Data Mining

Tamás Budavári - budavari@jhu.edu
Class 4

- Dependence and correlations
- Sampling from Gaussians
- Method of Least Squares

Bivariate and Multivariate

Dependence

- Consider random variables $X, Y \in \mathbb{R}$

> We can look at them separately but ...
>
> Are they "related" at all?

- Dependent variables

> $P(X,Y) \neq P(X)\,P(Y)$
>
> More on this later...

Covariance

- Definition

> $$\mathbb{C}\text{ov}[X,Y] = \mathbb{E}\Big[\big(X-\mathbb{E}[X]\big)\big(Y-\mathbb{E}[Y]\big)\Big]$$
>
> Other notations: $\mathrm{C}_{X,Y}$, $\sigma(X,Y)$, ...

- Sample covariance

> $$C = \frac{1}{N\!-\!1}\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y})$$

Quiz

1) If X and Y are independent, are they also uncorrelated?

```
[ ] Yes        [ ] No
```

2) If X and Y are uncorrelated, are they also independent?

```
[ ] Yes        [ ] No
```

Answers

1) If X and Y are independent, are they also uncorrelated?

```
[x] Yes        [ ] No
```

> Independence yields $\mathbb{E}[XY]=\mathbb{E}[X]\,\mathbb{E}[Y]$, hence the covariance
> $\mathbb{E}[(X\!-\!\mu_X)(Y\!-\!\mu_Y)] = \mathbb{E}[XY - X\mu_y - \mu_X Y + \mu_X\mu_Y] = 0$

2) If X and Y are uncorrelated, are they also independent?

```
[ ] Yes        [x] No
```

> For example, let random variable X have a normal distribution, $X \sim \mathcal{N}(0,1)$, and let $Y = X^2$. They are clearly dependent but are they correlated?
>
> $$\mathbb{E}[(X-0)(X^2-\mu_{X^2})] = \mathbb{E}[X^3 - X\,\mu_{X^2}] = \mathbb{E}[X^3] - \mathbb{E}[X]\,\mu_{X^2} = 0 - 0$$

More examples

Vector Notation

- Let $\boldsymbol{V}$ represent the 2-vector of random scalar variables X and Y

> $$\boldsymbol{V} = \begin{pmatrix} X \\ Y \end{pmatrix}$$

- Mean

> $$\mathbb{E}[\boldsymbol{V}] = \begin{pmatrix} \mathbb{E}[X] \\ \mathbb{E}[Y] \end{pmatrix} = \begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}$$

- Covariance matrix

> $$\Sigma(\boldsymbol{V}) = \mathbb{E}\Big[\big(\boldsymbol{V}-\mathbb{E}[\boldsymbol{V}]\big)\big(\boldsymbol{V}-\mathbb{E}[\boldsymbol{V}]\big)^T\Big] = \begin{pmatrix} \sigma_X^2 & \mathrm{C}_{X,Y} \\ \mathrm{C}_{X,Y} & \sigma_Y^2 \end{pmatrix}$$
>
> Same generalization of variance works in any dimensions

Bivariate Normal Distribution

- Independent and uncorrelated

> $$\mathcal{N}(x,y;\mu_x,\mu_y,\sigma_x,\sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y}\ \exp\left[-\frac{(x-\mu_x)^2}{2\sigma_x^2} - \frac{(y-\mu_y)^2}{2\sigma_y^2}\right]$$

- In general for 2-vector $\boldsymbol{x}$

> $$\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\Sigma) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}}\ \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$
>
> where $|\Sigma|$ is the determinant - other notation $\det\Sigma$ or $\det(\Sigma)$

- Uncorrelated if

> $$\Sigma = \begin{pmatrix}\sigma_X^2 & 0 \\ 0 & \sigma_Y^2\end{pmatrix}$$

Multivariate Normal Distribution

- In k dimensions - not bold but k-vectors

> $$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^k|\Sigma|}}\ \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

Sampling from Gaussians

- Uncorrelated $\mathcal{N}(0,I)$: Box-Muller transform

> Using 2 uniform randoms between 0 and 1
>
> $$Z_1 = \sqrt{-2\ln U_1}\ \cos\left(2\pi U_2\right)$$
>
> $$Z_2 = \sqrt{-2\ln U_1}\ \sin\left(2\pi U_2\right)$$

