Dimensionality reduction and regression

D. Jevremović

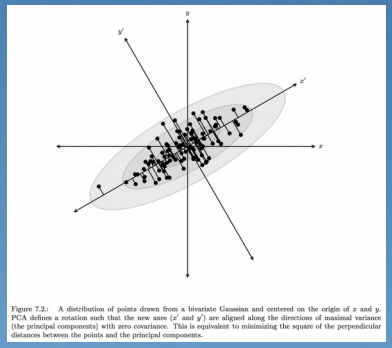
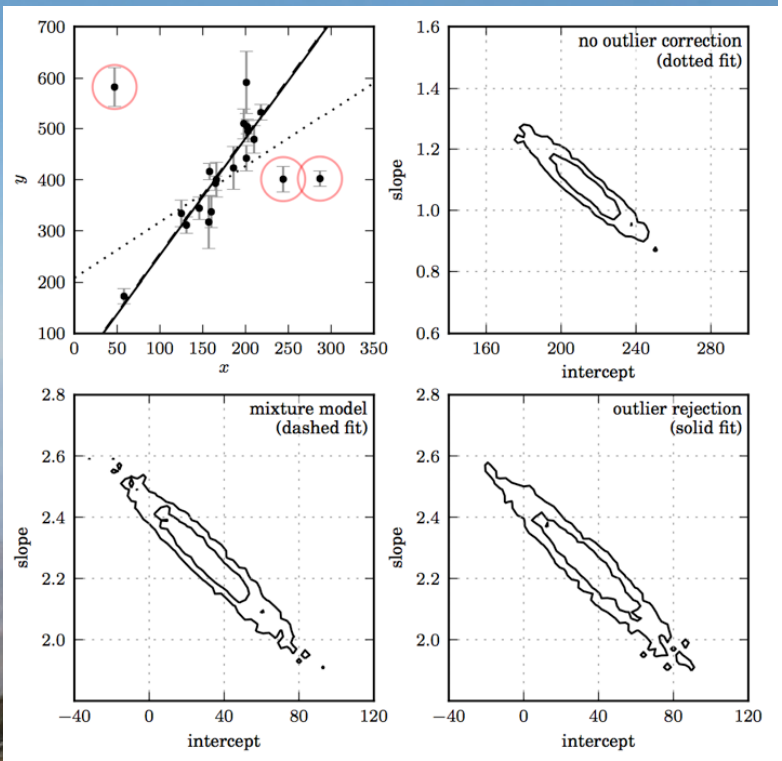


Figure 7.2.: A distribution of points drawn from a bivariate Gaussian and centered on the origin of x and y. PCA defines a rotation such that the new axes (x' and y') are aligned along the directions of maximal variance (the principal components) with zero covariance. This is equivalent to minimizing the square of the perpendicular distances between the points and the principal components.



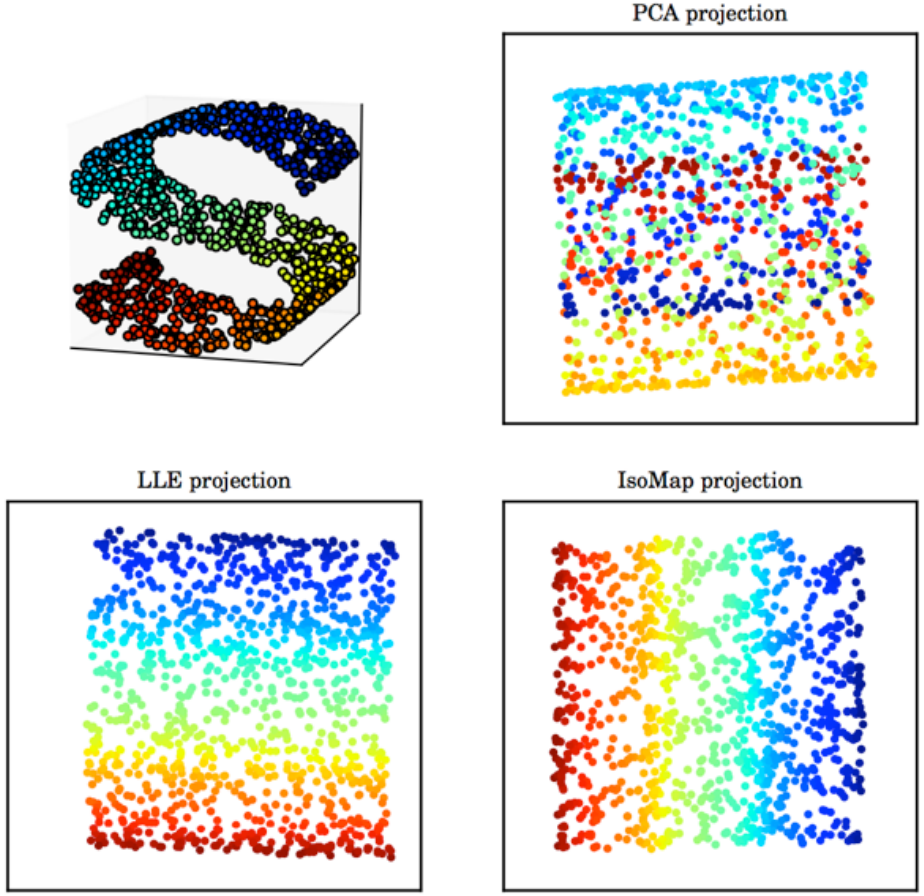
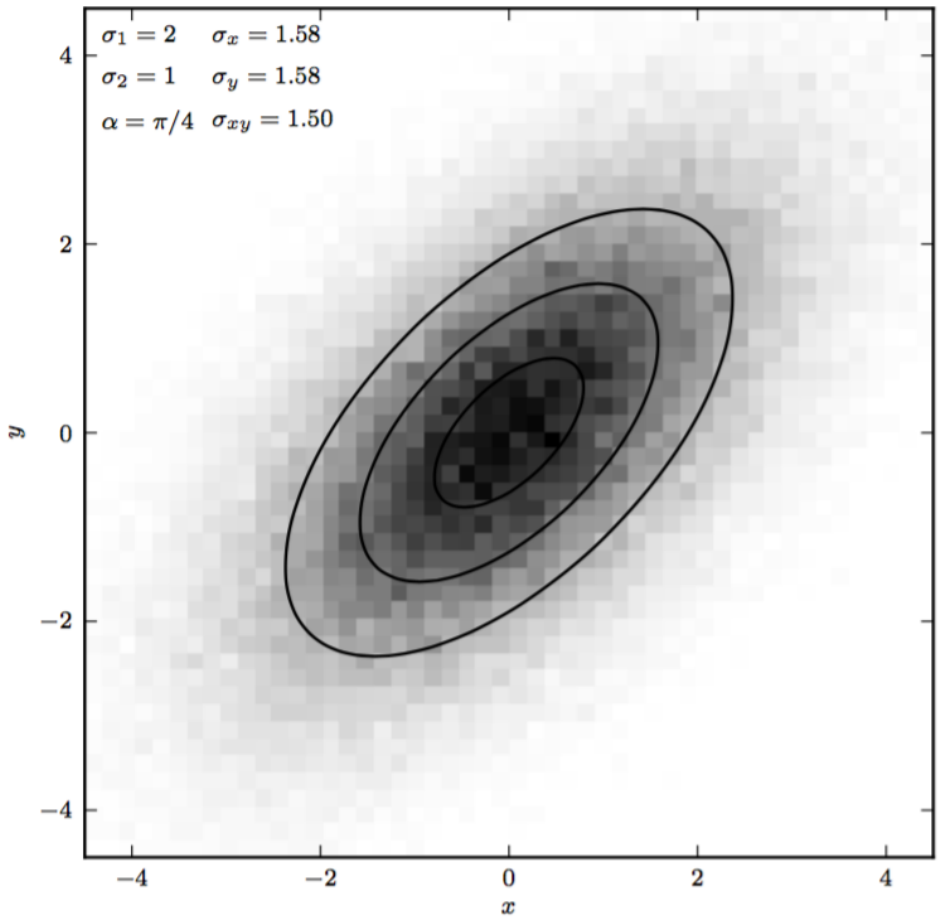


Figure 7.8.: A comparison of PCA and manifold learning. The top-left panel shows an example S-shaped data set (a two-dimensional manifold in a three-dimensional space). PCA identifies three principal components within the data. Projection onto the first two PCA components results in a mixing of the colors along the manifold. Manifold learning (LLE and IsoMap) preserves the local structure when projecting the data, preventing the mixing of the colors.

Outline

- **Dimensionality Reduction**
  - Covariance and Correlation
  - **Principal Component Analysis**
  - Non-negative Matrix Factorization
  - Independent Component Analysis
  - Manifold learning (Locally Linear Embedding)
- Regression and a few misc. points
  - (Gaussian) **errors in both variables**
  - regression with **non-Gaussian errors and/or outliers**
  - (fast matching using KD trees)



An example of a 2D pdf:

h(x,y)

Figure 3.22.: An example of data generated from a bivariate Gaussian distribution. The shaded pixels are a Hess diagram showing the density of points at each position.

Covariance and Correlation

$$V_x = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \mu_x)^2 \, h(x, y) \, dx \, dy \tag{3.71}$$

and

$$V_y = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (y - \mu_y)^2 \, h(x, y) \, dx \, dy, \tag{3.72}$$

where the mean values are defined as

$$\mu_x = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x \, h(x, y) \, dx \, dy \tag{3.73}$$

and analogously for μ_y. In addition, the covariance of x and y, which is a measure of the dependence of the two variables on each other, is defined as

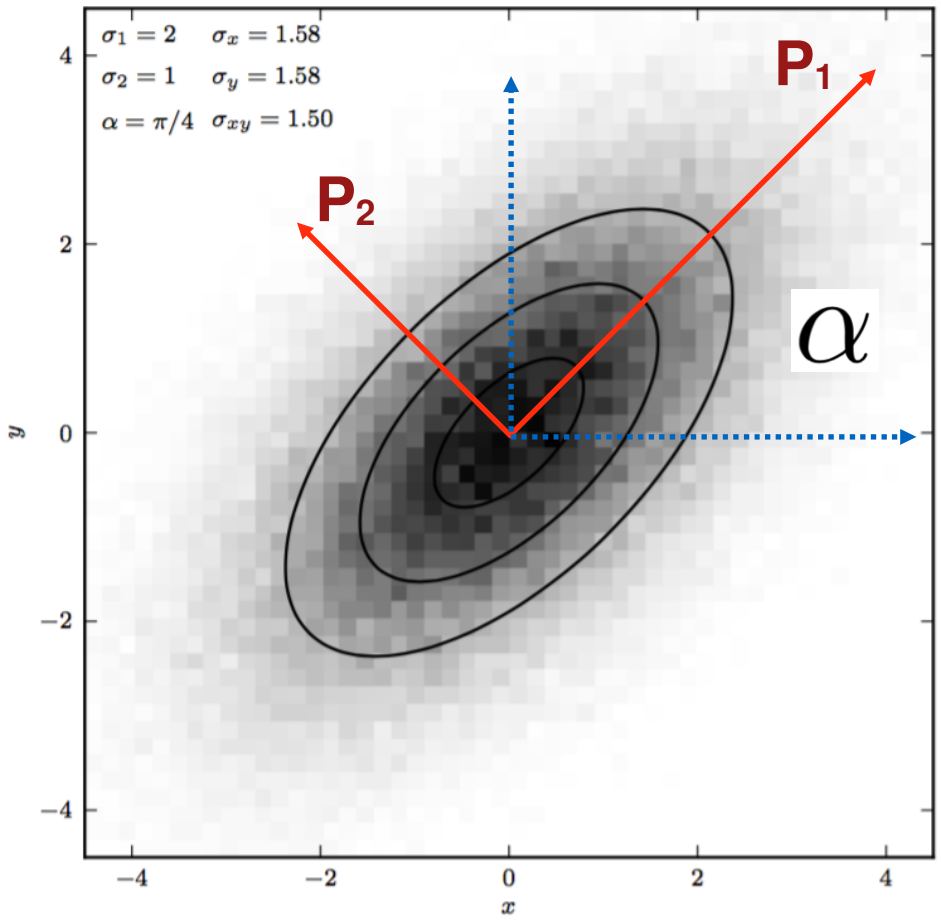
$$V_{xy} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \mu_x)(y - \mu_y) \, h(x, y) \, dx \, dy. \tag{3.74}$$

Sometimes, Cov(x,y) is used instead of V_{xy}. For later convenience, we define $\sigma_x = \sqrt{V_x}$, $\sigma_y = \sqrt{V_y}$, and $\sigma_{xy} = V_{xy}$ (note that there is no square root; i.e., the unit for σ_{xy} is the square of the unit for σ_x and σ_y). A very useful related result is that the variance of the sum $z = x + y$ is

$$V_z = V_x + V_y + 2 \, V_{xy}. \tag{3.75}$$

When x and y are uncorrelated ($V_{xy} = 0$), the variance of their sum is equal to the sum of their variances. For $w = x - y$,

$$V_w = V_x + V_y - 2 \, V_{xy}. \tag{3.76}$$



In x-y coordinate system, h(x,y) has non-vanishing covariance V_{xy}

In P1-P2 coord. system, $V_{P1P2} = 0$ (symmetry!)

