a three dimensional colorspace is an efficient one in the sense that most of the variance in spectral data is accounted for by the first eigenvectors.

In this exercise we will start with a collection of spectra (the distributions of electromagnetic energy over the different wavelengths). A spectrum is a density measure (just like a probability density): you have to integrate it over an interval to calculate the energy of the waves with a wavelength in that interval.

Evidently the spectrum being a function requires an infinite amount of information to represent. In practice the spectra are sampled. In our case with sample interval of 1 nm with wavelength from 380 nm to 800 nm.

Scientists have collected spectral data for all kinds of colored beams and published this data on the internet. In this exercise we will look at a dataset of spectra obtained by looking at natural materials like flowers and plants.

In figure **??** an image of a red petunia is shown.

#A plot of a spectrum of red petunia is shown in figure #**??**.

The second dataset of spectra comes from looking at colored patches from the Munsell color tree. Munsell was a scientist who came up with a color system to describe all possible colors distinguishable by the human eye. The Munsell color system came with a book with colored patches for each color, or with a 'tree' with colored patches (shown in figure ??).

```
from pylab import *

# first read the 'natural' data set
data = loadtxt('data/natural400_700_5.asc')
nX = transpose(data)
nXzm = nX - tile( mean(nX,1)[:,newaxis],219 )
# now every column of Xzm is a spectrum with the mean spectrum subtracted

# Then read the 'munsell' data set. This data set is stored in a file
# as a long list of spectral elements one number on each line in the
# file. The first 421 lines are from the first spectrum, then follow
# the 421 lines with the elements of the second spectrum, etc. In
# total there are 1269 different spectra.
data = loadtxt('data/munsell380_800_1.asc')
mX = transpose(data.reshape(1269,421))
mXzm = mX - tile( mean(mX,1)[:,newaxis], 1269 )

# Plot the first spectrum of both data sets. Because the sampling is
# not the same we should explicitly set the 'x' value (the wavelength)
nx = arange(400,701,5)
mx = arange(380,801,1)

figure(1);
clf();
subplot(2,1,1)
plot( nx, nX[:,23], 'g' )
subplot(2,1,2)
plot( mx, mX[:,100], 'b' )
savefig('figures/twospectra.pdf')
```

Write the Python code needed to do the PCA analysis for both datasets. Plot the *scree diagrams* (the eigenvalues from high to low) for both data sets. How much of the variance is captured with the first three principal components? (this is best done by plotting the cumulative version of the vector containing the sorted eigenvalues, normalized by the sum of all eigenvalues). What do the first three eigenvectors (eigenspectra) look like?

If all went well you will have found that for both data sets the first principal component is an approximate constant spectrum. All wavelengths are present. Note that the PCA analysis is not aware of the fact that we are dealing with spectra and that for spectra all elements need to be positive (or zero). So the eigenvectors of the covariance matrix will undoubtly contain negative values.

Also write a function to reconstruct a spectrum from only its k principal components, i.e. the k eigenvectors with the largest eigenvalues.

5.3 LabExercise: EigenStructure

In computer vision a basis task is to analyze the local structure of images. Local structure is characterized by the size at which the details are considered (the scale) and the appearance of the details.

In this exercise we look at these image details from a statistical point of view.

Consider an image detail of $n \times n$ pixels. By stacking the pixels in a row (or column) line fashion an image patch is characterized with a vector $\mathbf{x} \in \mathbb{R}^{n \times n}$. A typical detail ranges in size from $n = 11$ to $n = 51$.

Let us collect a lot of local details (of the same size!). Even one image is a source for a lot of local details. For a large image there are very many $n \times n$ details that fit into the image (we do count overlapping details!). For this experiment we will use the image `trui.png`.

The following Python code will read the image into memory from disk and display it on the screen. Then we select a detail of size 25×25 and display that as well.

```
from pylab import *
a = imread('data/trui.png')
figure(1)
subplot(1,2,1)
imshow(a)
d = a[100:126,100:126]
subplot(1,2,2)
imshow(d)
savefig('figures/trui_with_detail.pdf',bbox_inches='tight')
print(d.shape)
```

Because an 256×256 image contains $232 \times 232 = 53824$ details of 25×25 pixels the matrix X would be a 625×53824 matrix. Therefore in calculating S we will use the incremental calculations as outlined in a previous exercise (each example vector $\mathbf{x}_i$ is added to the running sum vector—to calculate the mean afterwards—and the $\mathbf{x}_i\mathbf{x}_i^\intercal$ is added to the 625×625 matrix. Then after all samples are processed we can calculate the mean vector $\mathbf{m}$ and the sample covariance matrix S.

