

# Machine Learning in Robotics

## Lecture 2: Regression

**Prof. Dongheui Lee**

*Institute of Automatic Control Engineering  
Technische Universität München*

dhlee@tum.de

# Regression problems

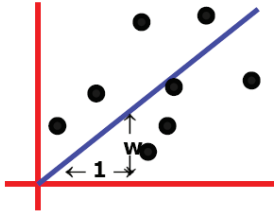
- The goal is to make quantitative (real valued) predictions on the basis of a vector of features or attributes
- Examples: house prices, stock values, survival time, fuel efficiency of cars, etc.
- Questions: What can we assume about the problem? How do we formalize the regression problem? How do we evaluate predictions?

# Linear Regression

- Linear regression assumes that expected value of the output given an input is linear.

$$y^{(i)} = wx^{(i)} + \epsilon \quad (1)$$

input $x$	output $y$
1	1
2	2
3	2.2
4	3.1
1.5	1.9



$x^{(i)}$  : i-th input

$y^{(i)}$  : i-th output

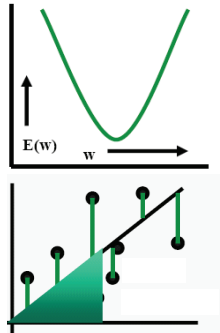
# Linear Least Squares Regression : Single Parameter

- Which value of  $w$  makes the output values most likely?
- One that minimizes sum of squares of residuals.

$$E = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - wx^{(i)})^2$$

$$w =$$

- We can use it for prediction.

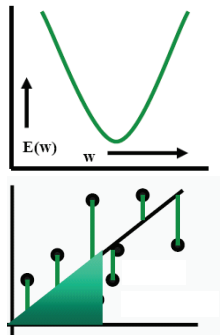


# Linear Least Squares Regression : Single Parameter

- Which value of  $w$  makes the output values most likely?
- One that minimizes sum of squares of residuals.

$$E = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - wx^{(i)})^2$$
$$w = \frac{\sum_{i=1}^n x^{(i)} y^{(i)}}{\sum_{i=1}^n x^{(i)2}}$$

- We can use it for prediction.



# Linear Least Squares Regression

We need to define a class of functions (types of predictions we will try to make) such as linear predictions:

$$f(x) = w_0 + w_1x \quad (2)$$

where  $w_0$  and  $w_1$  are the parameters we need to set.

We need an estimation criterion so as to be able to select appropriate values for our parameters ( $w_0$  and  $w_1$ ) based on the training set  $\{(x^{(i)}, y^{(i)}), \dots, (x^{(n)}, y^{(n)})\}$

For example, we can use the empirical loss:

$$E = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - f(x^{(i)}))^2 \quad (3)$$

# Estimating the parameters

- We minimize the empirical squared loss

$$\begin{aligned}
 E &= \frac{1}{n} \sum_{i=1}^n (y^{(i)} - f(x^{(i)}))^2 \\
 &= \frac{1}{n} \sum_{i=1}^n (y^{(i)} - w_0 - w_1 x^{(i)})^2
 \end{aligned}
 \tag{4}$$

- By setting the derivatives with respect to  $w_0$  and  $w_1$  to zero we get necessary conditions for the optimal parameter values

$$\begin{aligned}
 \frac{\partial}{\partial w_0} E &= 0 \\
 \frac{\partial}{\partial w_1} E &= 0
 \end{aligned}
 \tag{5}$$

# Estimating the parameters

- By setting the derivatives with respect to  $w_0$  and  $w_1$  to zero

$$\begin{aligned}\frac{\partial}{\partial w_0} E &= 0 \\ \frac{\partial}{\partial w_1} E &= 0\end{aligned}\tag{6}$$

- we get necessary conditions for the optimal parameter values

$$\begin{aligned}w_0 &= \frac{\sum y^{(i)} \sum x^{(i)2} - \sum x^{(i)} \sum x^{(i)} y^{(i)}}{n \sum x^{(i)2} - (\sum x^{(i)})^2} \\ w_1 &= \frac{n \sum x^{(i)} y^{(i)} - \sum x^{(i)} \sum y^{(i)}}{n \sum x^{(i)2} - (\sum x^{(i)})^2}\end{aligned}\tag{7}$$



# Linear regression problem with multiple variables

We can express the solution a bit more generally by resorting to a matrix notation

$$X = \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(n)} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_m^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_m^{(2)} \\ & \vdots & \vdots & & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_m^{(n)} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$$

so that  $f(\mathbf{x}) = X\mathbf{w}$ .

The result becomes

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

# Solving Linear regression in matrix notation

Our empirical loss becomes  $E = \frac{1}{n} \|y - Xw\|^2$ .