Figure 3.22.: An example of data generated from a bivariate Gaussian distribution. The shaded pixels are a Hess diagram showing the density of points at each position.

3.5.2. Bivariate Gaussian distributions

A generalization of the Gaussian distribution to the two-dimensional case is given by

$$p(x, y|\mu_x, \mu_y, \sigma_x, \sigma_y, \sigma_{xy}) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(\frac{-z^2}{2(1-\rho^2)}\right), \tag{3.79}$$

where

$$z^2 = \frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - 2\,\rho\,\frac{(x-\mu_x)\,(y-\mu_y)}{\sigma_x\,\sigma_y}, \tag{3.80}$$

and the (dimensionless) correlation coefficient between x and y is defined as

$$\rho = \frac{\sigma_{xy}}{\sigma_x\sigma_y} \tag{3.81}$$

(see figure 3.22). For perfectly correlated variables such that $y = ax + b$, $\rho = a/|a| \equiv \mathrm{sign}(a)$, and for uncorrelated variables, $\rho = 0$. The *population* correlation coefficient ρ is directly related to Pearson's *sample* correlation coefficient r discussed in §3.6.

The contours in the (x, y) plane defined by $p(x, y|\mu_x, \mu_y, \sigma_x, \sigma_y, \sigma_{xy})$=constant are ellipses centered on $(x = \mu_x,\ y = \mu_y)$, and the angle α (defined for $-\pi/2 \le \alpha \le \pi/2$) between the x-axis and the ellipses' major axis is given by

$$\tan(2\alpha) = 2\,\rho\,\frac{\sigma_x\sigma_y}{\sigma_x^2 - \sigma_y^2} = 2\frac{\sigma_{xy}}{\sigma_x^2 - \sigma_y^2}. \tag{3.82}$$

When the (x, y) coordinate system is rotated by an angle α around the point $(x = \mu_x, y = \mu_y)$,

$$\begin{aligned} P_1 &= (x - \mu_x)\cos\alpha + (y - \mu_y)\sin\alpha, \\ P_2 &= -(x - \mu_x)\sin\alpha + (y - \mu_y)\cos\alpha, \end{aligned} \tag{3.83}$$

the correlation between the two new variables P_1 and P_2 disappears, and the two widths are

$$\sigma^2_{1,2} = \frac{\sigma_x^2 + \sigma_y^2}{2} \pm \sqrt{\left(\frac{\sigma_x^2 - \sigma_y^2}{2}\right)^2 + \sigma_{xy}^2}. \tag{3.84}$$

The coordinate axes P_1 and P_2 are called the principal axes, and σ_1 and σ_2 represent the minimum and maximum widths obtainable for any rotation of the coordinate axes. In this coordinate system where the correlation vanishes, the bivariate Gaussian is the product of two univariate Gaussians (see eq. 3.78). We shall discuss a multidimensional extension of this idea (principal component analysis) in chapter 7.

Alternatively, starting from the principal axes frame, we can compute

$$\sigma_x = \sqrt{\sigma_1^2 \cos^2\alpha + \sigma_2^2 \sin^2\alpha}, \tag{3.85}$$

$$\sigma_y = \sqrt{\sigma_1^2 \sin^2\alpha + \sigma_2^2 \cos^2\alpha}, \tag{3.86}$$

and ($\sigma_1 \geq \sigma_2$ by definition)

$$\sigma_{xy} = (\sigma_1^2 - \sigma_2^2)\sin\alpha\cos\alpha. \tag{3.87}$$

Note that σ_{xy}, and thus the correlation coefficient ρ, vanish for both $\alpha = 0$ and $\alpha = \pi/2$, and have maximum values for $\pi/4$. By inverting eq. 3.83, we get

$$\begin{aligned} x &= \mu_x + P_1\cos\alpha - P_2\sin\alpha, \\ y &= \mu_y + P_1\sin\alpha + P_2\cos\alpha. \end{aligned} \tag{3.88}$$

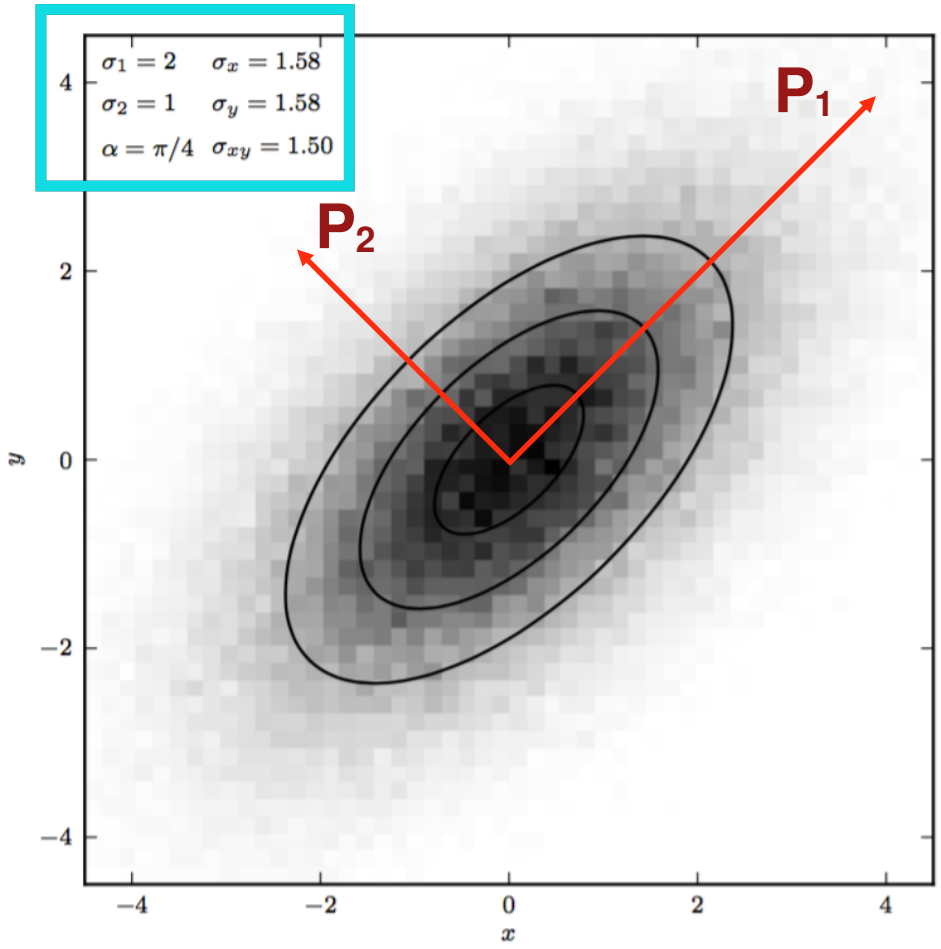


Figure 3.22.: An example of data generated from a bivariate Gaussian distribution. The shaded pixels are a Hess diagram showing the density of points at each position.

Covariance and Correlation

3.6. Correlation coefficients

Several correlation tests are implemented in `scipy.stats`, including `spearmanr`, `kendalltau`, and `pearsonr`:

```
from scipy import stats
x, y = np.random.random((2, 100))  # two random arrays
corr_coeff, p_value = stats.pearsonr(x, y)
rho, p_value = stats.spearmanr(x, y)
tau, p_value = stats.kendalltau(x, y)
```

We have already introduced the covariance of x and y (see eq. 3.74) and the correlation coefficient (see eq. 3.81) as measures of the dependence of the two variables on each other. Here we extend our discussion to the interpretation of the sample correlation coefficient.

Given two data sets of equal size N, $\{x_i\}$ and $\{y_i\}$, Pearson's sample correlation coefficient is

$$r = \frac{\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \overline{y})^2}}, \tag{3.102}$$

with $-1 \leq r \leq 1$. For uncorrelated variables, $r = 0$. If the pairs (x_i, y_i) are drawn from two uncorrelated univariate Gaussian distributions (i.e., the population correlation coefficient $\rho = 0$), then the distribution of r follows Student's t distribution with $k = N - 2$ degrees of freedom and

$$t = r\sqrt{\frac{N-2}{1-r^2}}. \tag{3.103}$$

Covariance and Correlation

3.6. Correlation coefficients

Several correlation tests are implemented in `scipy.stats`, including `spearmanr`, `kendalltau`, and `pearsonr`:

```
from scipy import stats
x, y = np.random.random((2, 100))  # two random arrays
corr_coeff, p_value = stats.pearsonr(x, y)
rho, p_value = stats.spearmanr(x, y)
tau, p_value = stats.kendalltau(x, y)
```

Let's try it!

1) start a new jupyter notebook
2) generate N pairs (x,y) by drawing each from Gaussian N(0,1)
3) compute r using eq. 3.102 from the previous slide
4) note that you *could* compute r using scipy code above
5) plot histogram of r (for N=10 and 100)
6) compute t (using eq. 3.103), plot its histogram and overplot Student's t distribution and comment

with $-1 \leq r \leq 1$. For uncorrelated variables, $r = 0$. If the pairs (x_i, y_i) are drawn from two uncorrelated univariate Gaussian distributions (i.e., the population correlation coefficient $\rho = 0$), then the distribution of r follows Student's t distribution with $k = N - 2$ degrees of freedom and

$$t = r\sqrt{\frac{N-2}{1-r^2}}. \qquad (3.103)$$

Covariance and Correlation

Pearson's correlation coefficient has two main deficiencies. First, the measurement errors for $\{x_i\}$ and $\{y_i\}$ are not used. As they become very large, the significance of any given value of r will decrease (if errors are much larger than the actual ranges of data values, the evidence for correlation vanishes). The case of nonnegligible errors is discussed in chapter 8. Second, Pearson's correlation coefficient is sensitive to Gaussian outliers (recall the discussion in the previous section) and, more generally, the distribution of r does not follow Student's t distribution if $\{x_i\}$ and $\{y_i\}$ are not drawn from a bivariate Gaussian distribution. In such cases, nonparametric correlation tests, discussed next, are a better option.

