Machine Learning in Robotics
Lecture 2: Regression

Prof. Dongheui Lee

Institute of Automatic Control Engineering
Technische Universität München

dhlee@tum.de

Regression problems

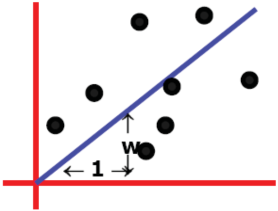
- The goal is to make quantitative (real valued) predictions on the basis of a vector of features or attributes
- Examples: house prices, stock values, survival time, fuel efficiency of cars, etc.
- Questions: What can we assume about the problem? How do we formalize the regression problem? How do we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

Linear Regression

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

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

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

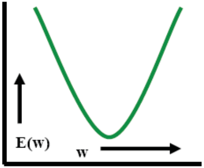
Linear Least Squares Regression : Single Parameter

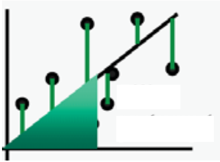
- Which value of w makes the output values most likely?

- One that minimizes sum of squares of residuals.

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

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





- 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:

$$f(x) = w_0 + w_1 x \tag{2}$$

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 $\{(x^{(i)}, y^{(i)}), \ldots, (x^{(n)}, y^{(n)})\}$
For example, we can use the empirical loss:

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

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

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

Linear regression problem with multiple variables

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

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

so that $f(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{w}$.
The result becomes

$$\boldsymbol{w}^\star = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Solving Linear regression in matrix notation

Our empirical loss becomes $E = \frac{1}{n} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$.
By setting the derivatives of E with respect to $\boldsymbol{w}$ to zero, we get the same optimality conditions as before, now expressed in a matrix form

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

which yields

$$\boldsymbol{w}^\star = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

Gradient Descent

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

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

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

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

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

- Incremental (Stochastic) Gradient Descent

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

Probabilistic approach

- Assume

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

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

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

- Likelihood

$$L(\boldsymbol{w}) = \prod_{i=1}^{n} p(y^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{w})$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \boldsymbol{x}^{(i)}\boldsymbol{w})^2}{2\sigma^2}\right)$$

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

Beyond linear regression

- The linear regression functions

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + \cdots + w_m x_m$$

  are convenient because they are linear in the parameters, not necessarily in the input $\boldsymbol{x}$
- We can easily generalize these classes of functions to be non-linear functions of the inputs x but still linear in the parameters w. For example: mth order polynomial prediction

$$f(\boldsymbol{x}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m$$

Quadratic Regression

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2$$

X_1	X_2	Y
3	2	7
1	1	3
:	:	:

$$\mathbf{X} = \begin{bmatrix} 3 & 2 \\ 1 & 1 \\ \vdots & \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 7 \\ 3 \\ \vdots \end{bmatrix}$$

$$\mathbf{Z} = \begin{bmatrix} 1 & 3 & 2 & 9 & 6 & 4 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & & & & & \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 7 \\ 3 \\ \vdots \end{bmatrix}$$

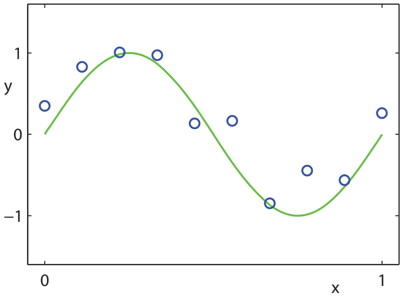
$$z = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$$

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

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

Polynomial Curve Fitting

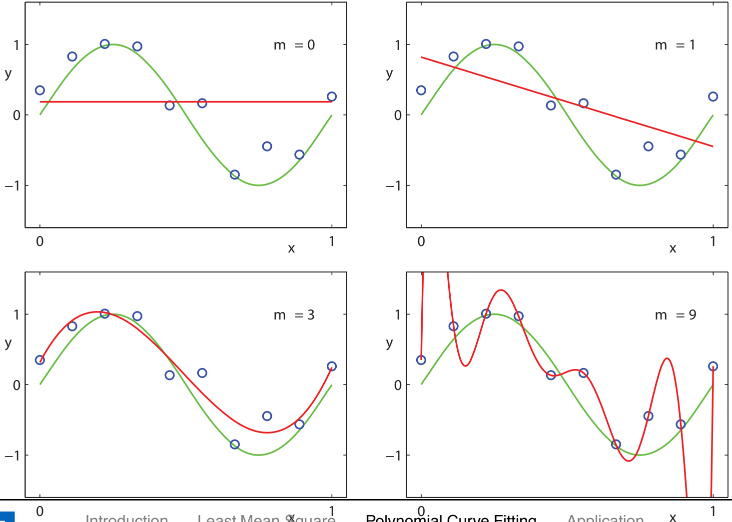
$$f(\boldsymbol{x}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m$$



Minimize the empirical error

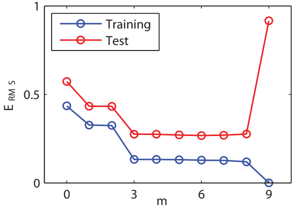
$$E = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}))^2$$