PCA analysis then reveals the EigenStructure of this image. Plot the 'scree diagram' showing the eigenvalues from high to low. Plot the first 6 eigenvectors as images!

Then write a function that displays a detail from the image (at a position that are parameters to the function) *and* the same detail but then reconstructed from only k eigenvectors (make k a parameter of your function). What would you think is a good choice for k to reconstruct image details with 'enough' accuracy?

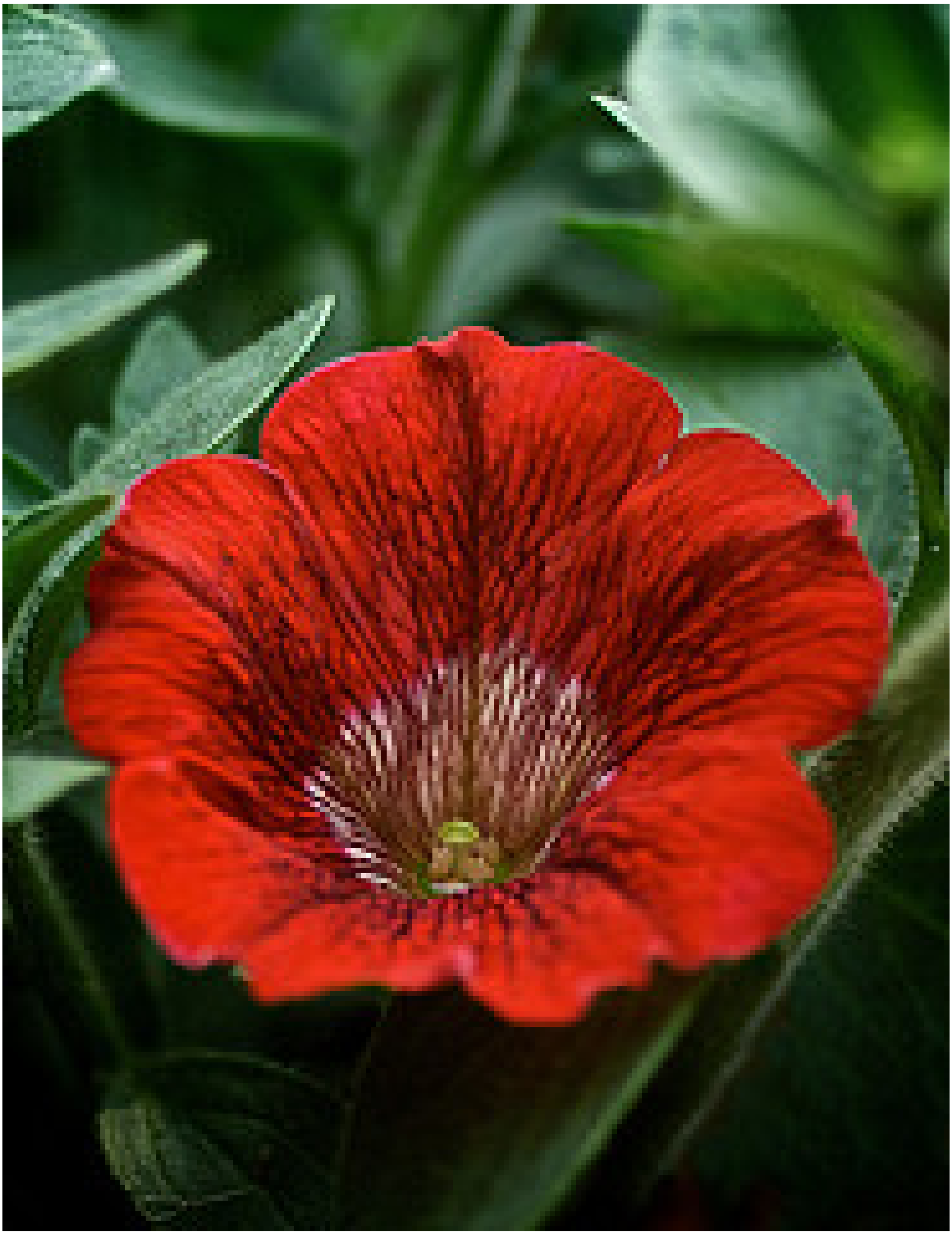

Figure 3: Red Petunia