3.6.1. Nonparametric correlation tests

The two best known nonparametric (distribution-free) correlation tests are based on Spearman's correlation coefficient and Kendall's correlation coefficient. These two correlation coefficients are themselves strongly correlated ($r = 0.98$ in the case of Gaussian distributions), though their magnitude is not equal (see below). Traditionally, Spearman's correlation coefficient is used more often because it is easier to compute, but Kendall's correlation coefficient is gaining popularity because it approaches normality faster than Spearman's correlation coefficient.

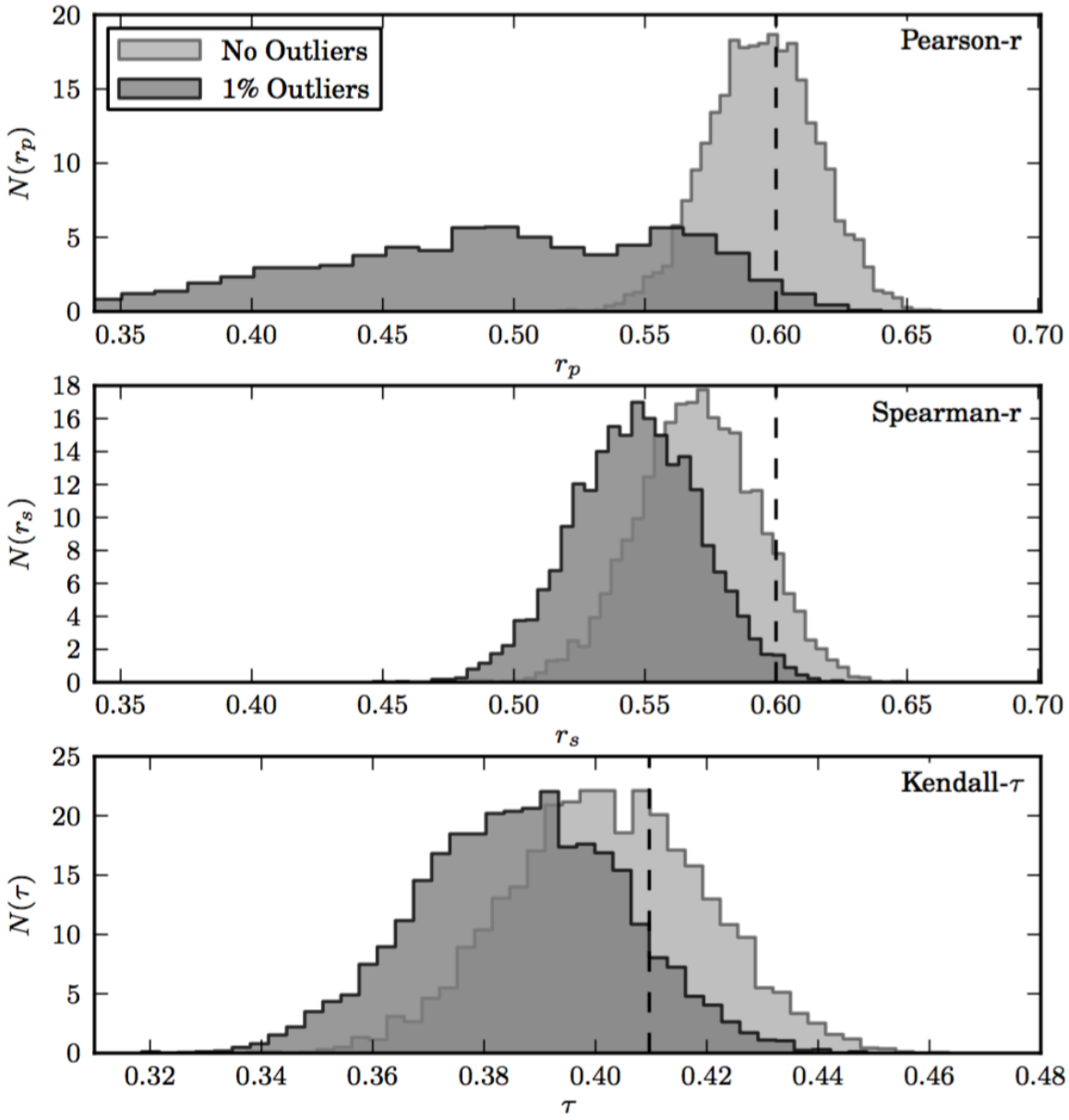


Figure 3.24.: Bootstrap estimates of the distribution of Pearson's, Spearman's, and Kendall's correlation coefficients based on 2000 resamplings of the 1000 points shown in figure 3.23. The true values are shown by the dashed lines. It is clear that Pearson's correlation coefficient is not robust to contamination.

Principal Component Analysis (PCA)

Motivation:

2D case: what is the direction of largest variance in the data? Equivalently, what is the rotation of the coordinate system in which results in no covariance between the variables?

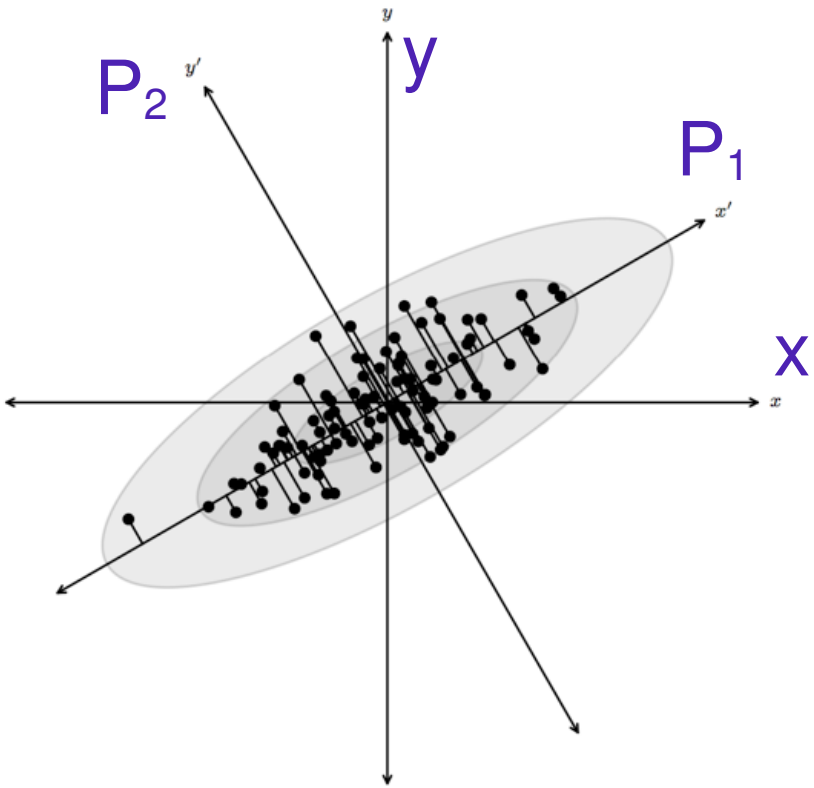


Figure 7.2.: A distribution of points drawn from a bivariate Gaussian and centered on the origin of x and y. PCA defines a rotation such that the new axes (x' and y') are aligned along the directions of maximal variance (the principal components) with zero covariance. This is equivalent to minimizing the square of the perpendicular distances between the points and the principal components.

The two variables x and y have a non-vanishing covariance; the covariance in the P_1 vs. P_2 coordinate system is 0 by construction (recall discussion of 2-D Gaussians) (Pearson 1901)

Principal Component Analysis (PCA)

Motivation:

High-D case (driven by the "curse of dimensionality"): there are ~4000 data points in SDSS spectra. Can we represent these spectra as linear combinations of a **much smaller** number of eigen-components?

The curse of dimensionality:

in high-D space, the ratio of the volume of the unit hyper-sphere and the volume of the hyper-cube that encloses it goes to 0 with as D increases

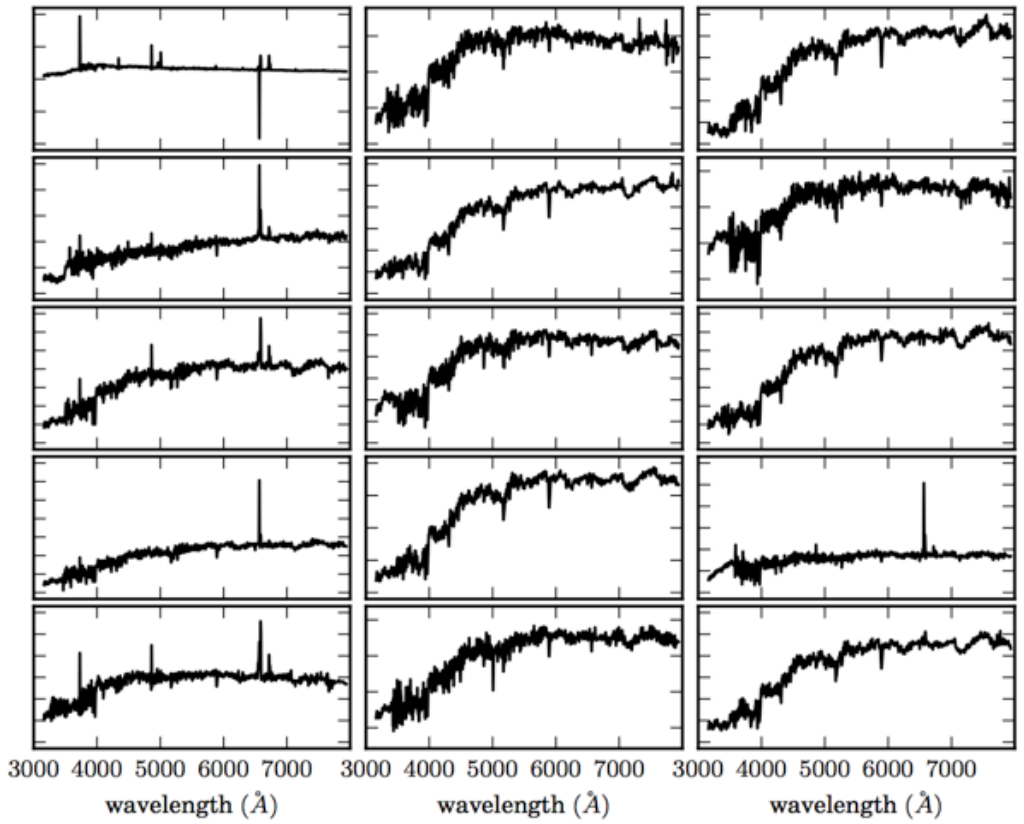


Figure 7.1.: A sample of 15 galaxy spectra selected from the SDSS spectroscopic data set (see §1.5.5). These spectra span a range of galaxy types, from star-forming to passive galaxies. Each spectrum has been shifted to its rest frame and covers the wavelength interval 3000–8000 Å. The specific fluxes, $F_\lambda(\lambda)$, on the ordinate axes have an arbitrary scaling.