- Tranform: scale, rotate, shift

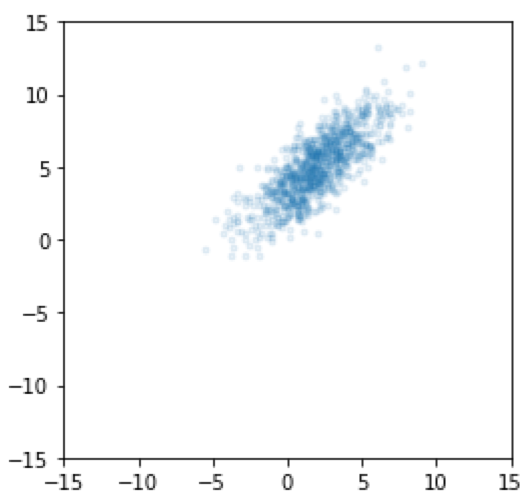
In [1]:
```
%pylab inline
from scipy.stats import norm as gaussian
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
# generate many 2D (column) vectors
X = gaussian.rvs(0,1,(2,1000))
X[0,:] *= 3  # scale axis 0
f = +pi/4    # rotate by f
R = array([[cos(f),-sin(f)],
           [sin(f), cos(f)]])
V = R.dot(X)
V += np.array([[2],
               [5]]) # shift with a vector
# plot on square figure
figure(figsize=(4,4)); a=15; xlim(-a,a); ylim(-a,a)
plot(V[0,:],V[1,:], '.', alpha=0.1)

# sample covariance matrix
averages = mean(V, axis=1)
print (averages.shape)
averages
```

```
(2,)
```

Out[2]:
```
array([1.92651401, 5.00789999])
```



In [3]:
```
#avg = averages.reshape(averages.size,1)
avg = averages[:,np.newaxis]
print ("Average: ")
print (avg)
print ("Cov:")
print (np.dot(V-avg, (V-avg).T) / (V[0,:].size-1))
```

```
Average:
[[1.92651401]
 [5.00789999]]
Cov:
[[5.07365656 3.92119892]
 [3.92119892 4.95435205]]
```

Method of Least Squares

The Idea

- Fit a model to training set $\left\{(x_i, y_i)\right\}$

> Parameterized function $f(x;\theta)$, where θ can represent multiple parameters

- Minimize the mean or sum of square errors or residuals (SSE, SSR, MSE, MSR?)

> Residual
> $r_i(\theta) = y_i - f(x_i;\theta)$
>
> Estimation
> $\hat{\theta} = \arg\min\limits_{\theta} \sum\limits_i \big[y_i - f(x_i;\theta)\big]^2$

- Optimization is simple for certain models

The Simplest Case

- Fitting a constant? Model with $f(x;\mu) = \mu$

> $$C(\mu) = \sum_{i=1}^{N} \big(y_i - \mu\big)^2$$

- Derivative $C' = dC\big/d\mu$ vanishes at solution $\hat{\mu}$

> $C'(\hat{\mu}) = 0$
>
> $$2\sum_{i=1}^{N} \big(y_i - \hat{\mu}\big)(-1) = 0$$
>
> $$\sum_{i=1}^{N} y_i - N\hat{\mu} = 0$$
>
> $$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} y_i$$ - average

Heteroscedasticity

- Same model with $f(x;\mu)=\mu$

> $$C(\mu)=\sum_{i=1}^{N}\frac{\left(y_i-\mu\right)^2}{\sigma_i^2}$$
>
> with $w_i=1\big/\sigma_i^2$
>
> $$C(\mu)=\sum_{i=1}^{N}w_i\left(y_i-\mu\right)^2$$

- Derivative $C'=dC\big/d\mu$ vanishes at $\hat{\mu}$

> $$C'(\hat{\mu})=0$$
>
> $$2\sum_i w_i\left(y_i-\hat{\mu}\right)(-1)=0$$
>
> $$\sum_i w_i y_i-\hat{\mu}\sum_i w_i=0$$
>
> $$\hat{\mu}=\frac{\sum w_i y_i}{\sum w_i}$$ - weighted average

Simple Fitting

- A linear model with $\boldsymbol{\theta}=(a,b)^T$ parametrization $f(x;\boldsymbol{\theta})=a+b\,x$

> $$\hat{\boldsymbol{\theta}}=\arg\min\sum_i\left[y_i-(a+b\,x_i)\right]^2$$

- Derivatives w.r.t. a and b should vanish

> We have 2 variables and 2 equations
>
> Quadratic becomes linear $\rightarrow$ analytic solution!