By setting the derivatives of  $E$  with respect to  $w$  to zero, we get the same optimality conditions as before, now expressed in a matrix form

$$\begin{aligned}\frac{\partial}{\partial w} E &= \frac{1}{n} \frac{\partial}{\partial w} (y - Xw)^T (y - Xw) \\ &= \frac{1}{n} \frac{\partial}{\partial w} (w^T X^T X w - 2y^T X w + y^T y) \\ &= \frac{1}{n} \left( \frac{\partial w^T X^T X w}{\partial w} - 2y^T X \right) \\ &= \frac{1}{n} (2w^T X^T X - 2y^T X) = 0\end{aligned}$$

which yields

$$w^* = (X^T X)^{-1} X^T y$$

# Gradient Descent

- Another way to minimize  $E(\mathbf{w})$
- Start with an initial value of  $\mathbf{w}$ , keep changing  $\mathbf{w}$  to reduce  $E(\mathbf{w})$

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} E(\mathbf{w})$$

$$w_j := w_j - 2\alpha(f(\mathbf{x}) - y)x_j$$

- Batch Gradient Descent
  - All training data is taken into account

$$w_j := w_j - \frac{\alpha}{n} \sum_{i=1}^n (f(\mathbf{x}^{(i)}) - y^{(i)})x_j^{(i)}$$

- Incremental ( Stochastic) Gradient Descent

$$w_j :=$$

# Gradient Descent

- Another way to minimize  $E(\mathbf{w})$
- Start with an initial value of  $\mathbf{w}$ , keep changing  $\mathbf{w}$  to reduce  $E(\mathbf{w})$

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} E(\mathbf{w})$$

$$w_j := w_j - 2\alpha(f(\mathbf{x}) - y)x_j$$

- Batch Gradient Descent
  - All training data is taken into account

$$w_j := w_j - \frac{\alpha}{n} \sum_{i=1}^n (f(\mathbf{x}^{(i)}) - y^{(i)})x_j^{(i)}$$

- Incremental ( Stochastic) Gradient Descent

$$w_j := w_j - \alpha(f(\mathbf{x}^{(i)}) - y^{(i)})x_j^{(i)}, \text{ for } i = 1 \text{ to } n$$

# Probabilistic approach

- Assume

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

$$\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$$

$$p(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \mathbf{x}^{(i)} \mathbf{w})^2}{2\sigma^2}\right)$$

- Likelihood

$$\begin{aligned} L(\mathbf{w}) &= \prod_{i=1}^n p(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \mathbf{x}^{(i)} \mathbf{w})^2}{2\sigma^2}\right) \end{aligned}$$

- Choose parameters to maximize the likelihood  
= same as minimizing LMS

# Beyond linear regression

- The linear regression functions

$$f(\mathbf{x}) = w_0 + w_1x_1 + \dots + w_mx_m$$

are convenient because they are linear in the parameters, not necessarily in the input  $\mathbf{x}$

- We can easily generalize these classes of functions to be non-linear functions of the inputs  $\mathbf{x}$  but still linear in the parameters  $\mathbf{w}$ . For example:  $m$ th order polynomial prediction

$$f(x) = w_0 + w_1x + w_2x^2 + \dots + w_mx^m$$

# Quadratic Regression

$$f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_1x_2 + w_5x_2^2$$

$X_1$	$X_2$	$Y$
3	2	7
1	1	3
$\vdots$	$\vdots$	$\vdots$

$\mathbf{X} =$	3	2	$\mathbf{y} =$	7
	1	1		3
	$\vdots$	$\vdots$		$\vdots$

$\mathbf{Z} =$	1	3	2	9	6	4	$\mathbf{y} =$	7
	1	1	1	1	1	1		3
	$\vdots$					$\vdots$		$\vdots$

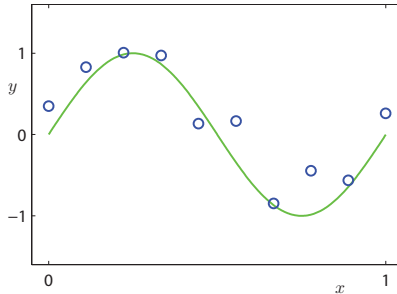
$$\mathbf{z} = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$$

$$\beta = (\mathbf{Z}^T \mathbf{Z})^{-1} (\mathbf{Z}^T \mathbf{y})$$

$$f(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_1 x_2 + \beta_5 x_2^2$$