Principal Component Analysis (PCA)

Each spectrum $\boldsymbol{x}_i(k)$ can be described by

$$x = \mu_x + P_1 \cos\alpha - P_2 \sin\alpha,$$
$$y = \mu_y + P_1 \sin\alpha + P_2 \cos\alpha.$$

$$\boldsymbol{x}_i(k) = \boldsymbol{\mu}(k) + \sum_j^R \theta_{ij} \boldsymbol{e}_j(k), \tag{7.17}$$

where i represents the number of the input spectrum, j represents the number of the eigenspectrum, and, for the case of a spectrum, k represents the wavelength. Here, $\boldsymbol{\mu}(k)$ is the mean spectrum and θ_{ij} are the linear expansion coefficients derived from

$$\theta_{ij} = \sum_k \boldsymbol{e}_j(k)(\boldsymbol{x}_i(k) - \boldsymbol{\mu}(k)). \tag{7.18}$$

R is the total number of eigenvectors (given by the rank of X, $\min(N,K)$). If the summation is over all eigenvectors, the input spectrum is fully described with no loss of information. Truncating this expansion (i.e., $r < R$),

$$\boldsymbol{x}_i(k) = \sum_i^{r<R} \theta_i \boldsymbol{e}_i(k), \tag{7.19}$$

will exclude those eigencomponents with smaller eigenvalues. These components will, predominantly, reflect the noise within the data set. This is reflected in figure 7.5: truncating the recon-

Principal Component Analysis (PCA)

sometimes called Karhunen-Loeve and Hotelling transform

How is it done:

Data matrix is formed by N sets (objects) of K measured features (attributes); e.g. K~4000 for SDSS spectra of, say, N=100,000 quasars; we subtract the mean of each feature (and sometimes normalized by their st. deviation, called whitening; in case of spectra, flux is normalized to 1 to avoid uninteresting correlations with source brightness)

Using matrix algebra, the first eigen-component is found by maximizing variance and other eigen-components by requiring covariance to vanish.

There are different approaches, whose performance depends on N and K.

Principal Component Analysis (PCA)

sometimes called Karhunen-Loeve and Hotelling transform

How is it done:

Different matrix algebra approaches to PCA:

1) Singular Value Decomposition of the data matrix:
efficient in most practical cases (exceptions below)

2) Eigenvalue Decomposition of the covariance matrix:
efficient for N >> K

3) Eigenvalue Decomposition of the correlation matrix:
efficient for K >> N

Principal Component Analysis (PCA)

Singular Value Decomposition

Suppose $\mathbf{M}$ is a $m \times n$ matrix whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then there exists a factorization, called a singular value decomposition of $\mathbf{M}$, of the form

$$\mathbf{M} = \mathbf{U\Sigma V}^*$$

where

- $\mathbf{U}$ is an $m \times m$ unitary matrix (if $K = \mathbb{R}$, unitary matrices are orthogonal matrices),
- $\mathbf{\Sigma}$ is a diagonal $m \times n$ matrix with non-negative real numbers on the diagonal,
- $\mathbf{V}$ is an $n \times n$ unitary matrix over K, and
- $\mathbf{V}^*$ is the conjugate transpose of $\mathbf{V}$.

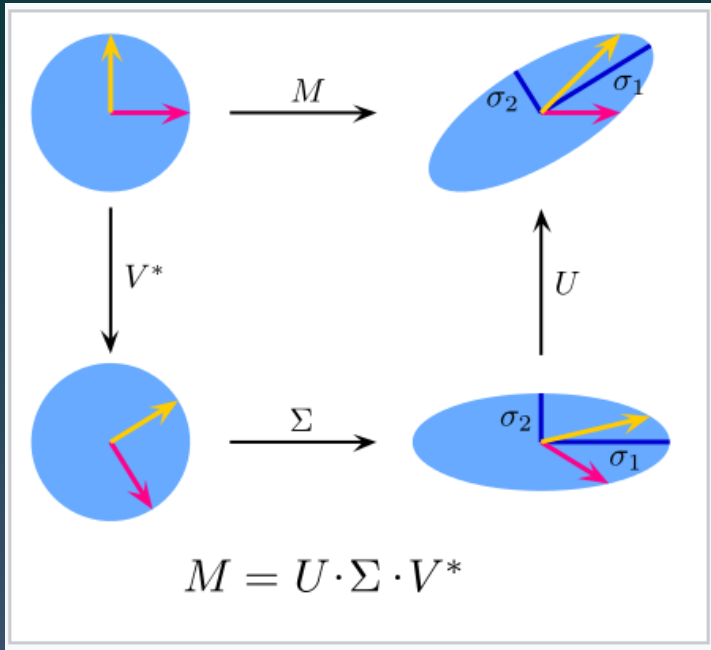
The diagonal entries σ_i of $\mathbf{\Sigma}$ are known as the **singular values** of $\mathbf{M}$. A common convention is to list the singular values in descending order. In this case, the diagonal matrix, $\mathbf{\Sigma}$, is uniquely determined by $\mathbf{M}$ (though not the matrices $\mathbf{U}$ and $\mathbf{V}$, see below).

If X is a matrix of N observations of K-dimensional data, then the PCA gives

T = X W,

where W is a K-by-K matrix whose columns are the eigenvectors of X^TX

Note: X^TX is the covariance matrix of X and it corresponds to matrix M above



The image shows:

Upper Left: The unit disc with the two canonical unit vectors

Upper Right: Unit disc transformed with M and singular Values σ_1 and σ_2 indicated

Lower Left: The action of $\mathbf{V}^*$ on the unit disc. This is just a rotation.

Lower Right: The action of $\mathbf{\Sigma V}^*$ on the unit disc. Sigma scales in vertically and horizontally.

X is a matrix of N observations of K-dimensional data

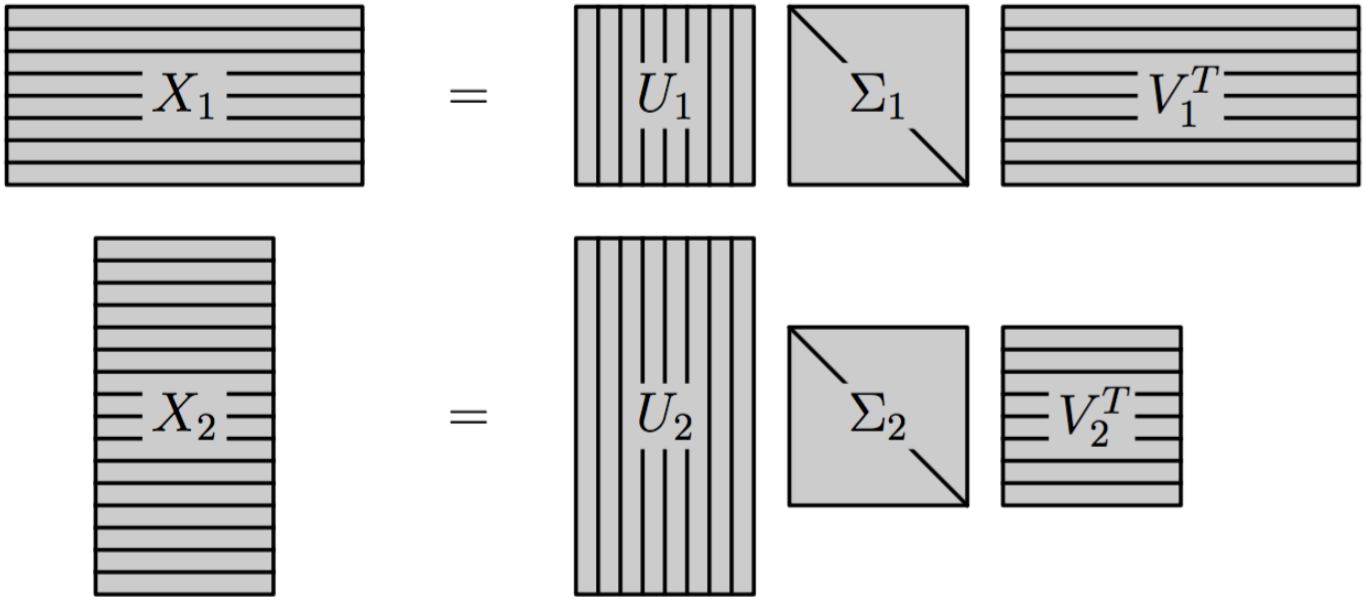


Figure 7.3.: Singular value decomposition (SVD) can factorize an $N \times K$ matrix into $U\Sigma V^T$. There are different conventions for computing the SVD in the literature, and this figure illustrates the convention used in this text. The matrix of singular values Σ is always a square matrix of size $[R \times R]$ where $R = \min(N, K)$. The shape of the resulting U and V matrices depends on whether N or K is larger. The columns of the matrix U are called the left-singular vectors, and the columns of the matrix V are called the right-singular vectors. The columns are orthonormal bases, and satisfy $U^T U = V^T V = I$.

NumPy and SciPy contain powerful suites of linear algebra tools. For example, we can confirm the above relationship using `svd` for computing the SVD, and `eigh` for computing the symmetric (or in general Hermitian) eigenvalue decomposition:

```
>>> import numpy as np
>>> X = np.random.random((100, 3))
>>> CX = np.dot(X.T, X)
>>> U, Sdiag, VT = np.linalg.svd(X, full_matrices=False)
>>> CYdiag, R = np.linalg.eigh(CX)
```

Principal Component Analysis (PCA)

How is it done:

PCA can be performed easily using Scikit-learn:

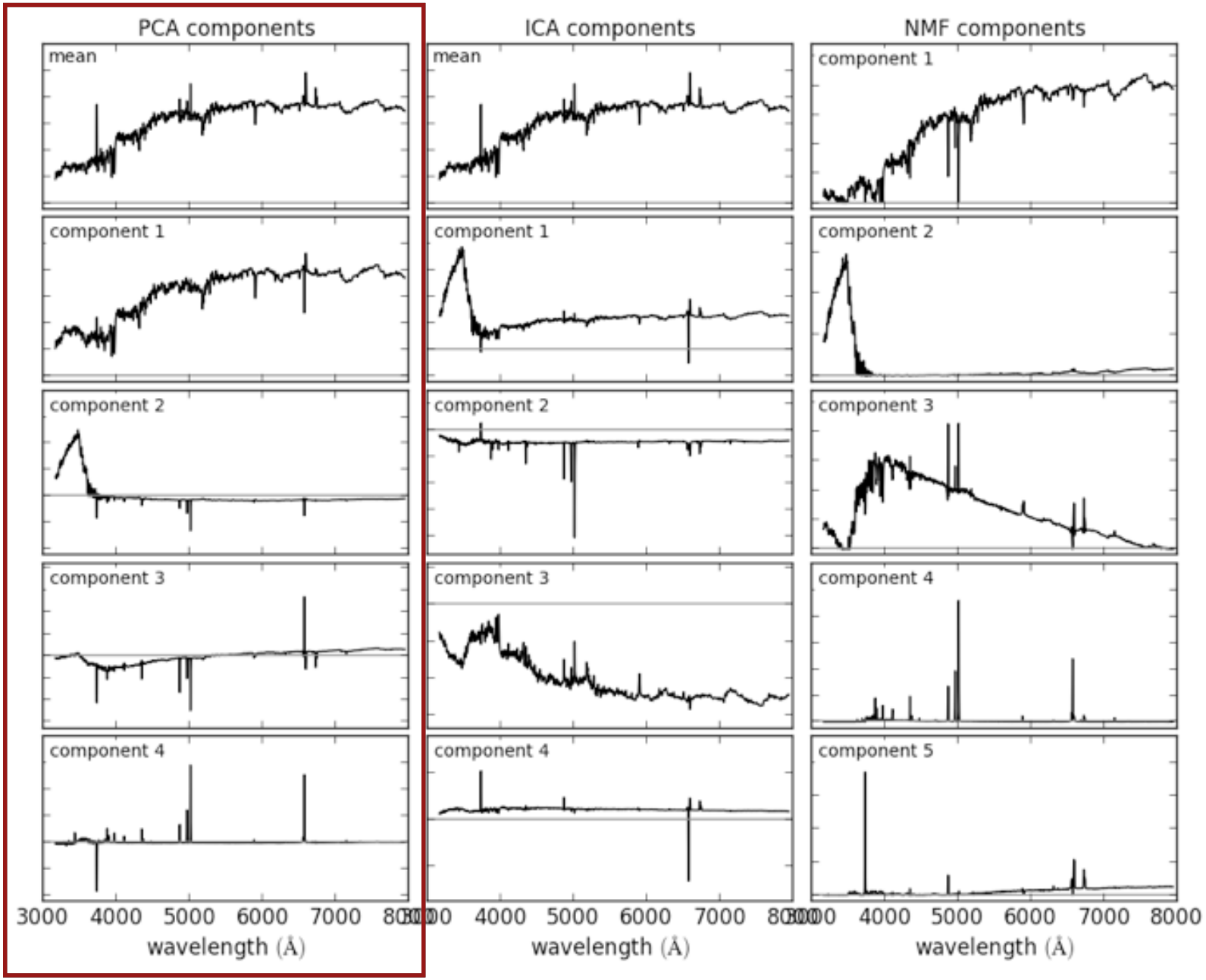
```
import numpy as np
from sklearn.decomposition import PCA

X = np.random.normal(size=(100, 3))  # 100 points in 3 dimensions
R = np.random.random((3, 10))  # projection matrix
X = np.dot(X, R)  # X is now 10-dim, with 5 intrinsic dims
pca = PCA(n_components=4)  # n_components can be optionally set
pca.fit(X)
comp = pca.transform(X)  # compute the subspace projection of X

mean = pca.mean_  # length 10 mean of the data
components = pca.components_  # 4 x 10 matrix of components
var = pca.explained_variance_  # the length 4 array of eigenvalues
```

