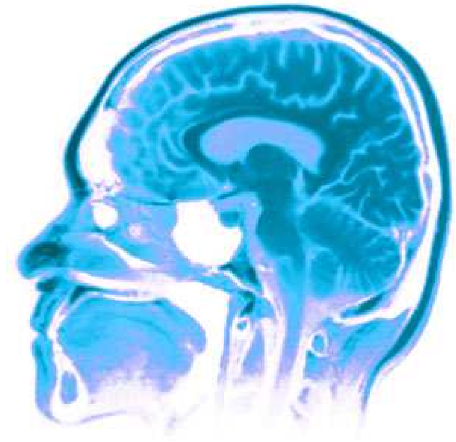
CPSC540



Linear regression

Nando de Freitas

2011

KPM Book Sections: *1.3, 1.7.1, 1.7.2 and 3.3*

Regression

- Regression is just like classification except the response variable is continuous, $y \in \mathbb{R}$.
- To make the output y depend on the input $\mathbf{x}$, we can write

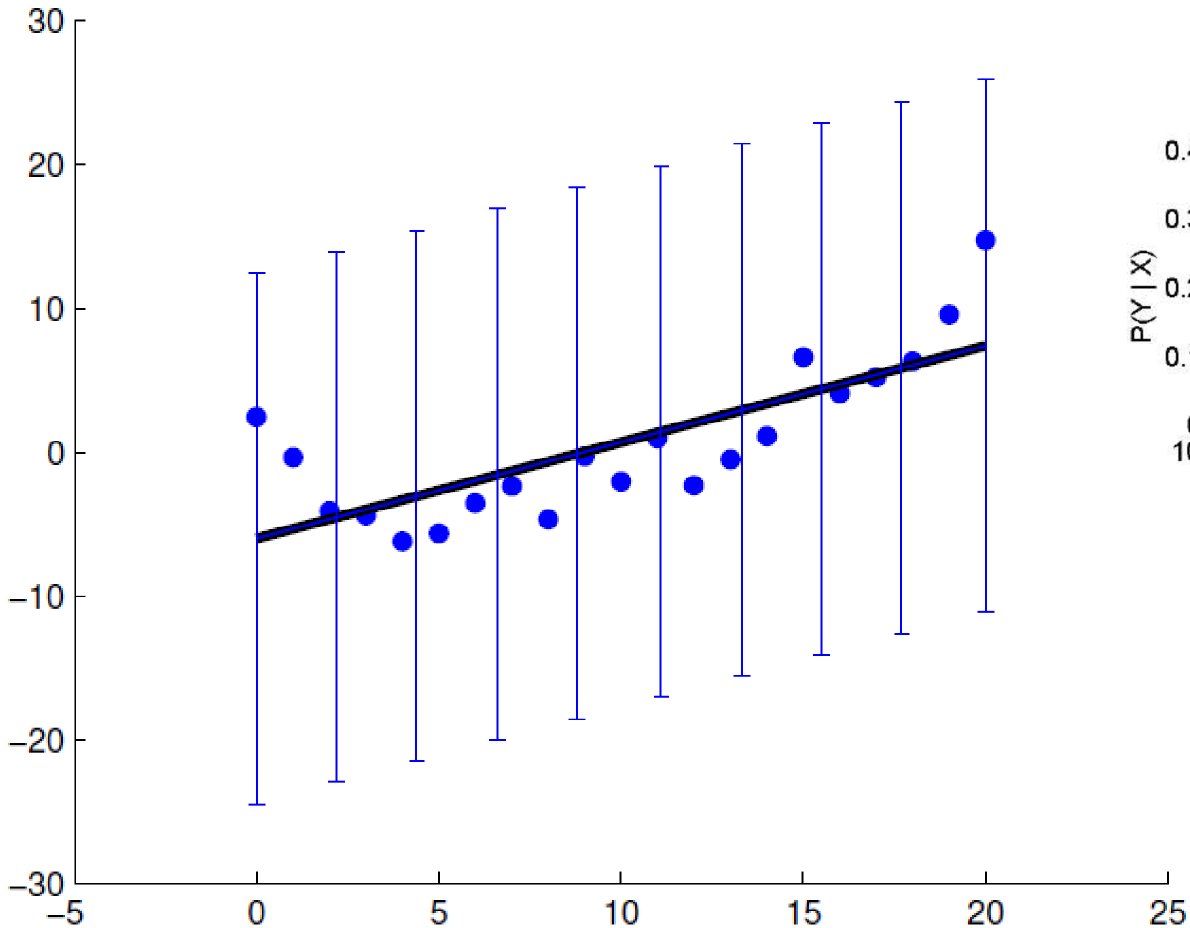
$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$$

