From previous lecture

Linear regression

Vstupy: rows in matrix X.
Expected outputs: vector $\vec{y}$.
We are looking for parameters $\vec{\theta}$, such that $E = (X\vec{\theta} - \vec{y})^T(X\vec{\theta} - \vec{y})$ was smallest as possible.

Training

Option 1: solve system of equations $X^T X\vec{\theta} = X^T\vec{y}$
Option 2: (stochastic) gradient descent: $\vec{\theta} = \vec{\theta} - \alpha X^T(X\vec{\theta} - \vec{y})$

E is convex function, it has at most on local minimum, which is also global and both methods will find same solution (apart from numerical errors).

Prediction from new input

$y_{new} = \vec{x}_{new}^T \cdot \vec{\theta}$

Today

Still regression (one real number as an output).
Sometimes data are nonlinear.

- Locally weighted linear regression
- Polynomial regression and its reduction on linear
- Neural nets (not today)

Weighed linear regression

Each example gets a weight. We minimize:

$$E = \sum_{i=1}^{n} w_i((\vec{x}^{(i)})^T\vec{\theta} - y^{(i)})^2 = (X\vec{\theta} - \vec{y})^T I_w (X\vec{\theta} - \vec{y})$$

$$I_w = \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & w_n \end{pmatrix}$$

Weighed linear regression

Each example gets a weight. We minimize:

$$E = \sum_{i=1}^{n} w_i((\vec{x}^{(i)})^T\vec{\theta} - y^{(i)})^2 = (X\vec{\theta} - \vec{y})^T I_w (X\vec{\theta} - \vec{y})$$

$$I_w = \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & w_n \end{pmatrix}$$

We want zero gradient:

$$\nabla_{\vec{\theta}} E = 2X^T I_w (X\vec{\theta} - \vec{y}) = \vec{0}$$

Solution:

$$X^T I_w X\vec{\theta} = X^T I_w \vec{y}$$

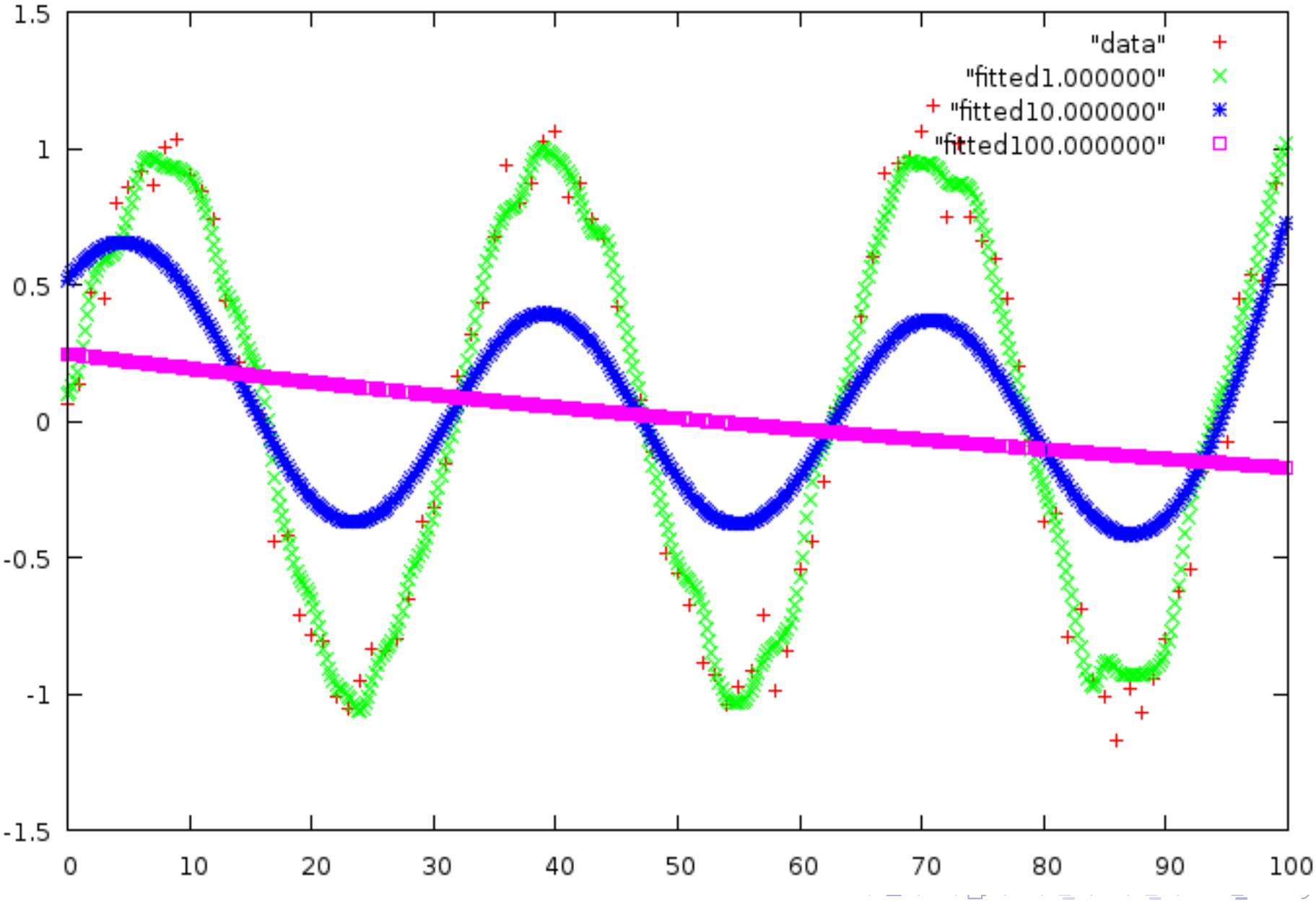
Locally weighted linear regression

We want to predict value at $\vec{x}_{new}$.

We put: $w_i = e^{\frac{-\|\vec{x}_{new} - \vec{x}^{(i)}\|^2}{\sigma^2}}$ (σ is an hyperparameter, which should be set separatelly)

We run weighted linear regression and predict the value (yes we do new training for each new output).

LVLR result


"data"
"fitted1.000000"
"fitted10.000000"
"fitted100.000000"
1.5
1
0.5
0
-0.5
-1
-1.5
0
10
20
30
40
50
60
70
80
90
100

Polynomial regression

One input x, model (at to degree 2):

$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

Polynomial regression

One input x, model (at to degree 2):

$$y = \theta_0 + \theta_1 x + \theta_2 x^2$$

Two inputs x_1, x_2, model (up to degree 2):

$$y = \theta_0 + \theta_{10} x_1 + \theta_{01} x_2 + \theta_{11} x_1 x_2 + \theta_{20} x_1^2 + \theta_{02} x_2^2$$

We can use same procedure as lat time and find values of θ. Or reduce the problem to linear regression.

Reduction of polynomial regression

For two inputs

Input: $(1, x_1, x_2)$ we change into:
$(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$
And we can solve linear regression (we do not change outputs).

Reduction of polynomial regression

For two inputs

Input: $(1, x_1, x_2)$ we change into:
$(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$
And we can solve linear regression (we do not change outputs).

In general

We have p basis functions: $\phi_1(\vec{x}), \phi_2(\vec{x}), \ldots, \phi_p(\vec{x})$, kde $\phi_i \in \mathbb{R}^m \to \mathbb{R}$.
We preprocess input matrix X into matrix Φ:

$$\begin{pmatrix} \phi_1(\vec{x}^{(1)}) & \phi_2(\vec{x}^{(1)}) & \ldots & \phi_p(\vec{x}^{(1)}) \\ \phi_1(\vec{x}^{(2)}) & \phi_2(\vec{x}^{(2)}) & \ldots & \phi_p(\vec{x}^{(2)}) \\ & & \vdots & \\ \phi_1(\vec{x}^{(n)}) & \phi_2(\vec{x}^{(n)}) & \ldots & \phi_p(\vec{x}^{(n)}) \end{pmatrix}$$

And we solve linear regression, for example the system: $\Phi^T\Phi\vec{\theta} = \Phi^T\vec{y}$

Basis fuctions - examples

Not only polynomials.

- $\phi(\vec{x}) = x_4 x_7$, $\phi(\vec{x}) = x_2$
- 0-1 functions: $\phi(\vec{x}) = x_6 > 0$
- Some preprocessings: $\phi(\vec{x}) = \log(x_5 + 1)$
- Kernel fuctions: $\phi(\vec{x}) = e^{\frac{-\|\vec{z}-\vec{x}\|^2}{\sigma^2}}$

Linear regression with preprocessing