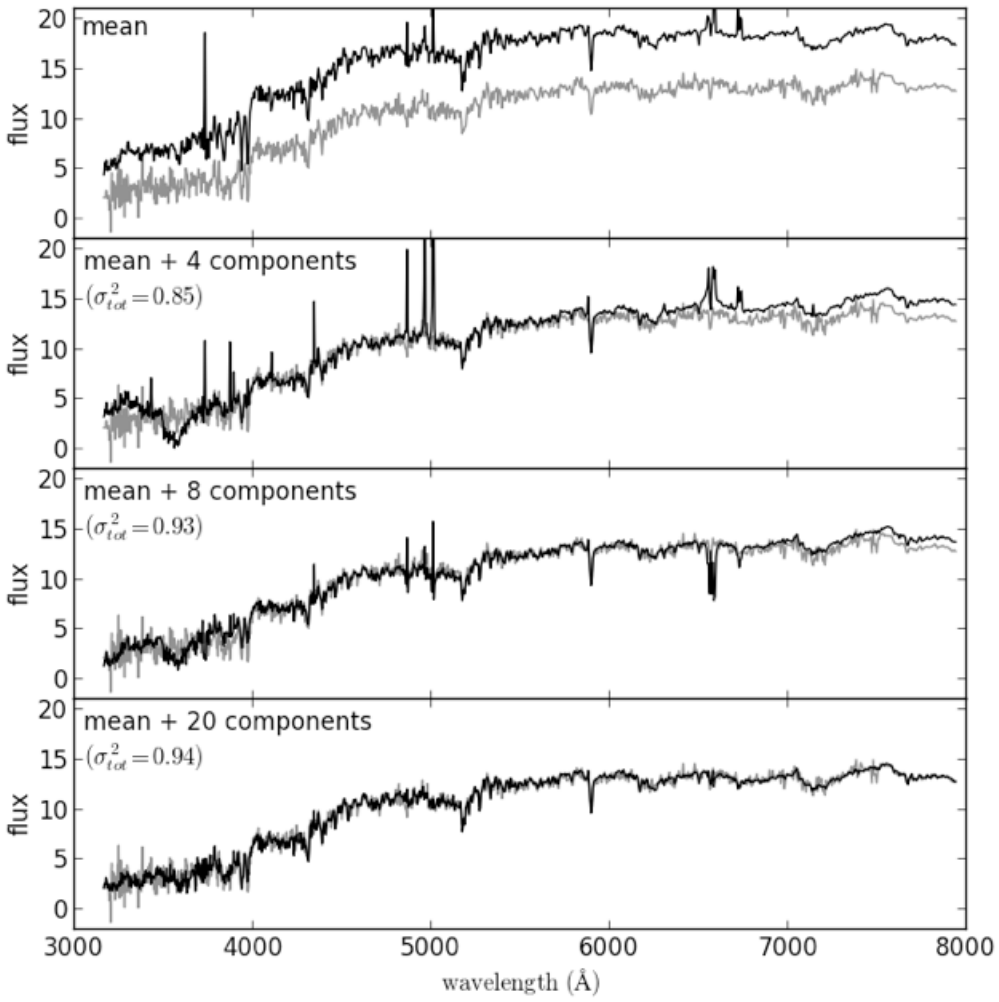
For detailed information and examples, see:
http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

- we will make this plot a little bit later…



Principal Component Analysis (PCA)

- make this plot by running
  %run fig_spec_reconstruction.py



How to choose the number of eigencomponents ?

Mean spectrum

4 eigencomponents

8 eigencomponents

20 eigencomponents

Principal Component Analysis (PCA)

How to choose the number of eigencomponents?

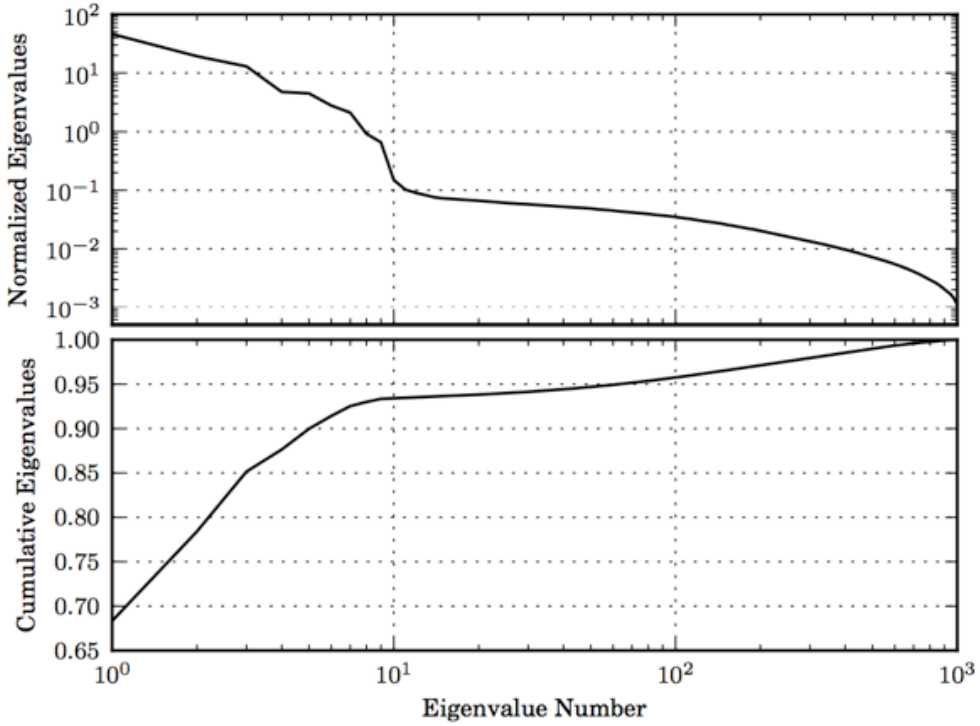


Figure 7.4.: The eigenvalues for the PCA decomposition of the SDSS spectra described in §7.3.2. The top panel shows the decrease in eigenvalue as a function of the number of eigenvectors, with a break in the distribution at ten eigenvectors. The lower panel shows the cumulative sum of eigenvalues normalized to unity. 94% of the variance in the SDSS spectra can be captured using the first ten eigenvectors.

There is no universal method, but typically we require that some fraction of data variance is captured by truncated series.

This plot (called the scree plot) shows that the first ten eigencomponents already capture about 94% of the variance in the dataset. Higher terms attempt to describe high-frequency measurement noise.

Principal Component Analysis (PCA)

A few more points about PCA

It can treat missing data by introducing weights for each point (Connolly & Szalay, 1999, AJ 117, 2052).

If the dataset cannot fit in memory, then PCA computation can be challenging. One way to address this is the use of iterative online algorithms.

In some problems, linear decomposition might not be appropriate (e.g. a set of Planck functions with varying temperature)

Eigencomponents can be negative, which in some applications provides only limited insight into underlying physics (e.g. galaxy spectra).

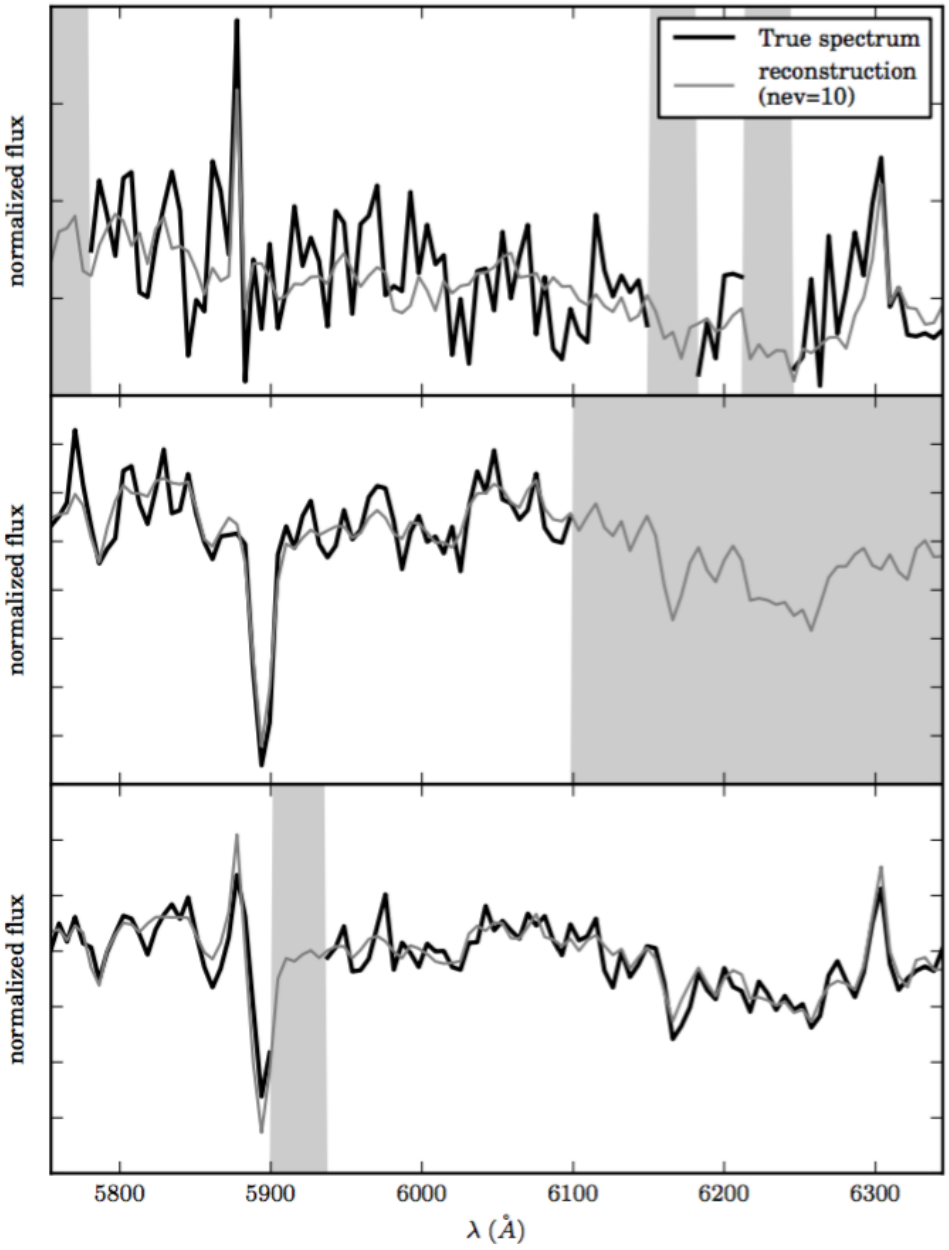


Figure 7.6.: The principal component vectors defined for the SDSS spectra can be used to interpolate across or reconstruct missing data. Examples of three masked spectral regions are shown comparing the reconstruction of the input spectrum (black line) using the mean and the first ten eigenspectra (blue line) The gray bands represent the masked region of the spectrum.