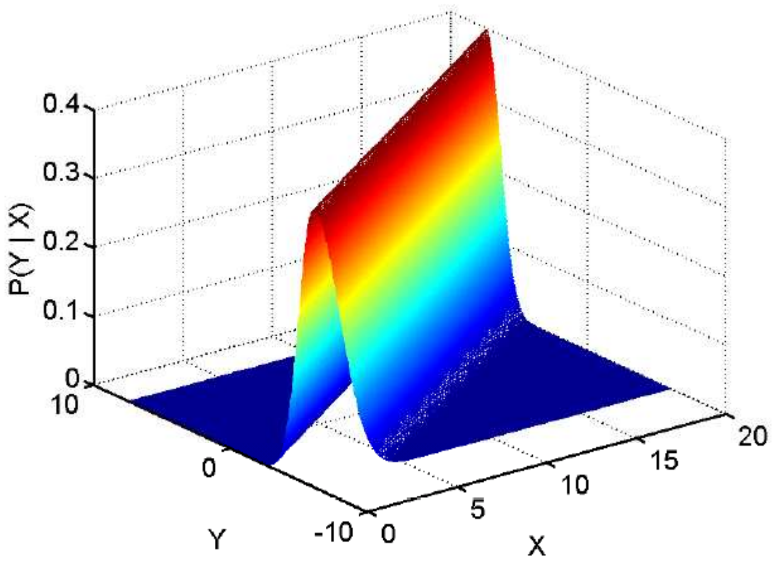
  In the simplest case, we assume μ is a linear function of $\mathbf{x}$, so $\mu = \mathbf{w}^T\mathbf{x}$, and that the noise is fixed, $\sigma^2(x) = \sigma^2$. This model is called **linear regression**.
- It can be equivalently written in the following form:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \epsilon$$

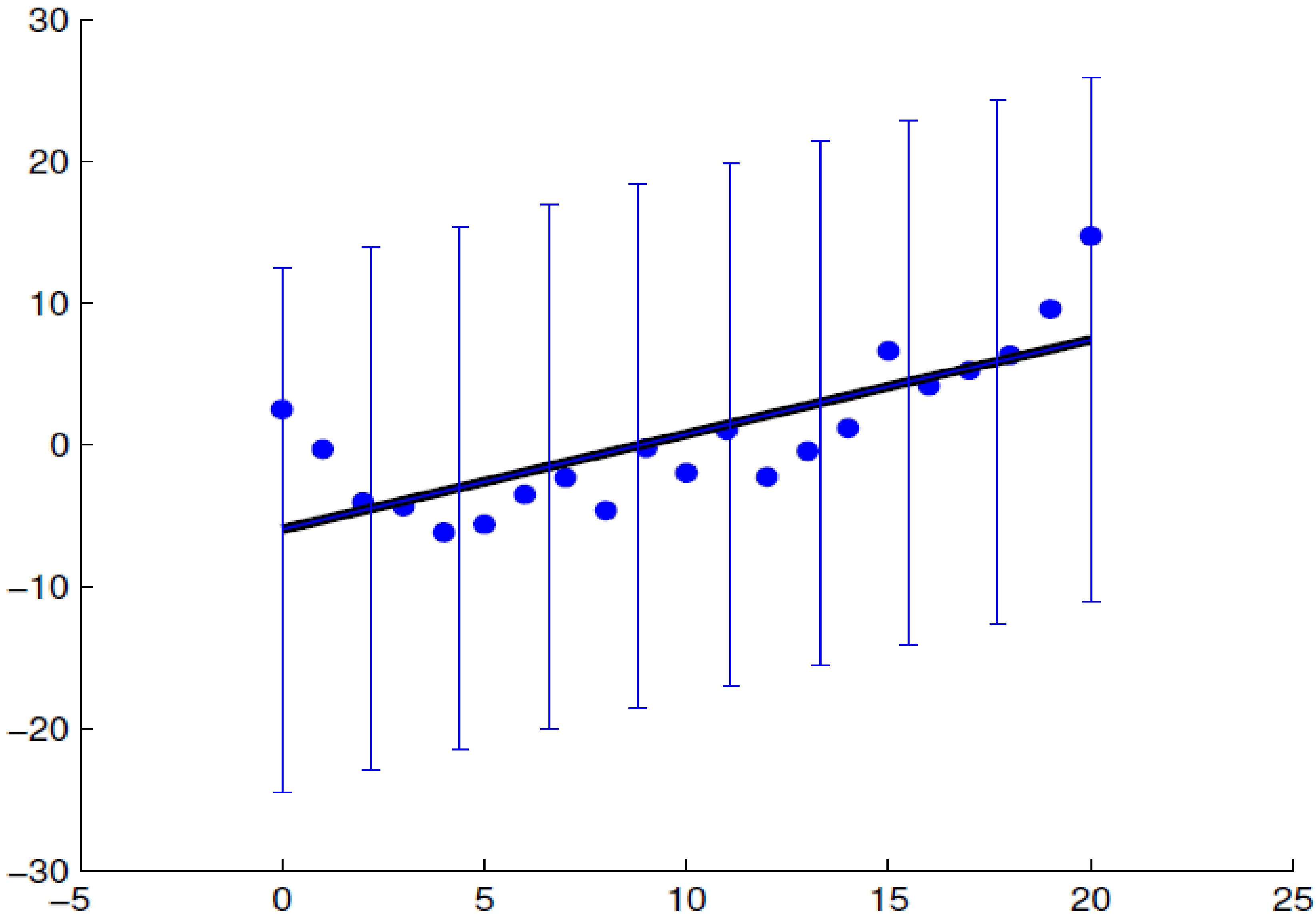
  where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the **residual error** between our linear predictions and the true response.