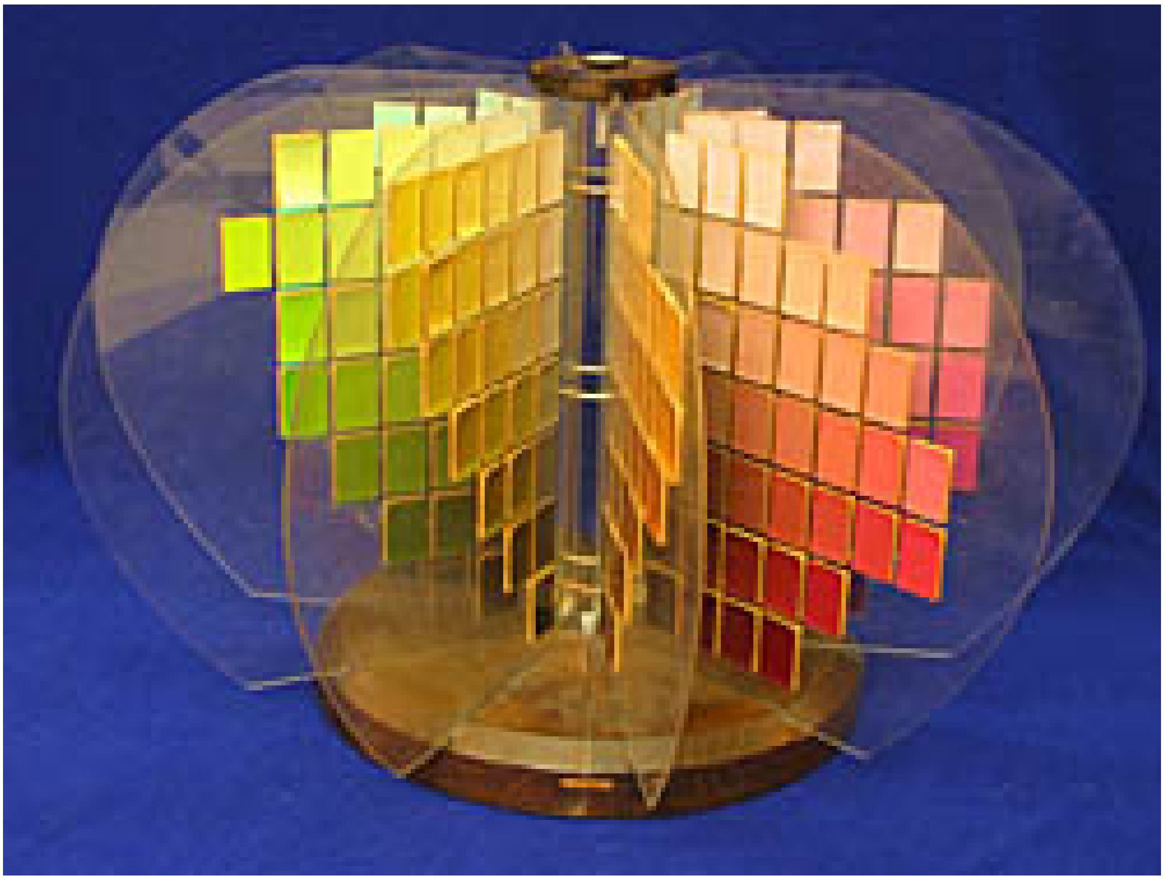

Figure 4: Munsell Color Tree

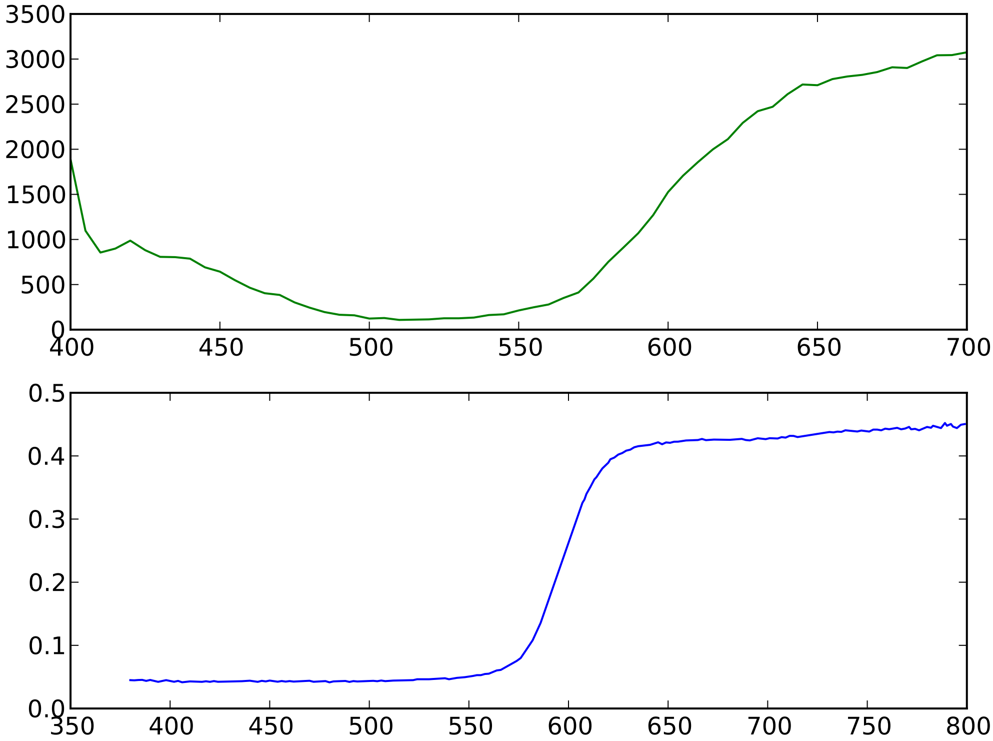

Figure 5: Two Spectra (top figure: natural spectrum, bottom figure: spectrum from Munsell color plate).

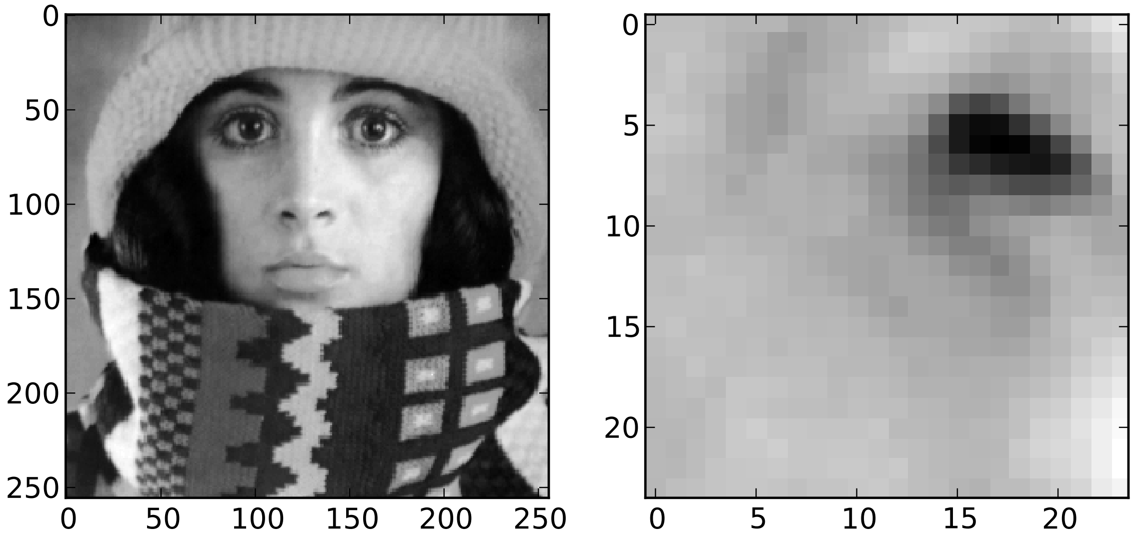

Figure 6: An image with a small 25×25 detail