Glossary

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant= $50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

From: http://statweb.stanford.edu/~tibs/stat315a/glossary.pdf

Non-negative Matrix Factorization (NMF)

Attempts to address the fact that PCA eigencomponents can be negative even when we have physical reasons (a priori knowledge) to expect them to be non-negative

Assumes that the data matrix X is a product of two positive matrices, $X=Y*W$

Solved iteratively for Y and W by minimizing the reconstruction error $||(X-WY)^2||$; sometimes problems with local minima are encountered (Lee & Seung, 2001)

We will see an example after introducing ICA

Independent Component Analysis (ICA)

Also known as the “coctail party problem”, it assumes that the signal is a linear combination of components:

Each spectrum, $x_i(k)$, can now be described by

$$x_1(k) = a_{11}s_1(k) + a_{12}s_2(k) + a_{13}s_3(k) + \cdots, \quad (7.34)$$
$$x_2(k) = a_{21}s_1(k) + a_{22}s_2(k) + a_{23}s_3(k) + \cdots, \quad (7.35)$$
$$x_3(k) = a_{31}s_1(k) + a_{32}s_2(k) + a_{33}s_3(k) + \cdots, \quad (7.36)$$

where $s_i(k)$ are the individual stellar spectra and a_{ij} the appropriate mixing amplitudes. In matrix format we can write this as

$$X = AS, \quad (7.37)$$

where X and S are matrices for the set of input spectra and stellar spectra, respectively. Extracting these signal spectra is equivalent to estimating the appropriate weight matrix, W, such that

$$S = WX. \quad (7.38)$$

This appears very similar to PCA, but...

Independent Component Analysis (ICA)

The principle that underlies ICA comes from the observation that the input signals, $s_i(k)$, should be statistically independent. Two random variables are considered statistically independent if their joint probability distribution, $f(x, y)$, can be fully described by a combination of their marginalized probabilities, that is,

$$f(x^p, y^q) = f(x^p) f(y^q), \tag{7.39}$$

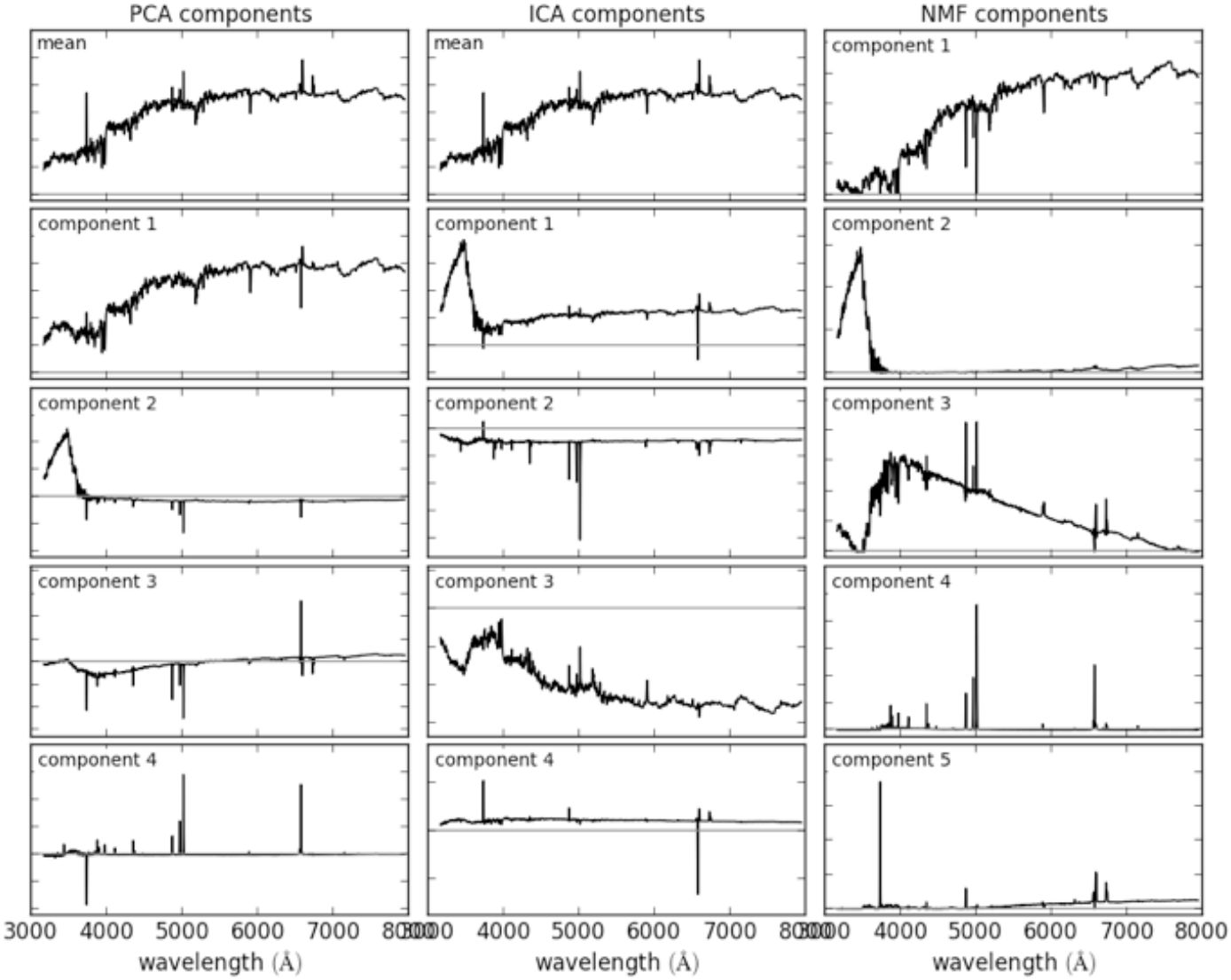
where p and q represent arbitrary higher-order moments of the probability distributions. For the case of PCA, $p = q = 1$ and the statement of independence simplifies to the weaker condition of uncorrelated data (see §7.3.1 on the derivation of PCA).

In most implementations of ICA algorithms the requirement for statistical independence is expressed in terms of the non-Gaussianity of the probability distributions. The rationale for this is that the sum of any two independent random variables will always be more Gaussian than either

Let's now compare PCA, NMF and ICA using our sample of SDSS galaxy spectra.

Comparison of PCA, ICA and NMF

- make this plot by running (it takes a minute)
  %run fig_spec_decompositions.py



Which one is the most astrophysical?

Manifold Learning

What do we do if the assumption of linearity does not hold?

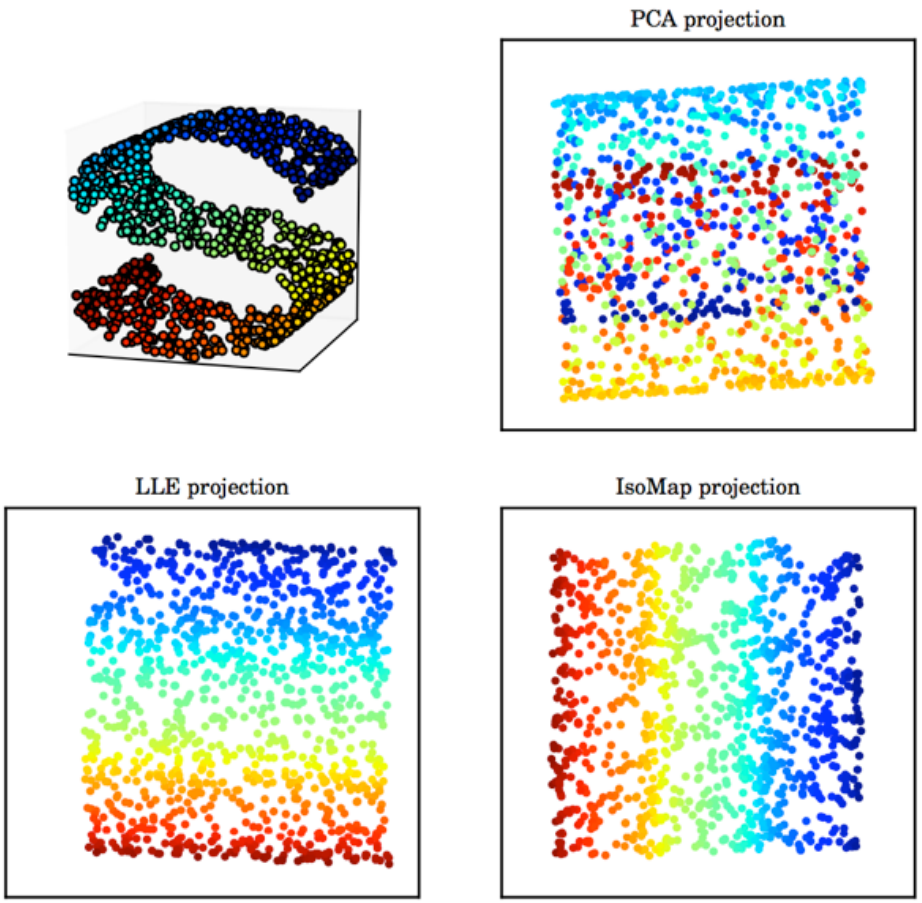


Figure 7.8.: A comparison of PCA and manifold learning. The top-left panel shows an example S-shaped data set (a two-dimensional manifold in a three-dimensional space). PCA identifies three principal components within the data. Projection onto the first two PCA components results in a mixing of the colors along the manifold. Manifold learning (LLE and IsoMap) preserves the local structure when projecting the data, preventing the mixing of the colors.

Here we have a 2D manifold in 3D space. PCA and other linear methods cannot uncover this structure.

Manifold learning techniques "unfold" or "unwrap" this manifold so that its structure becomes clear

Manifold Learning

Locally Linear Embedding

One of more popular techniques among a number of recent methods for non-linear dimensionality reduction.