Linear regression

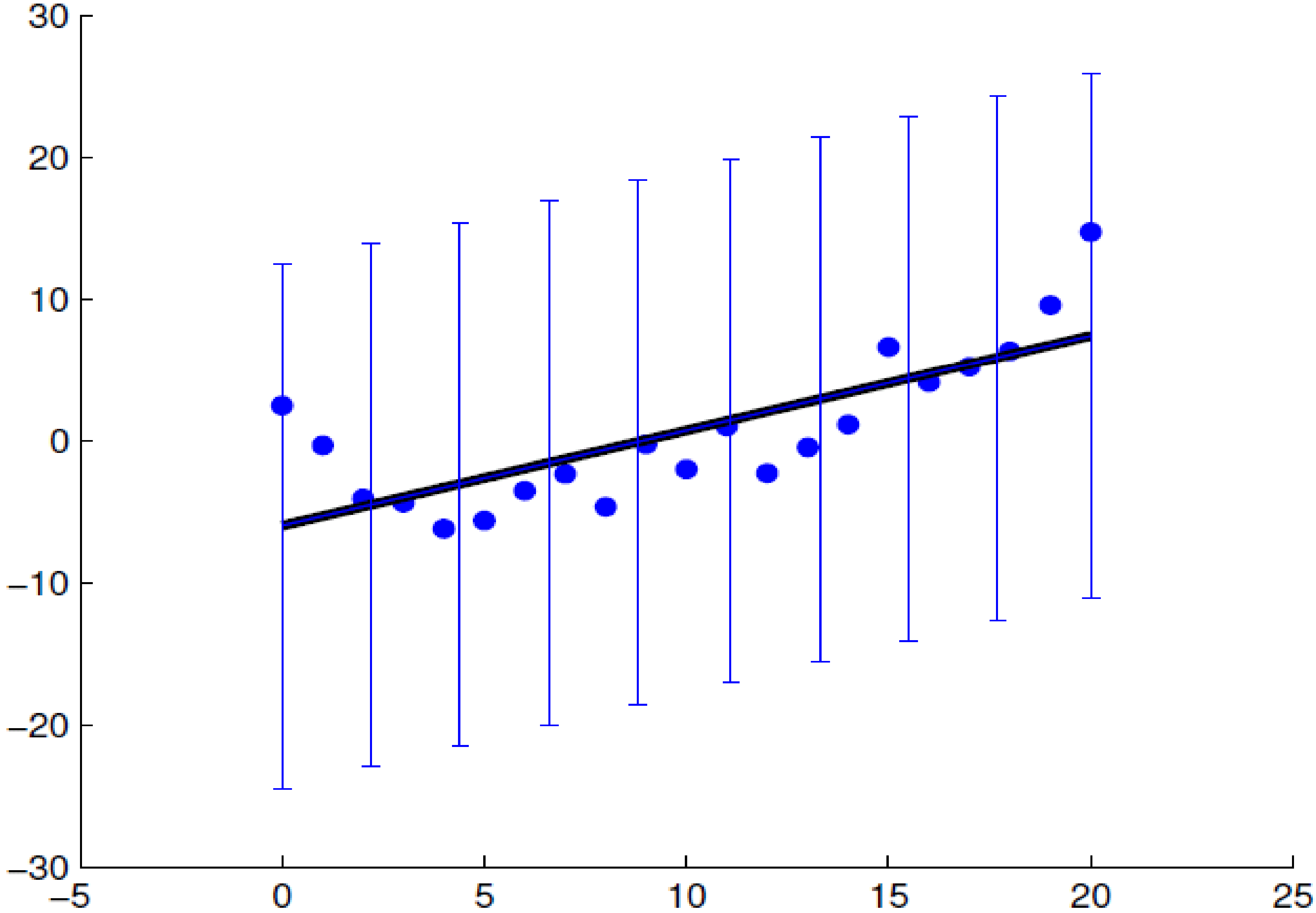




Linear regression



Linear regression

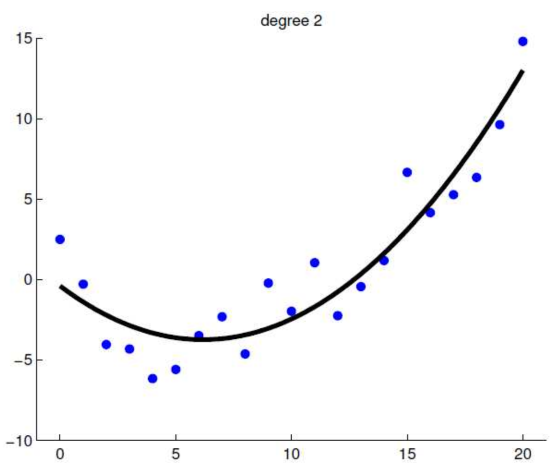


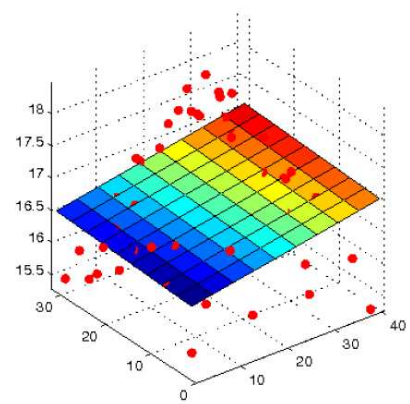
Basis functions

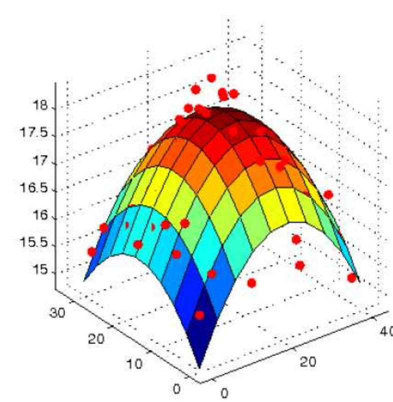
- Here, we can also introduce basis functions to deal with nonlinearity:

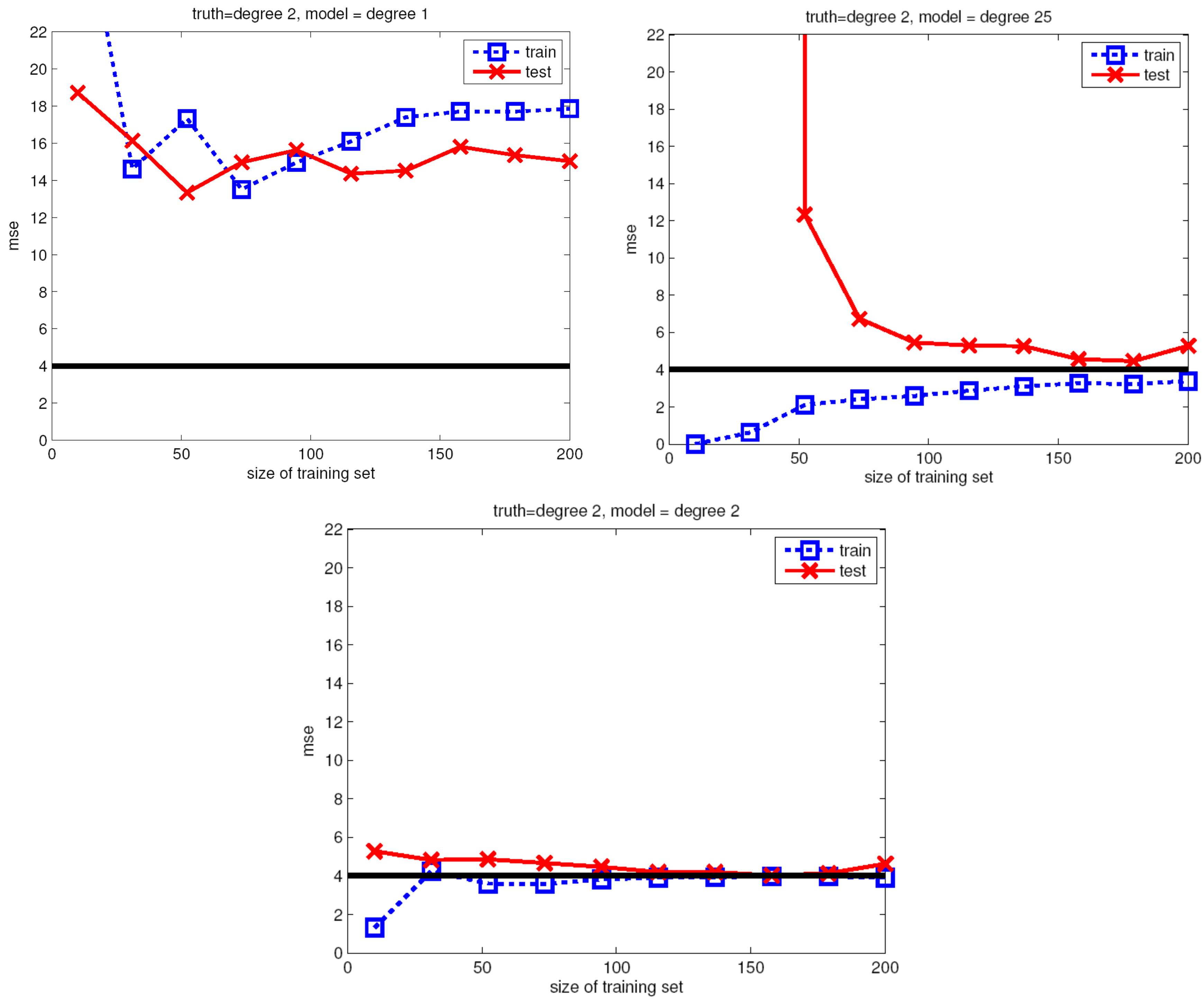
$$y(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + \epsilon$$

For example, $\boldsymbol{\phi}(x) = [1, x, x^2]$, $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2]$ or $\boldsymbol{\phi}(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2]$.