Unhomework

1. Derive the best fit parameters of (a,b)

Linear Regression

- A linear combination of known $\phi_k(\cdot)$ functions (basis functions)

$$f(x;\boldsymbol{\beta}) = \sum_{k=1}^{K} \beta_k \, \phi_k(x)$$

It's a dot product

$$f(x;\boldsymbol{\beta}) = \boldsymbol{\beta}^T \boldsymbol{\phi}(x)$$

with $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_K)^T$

- Linear in $\boldsymbol{\beta}$, cost function is quadratic

$$C = \sum_{i=1}^{N} \left\{ y_i - \sum_{k=1}^{K} \beta_k \, \phi_k(x_i) \right\}^2$$

- Introducing matrix X with components

$$X_{ik} = \phi_k(x_i)$$

- Linear in $\boldsymbol{\beta}$, cost function is quadratic

$$C = \sum_{i=1}^{N} \left\{ y_i - \sum_{k=1}^{K} X_{ik} \beta_k \right\}^2$$

Minimization

- Partial derivatives

$$\frac{\partial C}{\partial \beta_l} = 2 \sum_i \left\{ y_i - \sum_{k=1}^{K} X_{ik} \beta_k \right\} \left[- \frac{\partial f(x_i; \boldsymbol{\beta})}{\partial \beta_l} \right]$$

and

$$\frac{\partial f(x_i; \boldsymbol{\beta})}{\partial \beta_l} = \sum_k \frac{\partial \beta_k}{\partial \beta_l} \, \phi_k(x_i) = \phi_l(x_i) = X_{il}$$

Note: $\partial \beta_k \big/ \partial \beta_l = \delta_{kl}$ Kronecker delta

Detour: The Kronecker Delta

- Definition

$$\delta_{kl} = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{if } k \neq l \end{cases}$$

- Useful to remember

$$\sum_l \delta_{kl} \, a_l = a_k$$

Cf. identity matrix: $\boldsymbol{I} \, \boldsymbol{a} = \boldsymbol{a}$

Result

- At the optimum we have

$$\sum_i \left\{ y_i - \sum_k \hat{\beta}_k \, \phi_k(x_i) \right\} \, \phi_l(x_i) = 0$$

$$\sum_i \left\{ y_i - \sum_k X_{ik} \hat{\beta}_k \right\} \, X_{il} = 0$$

$$\sum_i X_{il} y_i - \sum_i \sum_k X_{il} X_{ik} \hat{\beta}_k = 0$$

$$\sum_i X_{il} y_i = \sum_k \left(\sum_i X_{il} X_{ik} \right) \hat{\beta}_k$$

- I.e.,

$$X^T y = X^T X \hat{\beta}$$

$$\hat{\beta} = (X^T X)^{-1} X^T y = X^+ y$$

- See **Moore-Penrose pseudoinverse**, **generalized inverse**
- See also **Singular Value Decomposition**

Hat matrix

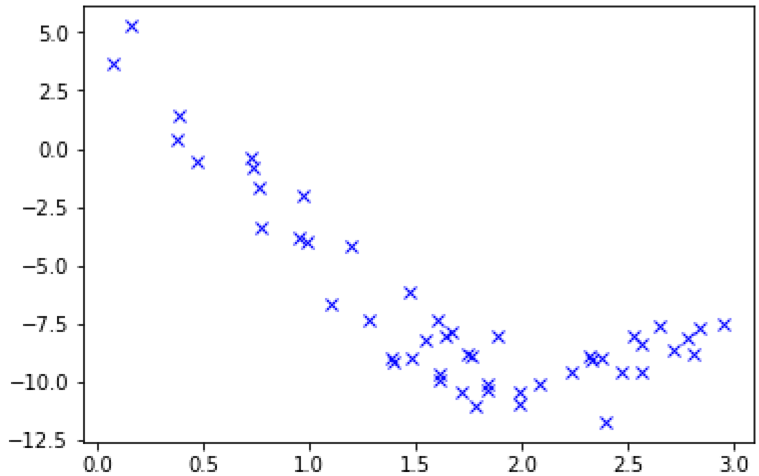
- Looking at the definition of X we see that the model at $\hat{\beta}$ predicts $\hat{y}_i$ values