In non-linear problems, it can have significantly better performance than PCA; e.g. it takes only two components to describe the same fraction of variance in SDSS galaxy spectra that requires dozens of PCA components (Vanderplas & Connolly 2009, AJ, 138, 1365)

The local manifold is determined by analyzing the nearest neighbor distribution around a given point; in some sense, one can think of a tangential hyper-plane approximation in high-D geometry

SDSS galaxy spectra

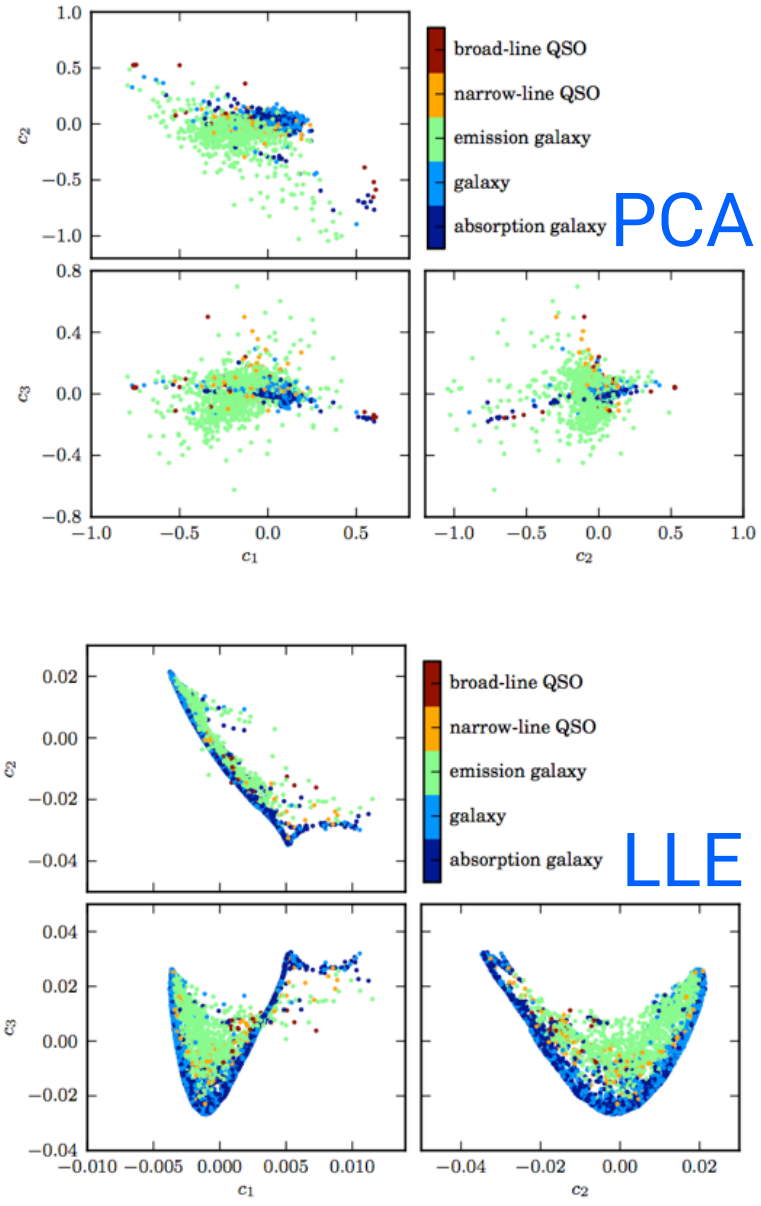


Figure 7.9.: A comparison of the classification of quiescent galaxies and sources with strong line emission using LLE and PCA. The top panel shows the segregation of galaxy types as a function of the first three PCA components. The lower panel shows the segregation using the first three LLE dimensions. The preservation of locality in LLE enables nonlinear features within a spectrum (e.g., variation in the width of an emission line) to be captured with fewer components. This results in better segregation of spectral types with fewer dimensions.

Similarly to the “S curve” example:

PCA: the structure in eigencoefficients is scrambled

LLE: the structure in eigencoefficients preserves information (from independent classification)

To reproduce: astroML, book figures, chapter 9

Which dimensionality reduction technique to use in practice?

Simple summary. Table 7.1 is a simple summary of the trade-offs along our axes of accuracy, interpretability, simplicity, and speed in dimension reduction methods, expressed in terms of high (H), medium (M), and low (L) categories.

Table 7.1.: Summary of the practical properties of the main dimensionality reduction techniques.

Method	Accuracy	Interpretability	Simplicity	Speed
Principal component analysis	H	H	H	H
Locally linear embedding	H	M	H	M
Nonnegative matrix factorization	H	H	M	M
Independent component analysis	M	M	L	L

Outline

- **Dimensionality Reduction**
  - Covariance and Correlation
  - **Principal Component Analysis**
  - Non-negative Matrix Factorization
  - Independent Component Analysis
  - Manifold learning (Locally Linear Embedding)
- Regression and a few misc. points
  - (Gaussian) **errors in both variables**
  - regression with **non-Gaussian errors and/or outliers**
  - (fast matching using KD trees)

LSQ regression and non-LSQ regression

Motivation:

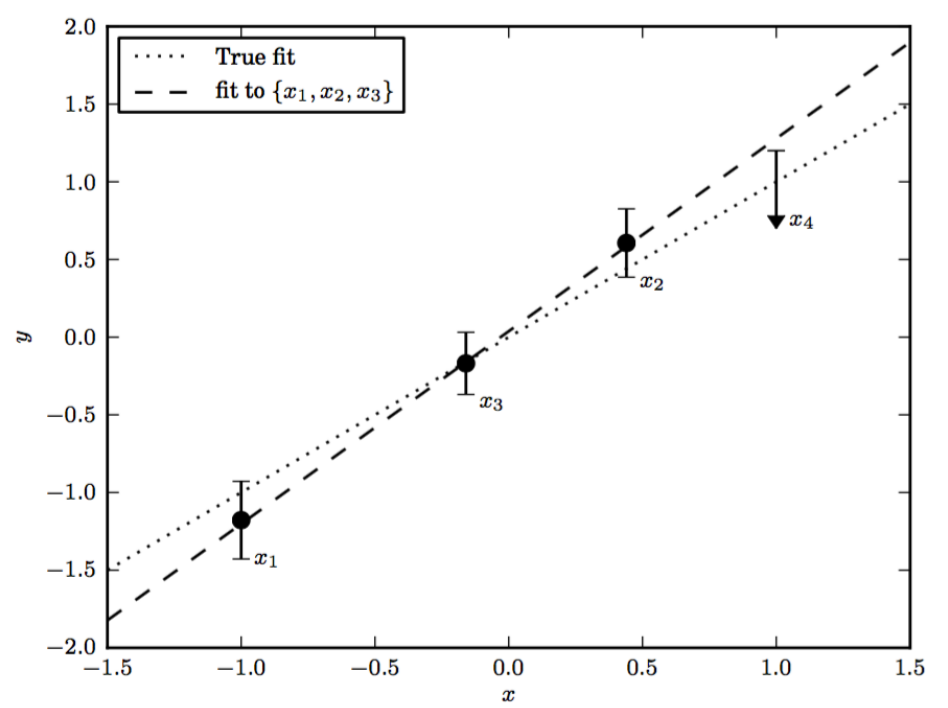
While e.g. least-square regression is often discussed, there are a few additional exceedingly useful and related tools in astroML for regression problems that often appear in practice:

– (Gaussian) errors in both variables
– regression with non-Gaussian errors and/or outliers

Recall that we addressed ordinary Least Squares Method in Week 3 lecture(on the Maximum Likelihood method)

$$p(\{y_i\}|\{x_i\},\boldsymbol{\theta},I) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(y_i-(\theta_0+\theta_1 x_i))^2}{2\sigma_i^2}\right)$$

The origin of "Least Squares": 2 comes from Gaussian errors



$$\theta_1 = \frac{\sum_i^N x_i y_i - \bar{x}\bar{y}}{\sum_i^N (x_i-\bar{x})^2},$$
$$\theta_0 = \bar{y} - \theta_1 \bar{x},$$

$$\sigma^2 = \sum_{i=1}^{N} (y_i - \theta_0 + \theta_1 x_i)^2,$$
$$\sigma_{\theta_1}^2 = \sigma^2 \frac{1}{\sum_i^N (x_i-\bar{x})^2},$$
$$\sigma_{\theta_0}^2 = \sigma^2 \left(\frac{1}{N} + \frac{\bar{x}^2}{\sum_i^N (x_i-\bar{x})^2}\right)$$

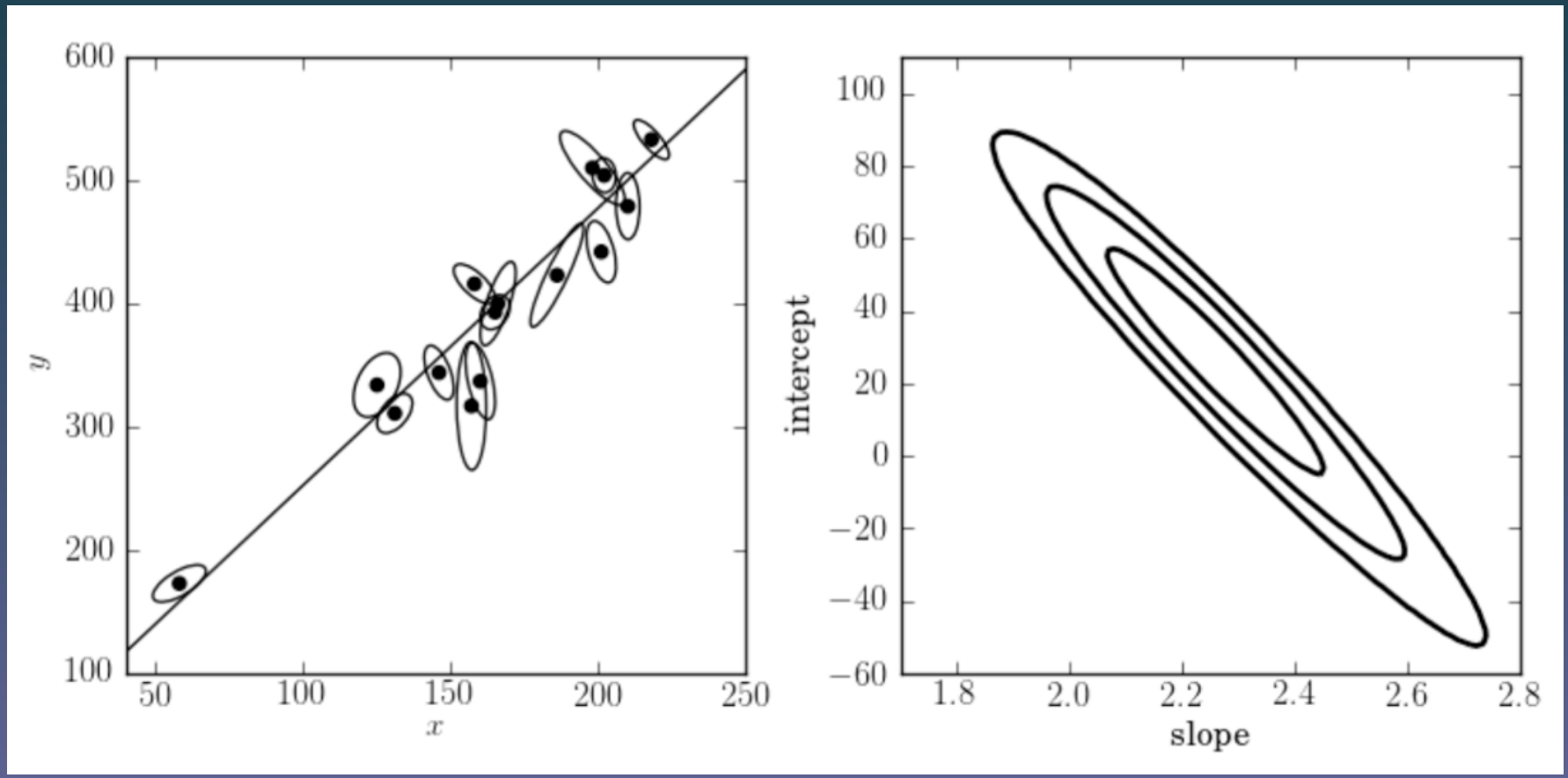
Main assumptions:
1) Gaussian errors for y
2) No errors in x

Gaussian errors in both variables

(see Hogg, Bovy & Lang, 2010, arXiv:1008.4686)