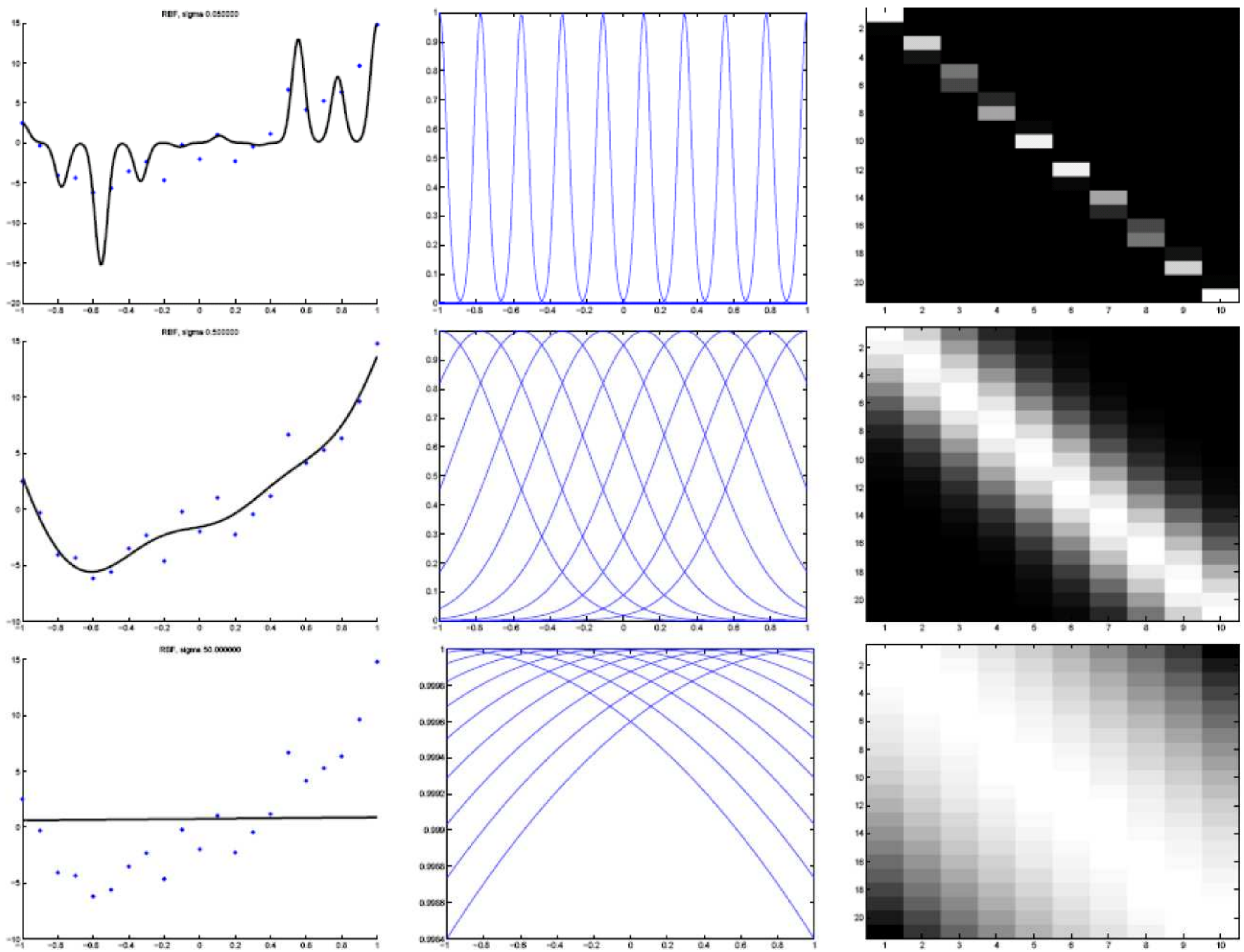




truth=degree 2, model = degree 1
mse
size of training set
train
test
truth=degree 2, model = degree 25
mse
size of training set
train
test
truth=degree 2, model = degree 2
mse
size of training set
train
test

Kernel regression

- Another way to perform nonlinear regression is to use kernels to define the basis functions, $\boldsymbol{\phi}(\mathbf{x}) = [\kappa(\mathbf{x}, \boldsymbol{\mu}_1), \ldots, \kappa(\mathbf{x}, \boldsymbol{\mu}_{D'})]$.



Negative log likelihood

- The negative log likelihood for linear regression can be written as follows:

$$
\begin{aligned}
NLL(\mathbf{w}) &= -\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \mathbf{w}) = \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \mathbf{w})^2 + \text{const} \\
&= ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \text{const} = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \text{const}
\end{aligned}
$$

MSE

- We can equivalently minimize

$$\text{mse}(\mathbf{w}) = \frac{1}{2N}\mathbf{w}^T\underbrace{\left(\mathbf{X}^T\mathbf{X}\right)}_{\mathbf{A}}\mathbf{w} - \frac{1}{N}\mathbf{w}^T\underbrace{\left(\mathbf{X}^T\mathbf{y}\right)}_{\mathbf{a}}$$

where

$$\mathbf{X}^T\mathbf{X} = \sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T = \sum_{i=1}^{N}\begin{pmatrix} x_{i,1}^2 & \cdots & x_{i,1}x_{i,D} \\ & \ddots & \\ x_{i,D}x_{i,1} & \cdots & x_{i,D}^2 \end{pmatrix}$$

is the **sum of squares** matrix and

$$\mathbf{X}^T\mathbf{y} = \sum_{i=1}^{N}\mathbf{x}_i y_i$$

Gradient and Hessian

- The gradient and Hessian are given by

$$\nabla_{\mathbf{w}} \text{mse}(\mathbf{w}) = \frac{1}{N}[\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}] = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i(\mathbf{w}^T\mathbf{x}_i - y_i)$$

$$\nabla^2_{\mathbf{w}} \text{mse}(\mathbf{w}) = \nabla_{\mathbf{w}}(\nabla_{\mathbf{w}} \text{mse}(\mathbf{w})^T) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i\mathbf{x}_i^T = \frac{1}{N}\mathbf{X}^T\mathbf{X}$$

The Hessian is positive definite (assuming $\mathbf{X}$ is full rank).

MLE = OLS

- The MSE has a unique global minimum. We can solve for this analytically by equating the gradient to zero:

$$\begin{aligned}\nabla_{\mathbf{w}}\text{mse}(\mathbf{w}) &= \mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} = \mathbf{0} \\ \mathbf{X}^T\mathbf{X}\hat{\mathbf{w}} &= \mathbf{X}^T\mathbf{y} \\ \hat{\mathbf{w}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$

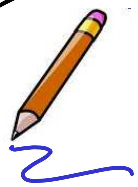
- The solution, $\hat{\mathbf{w}}$, is called the **ordinary least squares** or **OLS** solution.