# Polynomial Curve Fitting

$$f(\mathbf{x}) = w_0 + w_1x + w_2x^2 + \dots + w_mx^m$$

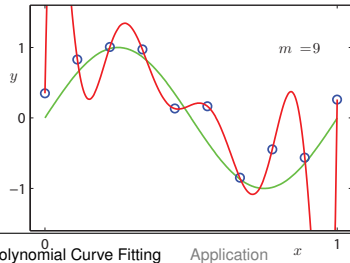
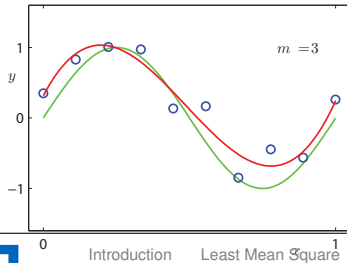
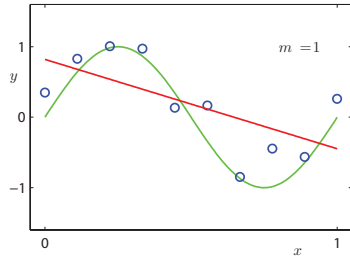
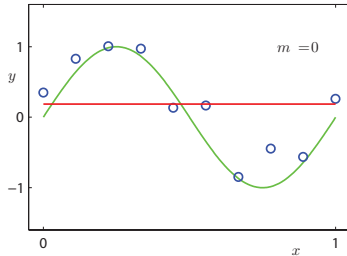


Minimize the empirical error

$$E = \frac{1}{n} \sum_{i=1}^n (y^{(i)} - f(x^{(i)}))^2$$

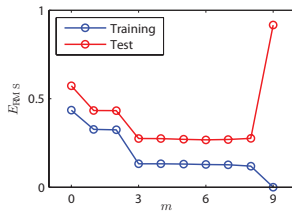


# Polynomial Curve Fitting with different orders



# Polynomial Curve Fitting

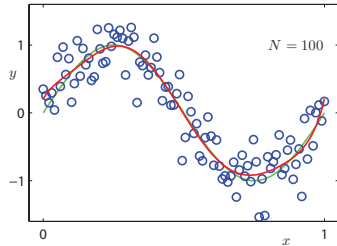
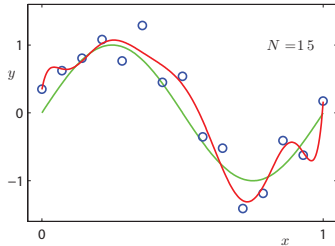
Root-mean-square error & Polynomial coefficients



	$m = 0$	$m = 1$	$m = 3$	$m = 9$
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$			17.37	48568.31
$w_4^*$				-231639.30
$w_5^*$				640042.26
$w_6^*$				-1061800.52
$w_7^*$				1042400.18
$w_8^*$				-557682.99
$w_9^*$				125201.43

# Polynomial Curve Fitting

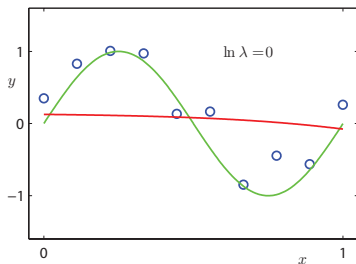
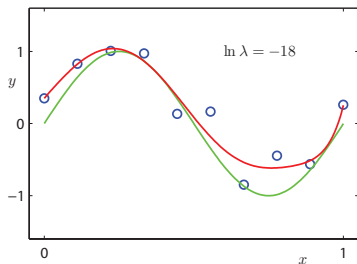
9th order polynomials by increasing the training data,  $n = 15$  and  $n = 100$



# Regularization

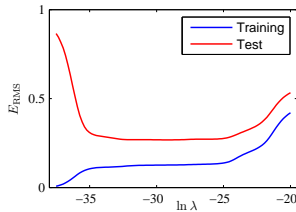
Penalize large coefficient values

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^n (f(x^{(i)}, \mathbf{w}) - y^{(i)})^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$



# Regularization

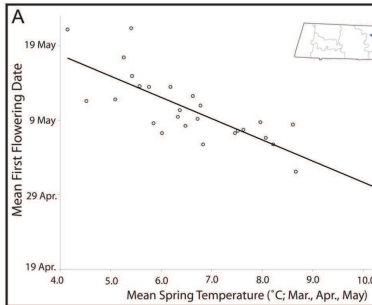
Root-mean-square error & Polynomial coefficients



	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^*$	0.35	0.35	0.13
$w_1^*$	232.37	4.74	-0.05
$w_2^*$	-5321.83	-0.77	-0.06
$w_3^*$	48568.31	-31.97	-0.05
$w_4^*$	-231639.30	-3.89	-0.03
$w_5^*$	640042.26	55.28	-0.02
$w_6^*$	-1061800.52	41.32	-0.01
$w_7^*$	1042400.18	-45.95	-0.00
$w_8^*$	-557682.99	-91.53	0.00
$w_9^*$	125201.43	72.68	0.01

# Phenological Application: Temperature-phenology relationship

Can we detect a response to temperature in phenology?