Example code:

http://www.astroml.org/book_figures/chapter8/fig_total_least_squares.html



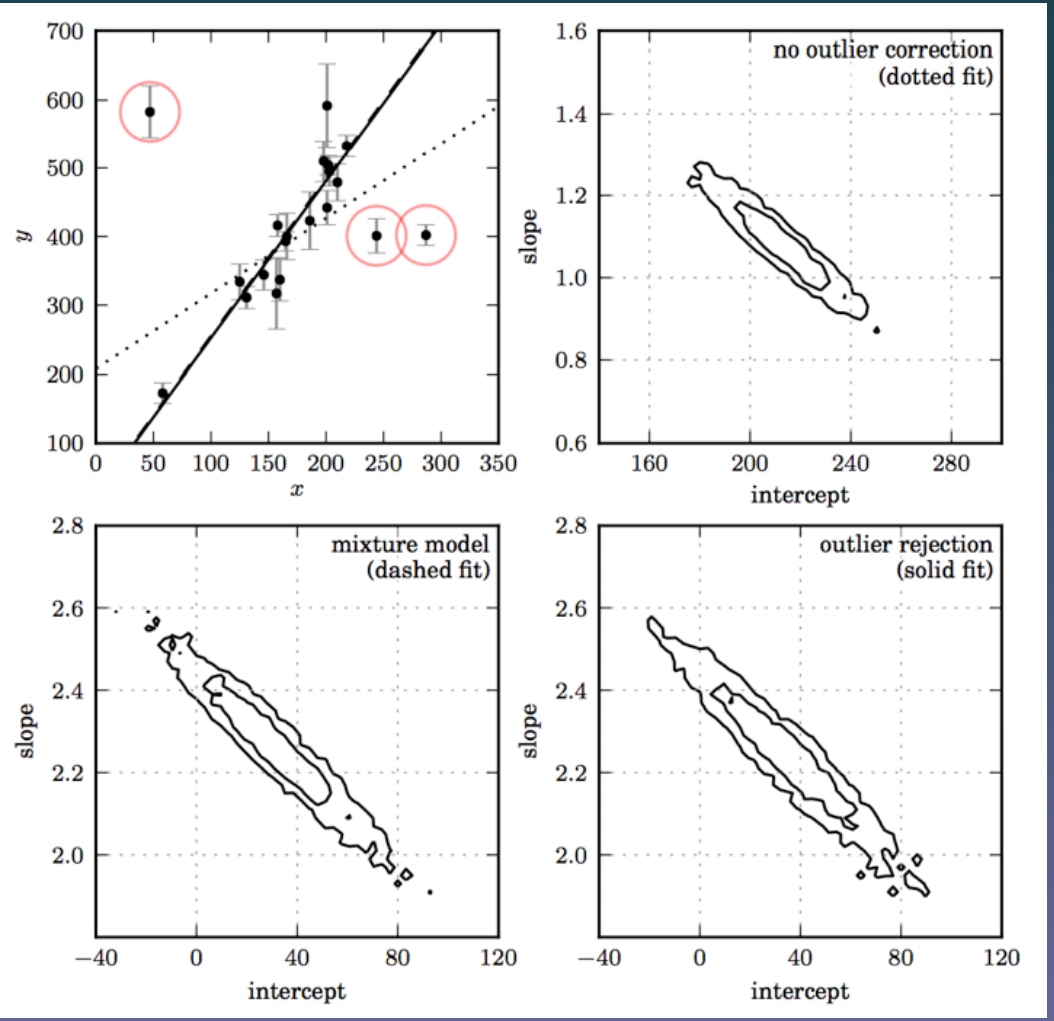
Execute the above code and then switch to notebook

Regression with non-Gaussian errors and outliers (see Hogg, Bovy & Lang, 2010, arXiv:1008.4686)

Example code:
http://www.astroml.org/book_figures/chapter8/fig_outlier_rejection.html

The code uses MCMC

It's easy to change the outlier model (the mixture likelihood)...



Regression with non-Gaussian errors and outliers

M estimators (M stands for "maximum-likelihood-type") approach the problem of outliers by modifying the underlying likelihood estimator to be less sensitive than the classic L_2 norm. M estimators are a class of estimators that include many maximum-likelihood approaches (including least squares). They replace the standard least squares, which minimizes the sum of the squares of the residuals between a data value and the model, with a different function. Ideally the M estimator has the property that it increases less than the square of the residual and has a unique minimum at zero.

Huber loss function

An example of an M estimator that is common in robust regression is that of the Huber loss (or cost) function [9]. The Huber estimator minimizes

$$\sum_{i=1}^{N} e(y_i|y), \tag{8.65}$$

where

$$e(t) = \begin{cases} \frac{1}{2}t^2 & \text{if } |t| \leq c, \\ c|t| - \frac{1}{2}c^2 & \text{if } |t| \geq c, \end{cases} \tag{8.66}$$

Regression with non-Gaussian errors and outliers

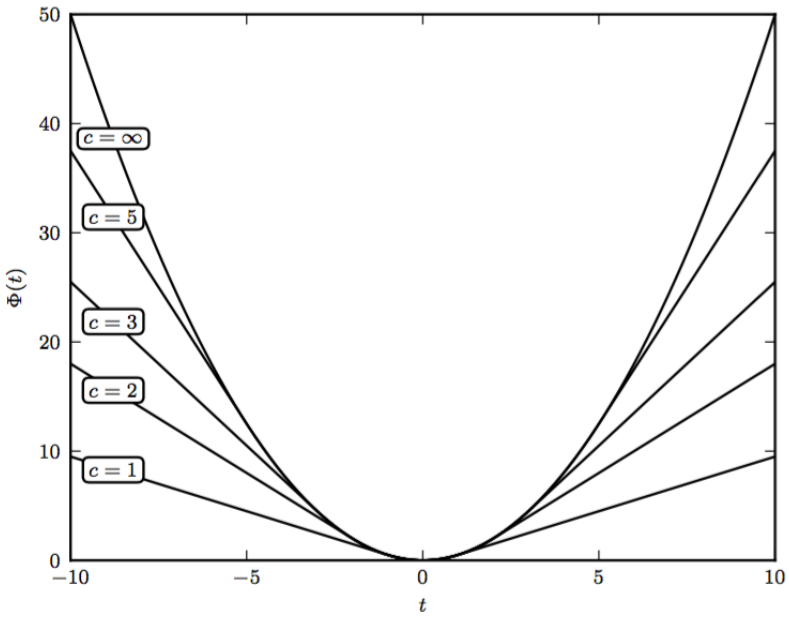


Figure 8.7.: The Huber loss function for various values of c.

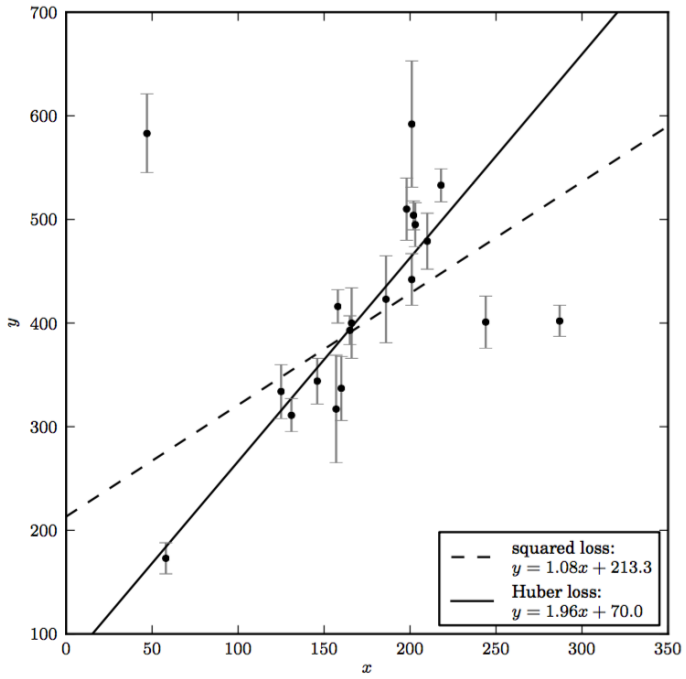


Figure 8.8.: An example of fitting a simple linear model to data which includes outliers (data is from table 1 of [8]). A comparison of linear regression using the squared-loss function (equivalent to ordinary least-squares regression) and the Huber loss function, with $c = 1$ (i.e., beyond 1 standard deviation, the loss becomes linear).

Regression with non-Gaussian errors and outliers

Bayesian mixture model:

distribution. The mixture model includes three additional parameters: μ_b and V_b, the mean and standard deviation of the background, and p_b, the probability that any point is an outlier. With this model, the likelihood becomes (cf. eq. 5.83; see also [8])

$$p(\{y_i\}|\{x_i\},\{\sigma_i\},\theta_0,\theta_1,\mu_b,V_b,p_b) \propto$$
$$\prod_{i=1}^{N}\left[\frac{1-p_b}{\sqrt{2\pi\sigma_i^2}}\exp\left(-\frac{(y_i-\theta_1 x_i-\theta_0)^2}{2\sigma_i^2}\right)+\frac{p_b}{\sqrt{2\pi(V_b+\sigma_i^2)}}\exp\left(-\frac{(y_i-\mu_b)^2}{2(V_b+\sigma_i^2)}\right)\right]. \quad (8.67)$$

Using MCMC sampling and marginalizing over the background parameters yields the dashed-line fit in figure 8.9. The marginalized posterior for this model is shown in the lower-left panel. This fit is much less affected by the outliers than is the simple regression model used above.

Finally, we can go further and perform an analysis analogous to that of §5.6.7, in which we attempt to identify bad points individually. In analogy with eq. 5.94 we can fit for nuisance parameters g_i, such that if $g_i = 1$, the point is a "good" point, and if $g_i = 0$ the point is a "bad" point. With this addition our model becomes

$$p(\{y_i\}|\{x_i\},\{\sigma_i\},\{g_i\},\theta_0,\theta_1,\mu_b,V_b) \propto$$
$$\prod_{i=1}^{N}\left[\frac{g_i}{\sqrt{2\pi\sigma_i^2}}\exp\left(-\frac{(y_i-\theta_1 x_i-\theta_0)^2}{2\sigma_i^2}\right)+\frac{1-g_i}{\sqrt{2\pi(V_b+\sigma_i^2)}}\exp\left(-\frac{(y_i-\mu_b)^2}{2(V_b+\sigma_i^2)}\right)\right]. \quad (8.68)$$

This model is very powerful: by marginalizing over all parameters but a particular g_i, we obtain a posterior estimate of whether point i is an outlier. Using this procedure, the "bad" points have been marked with a circle in the upper-left panel of figure 8.9. If instead we marginalize over the

Gaussian regression

- we don't have to use polynomials; we can use any set of basis functions
- Gaussian basis functions are often very convenient, e.g. for convolution
- see notebook for an example

The Gaussian distribution has two main properties that make it special. First, it lends itself to analytic treatment in many cases; most notably, a convolution of two Gaussian distributions is also Gaussian (it's hard to believe, but computers haven't existed forever). The convolution of a function $f(x)$ with a function $g(x)$ (both assumed real functions) is defined as

$$(f \star g)(x) = \int_{-\infty}^{\infty} f(x')\, g(x - x')\, dx' = \int_{-\infty}^{\infty} f(x - x')\, g(x')\, dx'. \tag{3.44}$$

In particular, the convolution of a Gaussian distribution $\mathcal{N}(\mu_o, \sigma_o)$ (e.g., an intrinsic distribution we are trying to measure) with a Gaussian distribution $\mathcal{N}(b, \sigma_e)$ (i.e., Gaussian error distribution with bias b and random error σ_e) produces parameters for the resulting Gaussian[4] $\mathcal{N}(\mu_C, \sigma_C)$ given by

$$\mu_C = (\mu_o + b) \quad \text{and} \quad \sigma_C = \left(\sigma_o^2 + \sigma_e^2\right)^{1/2}. \tag{3.45}$$

Similarly, the Fourier transform of a Gaussian is also a Gaussian (see §10.2.2). Another unique feature of the Gaussian distribution is that the sample mean and the sample variance are independent.

Fast matching using KD trees

Example code:
http://www.astroml.org/examples/algorithms/plot_crossmatch.html

Much faster than naive matching

Easy to use