$$\hat{y} = X \, \hat{\beta} = X \, (X^T X)^{-1} X^T y$$

which is

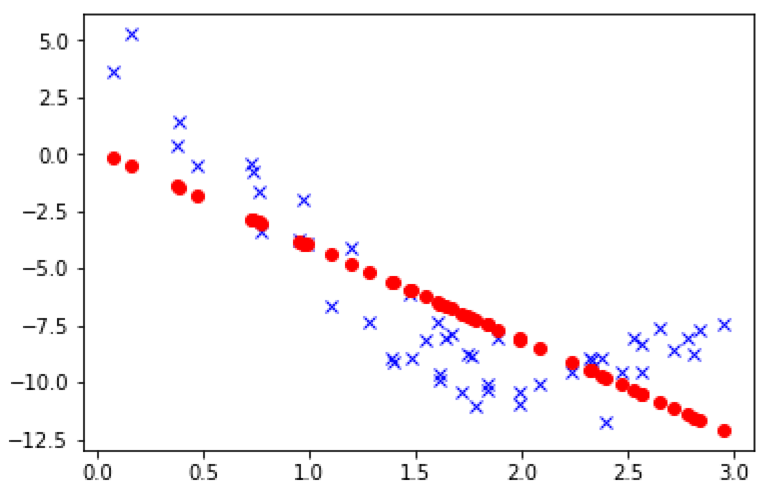
$$\hat{y} = H \, y \quad \text{with} \quad H = X \, (X^T X)^{-1} X^T$$

In [4]:
```
# generate sample with error
x = 3 * random.rand(50) # between 0 and 3
e = 1 * random.randn(x.size) # noise
#y = (0.1*x**3 + 0.5*x**2 + 2*x + 1) + e;  plot(x,y,'bo');
y = 10*cos(x+1) + e;  plot(x,y,'bx');
```



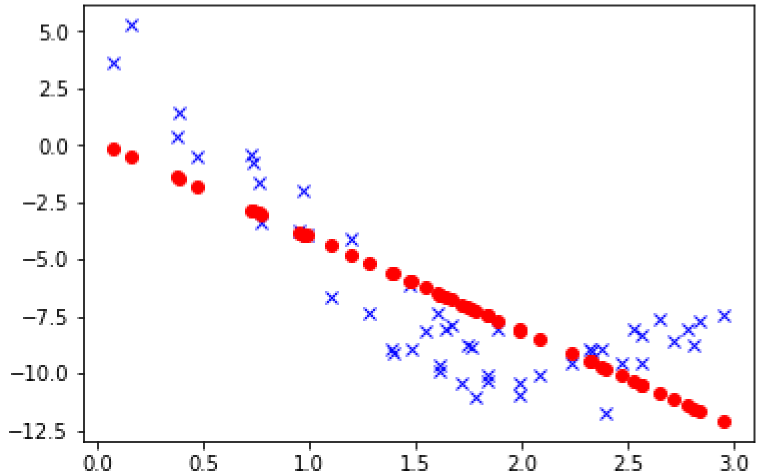
In [5]:
```
# linear model f(x) = b0 + b1 x
X = ones((x.size,2));
X[:,1] = x

Xpinv = dot(inv(dot(X.T,X)),X.T)
bHat = dot(Xpinv,y)
yHat = dot(X,bHat)

plot(x,y,'bx'); plot(x,yHat,'ro'); bHat
```

Out[5]: array([0.16613221, -4.15910278])