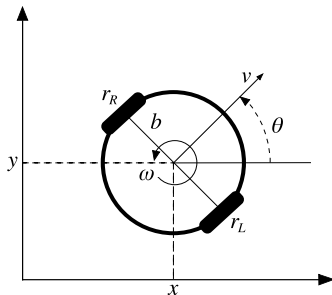
no.	Visualization	Description	no.	Visualization	Description
58		No flower petals visible	62		About 20 % of flowers open
59		First flower petals visible (in petalled form)	63		About 30 % of flowers open
60		First flowers open	64		About 40 % of flowers open
61		Beginning of flowering, about 50 % of flowers open (in contrast to 60 distributed over the whole perianth and base)	65		Full flowering, at least 50 % of flowers open, petals falling



Jochner. 2008, Ellwood et al. 2013.

# Robotics Applications: Odometry calibration

Estimate the pose  $(x, y, \theta)$  of a mobile robot given the angular velocity of each wheel  $(\omega_L, \omega_R)$



$$\dot{x} = v \cos(\theta)$$

$$\dot{y} = v \sin(\theta)$$

$$\dot{\theta} = \omega$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = W(r_R, r_L, b) \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

Odometry calibration consists in estimating  $W$

# Robotics Applications: Odometry calibration

$$\begin{aligned}
 x_{t+1} &= x_t + v \cos(\theta) \Delta T \\
 y_{t+1} &= y_t + v \sin(\theta) \Delta T \\
 \theta_{t+1} &= \theta_t + \omega \Delta T
 \end{aligned}
 \Rightarrow
 \begin{aligned}
 \begin{bmatrix} x_N - x_0 \\ y_N - y_0 \end{bmatrix} &= \mathbf{X}(\omega_R, \omega_L) \begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix} \\
 \theta_N - \theta_0 &= \mathbf{Z}(\omega_R, \omega_L) \begin{bmatrix} w_{21} \\ w_{22} \end{bmatrix}
 \end{aligned}$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

Executing  $M$  trajectories the parameters  $(w_{11}, w_{12}, w_{21}, w_{22})$  can be identified using Linear Regression



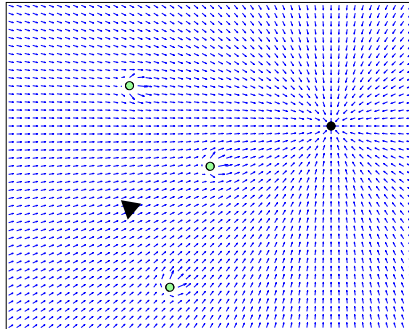
Antonelli G., Chiaverini S., and Fusco G. *A Calibration Method for Odometry of Mobile Robots Based on the Least-Squares Technique: Theory and Experimental Validation*. Transactions on Robotics. 2005.



# Robotics Applications: Obstacle avoidance

Assign an attractive potential ( $u_{att}$ ) to the goal position and repulsive potentials ( $u_{rep}$ ) to the obstacles

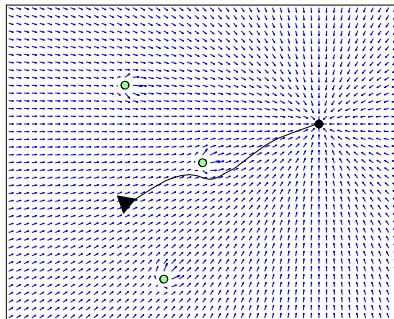
$$u = u_{att} + u_{rep} \rightarrow f_j = \frac{\partial}{\partial p_j} u_{att} + \frac{\partial}{\partial p_j} u_{rep}$$



# Robotics Applications: Obstacle avoidance

A collision-free trajectory is generated using Gradient Descent

$$\mathbf{p}^{(i+1)} = \mathbf{p}^{(i)} - \alpha \frac{\mathbf{f}^{(i)}}{\|\mathbf{f}\|}$$



Khatib O. *Real-time obstacle avoidance for manipulators and mobile robots.*  
International Journal of Robotics Research. 1986.



# Reference

- Bishop, Pattern Recognition and Machine Learning  
Chap. 1.1, 3, 6.4.1, 6.4.2
- Mitchell, Machine Learning  
Chap. 4.4.3, 8.3