MLE of the variance

- Once we have found the ML estimate of the weights, $\hat{\mathbf{w}}$, we can easily find the ML estimate for the variance by solving $\frac{\partial}{\partial \sigma^2} NLL(\hat{\mathbf{w}}, \sigma^2) = 0$ to get

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i^T \hat{\mathbf{w}})^2$$

- This is just the empirical variance of the residual errors when we "plug in" our estimate of $\hat{\mathbf{w}}$.



Weighted least squares

- Sometimes some measurements are more reliable than others; this is called **heteroscedastic data**.

- We can model this by assigning a different precision (inverse variance) $\lambda_i = \sigma_i^{-2}$ to each data point; these act as weighting terms.

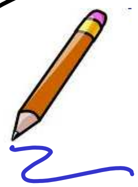
- The new negative log-likelihood is given by the following (up to irrelevant constants):

$$J(\mathbf{w}) = \sum_{i=1}^{N} \lambda_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

- It is easy to show that the corresponding MLE is given by

$$\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{\Lambda} \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{\Lambda} \mathbf{y})$$

  where $\mathbf{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$ is a diagonal matrix of precisions. This is known as **weighted least squares**.



Multivariate linear regression

- **Multivariate linear regression**, also called **multiple-output linear regression**, is just like "regular" linear regression, except the output is a vector. Hence we replace the weight vector with a weight matrix:

  $$\mathbf{y}_i = \mathbf{W}^T \mathbf{x}_i + \boldsymbol{\epsilon}_i$$

  where $\mathbf{x}_i$ is a column vector of D_x inputs (covariates), $\mathbf{y}_i$ is a column vector of D_y outputs (responses), $\mathbf{W}$ is a $D_x \times D_y$ weight matrix (so we have one column per output), and $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$.

- In matrix notation, we have

  $$\mathbf{Y}_{(N \times D_y)} = \mathbf{X}_{(N \times D_x)} \mathbf{W}_{(D_x \times D_y)} + \boldsymbol{\epsilon}_{(N \times D_y)}$$

Multivariate linear regression

- If $\mathbf{\Sigma} = \text{diag}(\sigma_j)$ is diagonal, we can compute the MLE for each column of $\mathbf{W}$ separately. To see why, let $\mathbf{w}_{:j}$ be the j'th column of $\mathbf{W}$, and $\mathbf{y}_{:j}$ be the j'th column of $\mathbf{Y}$. The NLL cost function is given by

$$J(\mathbf{W}) \quad = \quad \sum_{i=1}^{N} \sum_{j=1}^{D} \frac{1}{2\sigma_j^2} (y_{i,j} - \mathbf{w}_j^T \mathbf{x}_i)^2 = \sum_{j=1}^{D} \frac{1}{2\sigma_j^2} ||\mathbf{X}\mathbf{w}_{:j} - \mathbf{y}_{:j}||_2^2$$

  Hence $J(\mathbf{W})$ decomposes into separate problems, one per column. The σ_j terms will cancel out when we take derivatives to yield

$$\hat{\mathbf{w}}_{:j} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}_{:j}$$

  Concatentating the solutions gives

$$\hat{\mathbf{W}} = (\hat{\mathbf{w}}_{:1} \ldots \hat{\mathbf{w}}_{:D_y}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y}_{:1} \ldots \mathbf{y}_{:D_y}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

Multivariate linear regression

- Multiple-output linear regression is an example of **multi-task learning**. Of course, if we fit each response separately, we are solving each "task" separately, and we do not get any benefit from having multiple problems to solve.

- We can use hierarchical Bayesian methods to "borrow statistical strength" from easy tasks to help learn hard tasks; this can reduce the amount of training data we need to fit the model.

PMTK – linear regression

- ```
  % Matrix method
  Xtrain1 = [ones(size(Xtrain,1),1) Xtrain];
  w = Xtrain1 \ ytrain;

  % Scalar method
  xbar = mean(xtrain); ybar = mean(ytrain); N = length(ytrain);
  w1 = sum( (xtrain-xbar) .* (ytrain-ybar) ) / sum( (xtrain-xbar).^2 );
  w0 = ybar - w1*xbar;
  assert(approxeq([w0 w1], w))

  % Predict
  Xtest1 = [ones(size(Xtest,1),1) Xtest];
  ypredTest = Xtest1*w;
  ```
  ```

PMTK – logistic regression