Polynomial Curve Fitting with different orders



Polynomial Curve Fitting

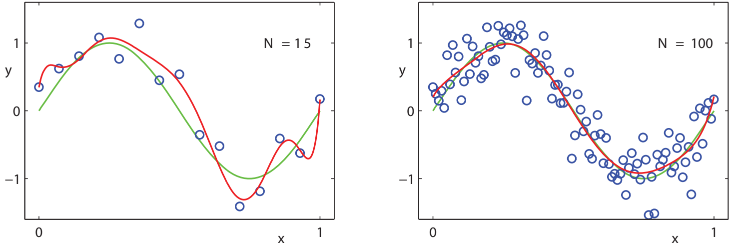
Root-mean-square error & Polynomial coefficients



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

Polynomial Curve Fitting

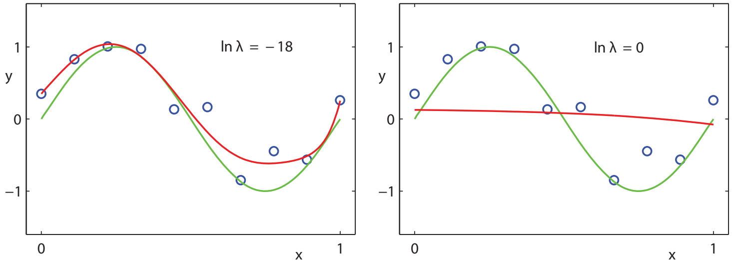
9th order polynomials by increasing the training data, $n = 15$ and $n = 100$



Regularization

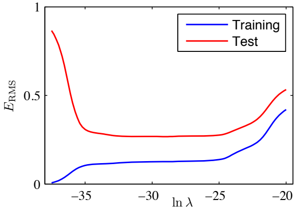
Penalize large coefficient values

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



Regularization

Root-mean-square error & Polynomial coefficients



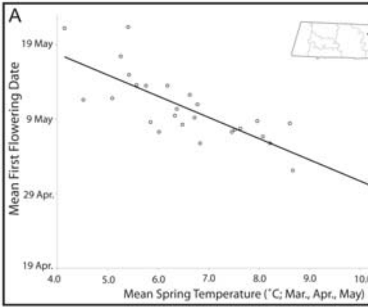
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$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

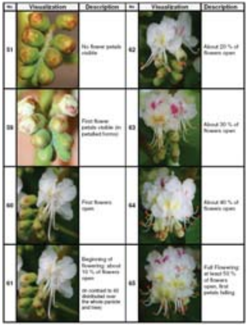
Applet

http://mste.illinois.edu/users/exner/java.f/leastsquares/

Phenological Application: Temperature-phenology relationship

Can we detect a response to temperature in phenology?

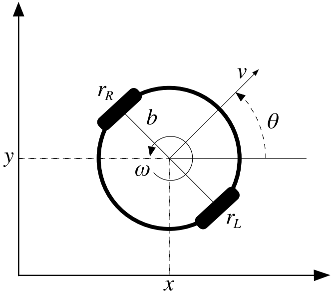




Jochner. 2008, Ellwood et al. 2013.

Robotics Applications: Odometry calibration

Estimate the pose (x, y, θ) of a mobile robot given the angular velocity of each wheel (ω_L, ω_R)



$$\dot{x} = v cos(\theta)$$
$$\dot{y} = v sin(\theta)$$
$$\dot{\theta} = \omega$$
$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \boldsymbol{W}(r_R, r_L, b) \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$

Odometry calibration consists in estimating $\boldsymbol{W}$

Robotics Applications: Odometry calibration

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

$$
\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}
\qquad
\theta_N - \theta_0 = \boldsymbol{Z}(\omega_R, \omega_L) \begin{bmatrix} w_{21} \\ w_{22} \end{bmatrix}
$$

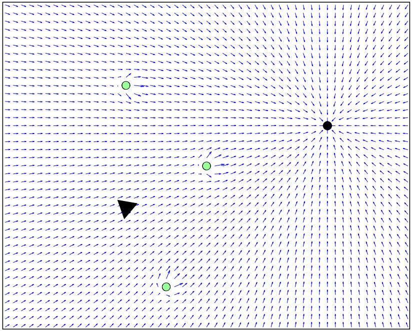
Executing M trajectories the parameters $(w_{11}, w_{12}, w_{21}, w_{22})$ can be identified using Linear Regression

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

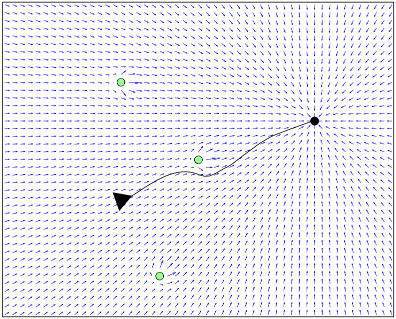
$$u = u_{att} + u_{rep} \to f_j = \frac{\partial}{\partial p_j} u_{att} + \frac{\partial}{\partial p_j} u_{rep}$$



Robotics Applications: Obstacle avoidance

A collision-free trajectory is generated using Gradient Descent

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



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