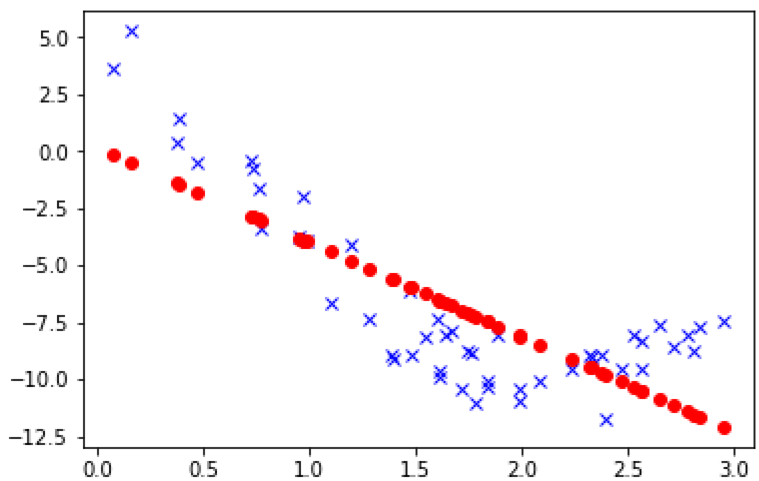


In [6]:
```
# same using methods
Xpinv = inv(X.T.dot(X)).dot(X.T)
bHat = Xpinv.dot(y)
yHat = X.dot(bHat)

plot(x,y,'bx'); plot(x,yHat,'ro'); bHat
```

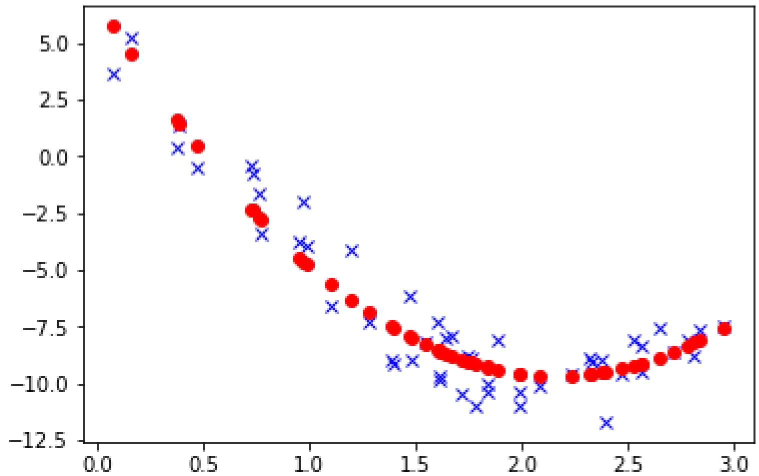
Out[6]: array([0.16613221, -4.15910278])



In [7]:
```
# same again with pinv() and the Hat matrix
H = X.dot(linalg.pinv(X))
yHat = H.dot(y)

plot(x,y,'bx'); plot(x,yHat,'ro');
```



In [8]:
```
# linear model f(x) = b0 + b1 x + b2 * x^2
X = ones( (x.size,3));
X[:,1] = x # partials wrt. b1
X[:,2] = x*x # wrt. b2

# sames as before
bHatQ = linalg.pinv(X).dot(y)
yHatQ = X.dot(bHatQ)

# or like this
H = dot(X,linalg.pinv(X))
yHatQ = dot(H,y)

plot(x,y,'bx'); plot(x,yHatQ,'ro');
```



Unhomework

1. Fit a 3rd order polynomial to the same data
2. Fit $f(x; \beta_0, \beta_1) = \beta_0 \sin(x) + \beta_1 \cos(x)$
3. Evaluate the best fits on a grid of 1000 equally-spaced points in $[-1, 4]$
4. Plot them in one figure

Heteroscedastic error

- Simple modification

> $$C = \sum_{i=1}^{N} w_i \left\{ y_i - \sum_{k=1}^{K} X_{ik}\beta_k \right\}^2$$
>
> yields
>
> $$\sum_i w_i \left\{ y_i - \sum_k X_{ik}\hat{\beta}_k \right\} X_{il} = 0$$
>
> $$\sum_i X_{il} w_i y_i = \sum_k \left(\sum_i X_{il} w_i X_{ik} \right) \hat{\beta}_k$$

- Diagonal weight matrix W

> $$X^T W y = X^T W X \hat{\beta}$$
>
> $$\hat{\beta} = (X^T W X)^{-1} X^T W y$$