- ```
  winit = randn(D,1);
  options.Display = 'none';
  [wMLE] = minFunc(@(w)LogisticLossSimple(w,X,y), winit, options);
  ```

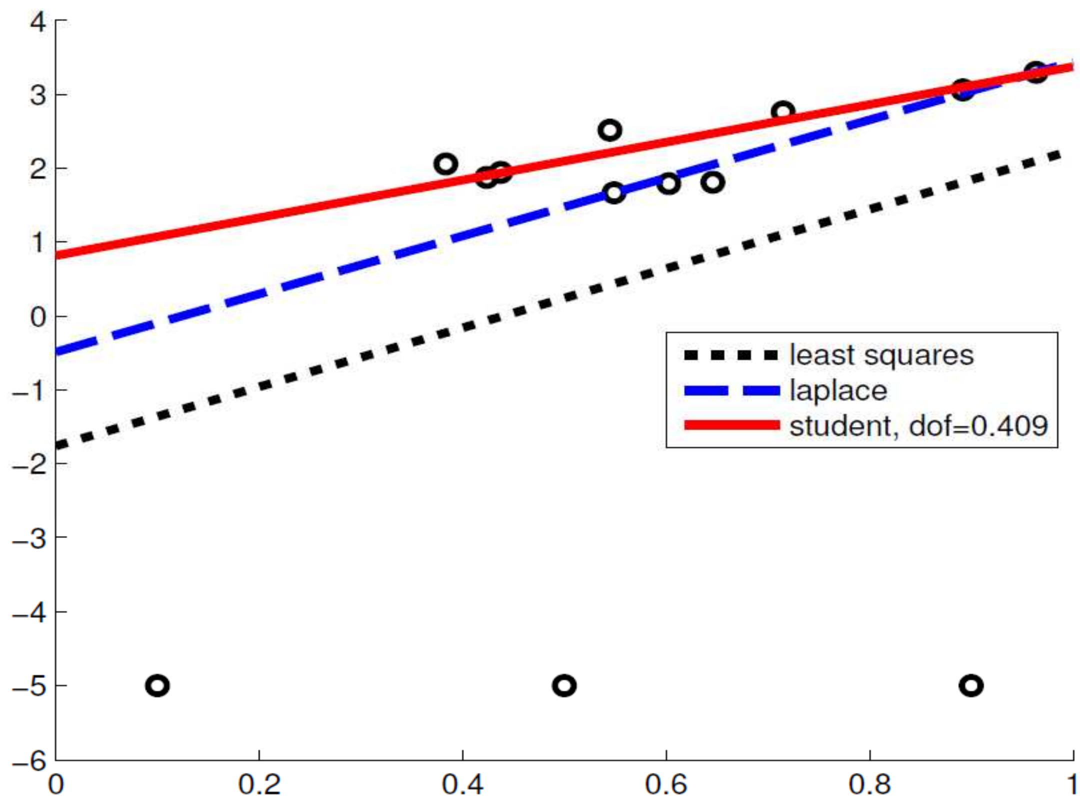
- ```
  function [nll,g,H] = LogisticLossSimple(w,X,y)
  % Negative log likelihood for binary logistic regression
  % w: d*1
  % X: n*d
  % y: n*1, should be -1 or 1

  y01 = (y+1)/2;
  mu = sigmoid(X*w);
  nll = -sum(y01 .* log(mu) + (1-y01) .* log(1-mu));

  if nargout > 1
    g = X'*(mu-y01);
  end

  if nargout > 2
    H = X'*diag(mu.*(1-mu))*X;
  end
  ```

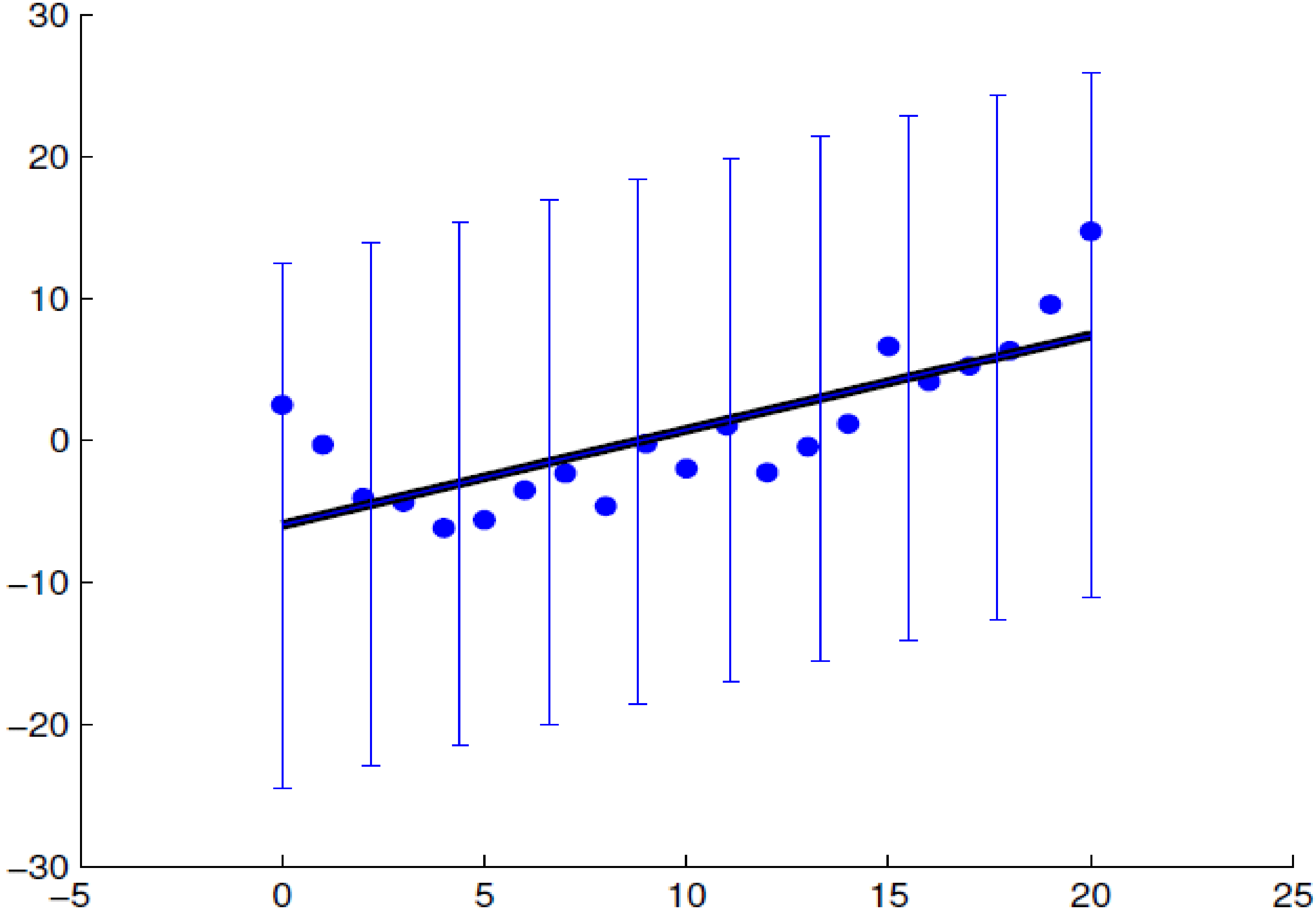
Robust regression



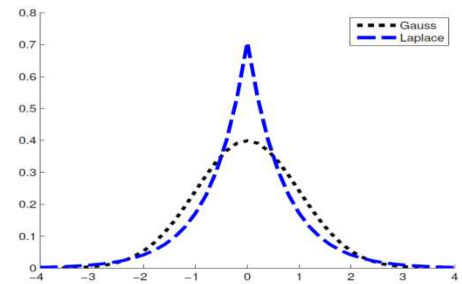
- One way to achieve **robustness** to **outliers** is to replace the Gaussian distribution for the response variable with a distribution that has **heavy tails**.

- Such a distribution will assign higher likelihood to outliers, without having to perturb the straight line to "explain" them.

Robust regression



Laplace distribution



- The **Laplace** distribution, also known as the **double sided exponential** distribution, has the following pdf:

$$\mathrm{Lap}(x|\mu, b) \quad := \quad \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

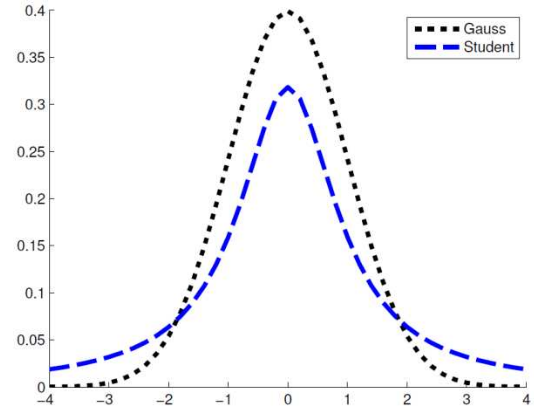
- Here μ is a location parameter and $b > 0$ is a scale parameter. The mean of the distribution is μ, and the variance is $2b^2$.

- The MLE for μ in a Laplace distribution is the median of the data, whereas the MLE for μ in a Gaussian distribution is the mean of the data

- If we use the Laplace distribution as the output density for linear regression, we get the following log-likelihood:

$$\log p(\mathcal{D}|\mathbf{w}, b) = \sum_{i=1}^{N} \log \mathrm{Lap}(y_i|\mathbf{w}^T\mathbf{x}_i, b) = -N\log(2b) - \frac{1}{b}\sum_{i=1}^{N} |y_i - \mathbf{w}^T\mathbf{x}_i|$$

Student T distribution



- The **Student T distribution** also has pdf:

$$\mathcal{T}(x|\mu,\sigma^2,\nu) \quad \propto \quad \left[1+\frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2\right]^{-\left(\frac{\nu+1}{2}\right)}$$

  where μ is the mean, $\sigma^2 > 0$ is the scale parameter, and $\nu > 0$ is called the **degrees of freedom**.

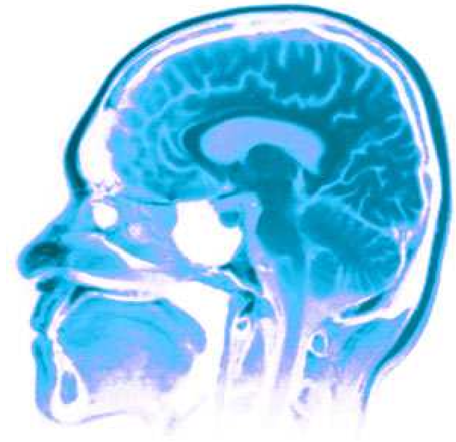
- If $\nu = 1$, it is known as the **Cauchy** distribution. As $\nu \to \infty$, the distribution rapidly approaches a Gaussian. Use $\nu \sim 3$; bigger than 2 to ensure it has finite variance and less than 5 to maintain heavy tails.

- The Student has heavier tails than the Laplace, making it more robust.

- Student distribution for robust linear regression model:

$$p(y_i|\mathbf{x}_i,\mathbf{w},\sigma^2,\nu) = \mathcal{T}(y_i|\mathbf{w}^T\mathbf{x}_i,\sigma^2,\nu)$$

Next class



Unconstrained optimization

Nando de Freitas
2011
KPM Book Sections: *11.2, 11.3 and 30.4*