Sapienza University of Rome

Master in Engineering in Computer Science

Artificial Intelligence & Machine Learning

A.Y. 2024/2025

Prof. Fabio Patrizi

2. Linear Regression

Fabio Patrizi

Overview

- Linear models for regression
- Least Squared Error
- Normal Equations and Stochastic Gradient Descent
- Least Squares and Maximum Likelihood
- Regularization

References

- Lecture notes and slides
- [AIMA] 19.6.1 - 19.6.3

Regression

Regression problem:

- Target function $f : X \to Y$
- $X \subseteq \mathbb{R}^m$
- $Y = \mathbb{R}$ (or even $\mathbb{R}^n$)

$f : \mathbb{R}^m \to \mathbb{R}$

- $D = \{\langle \mathbf{x}_1, t_1 \rangle, \ldots, \langle \mathbf{x}_N, t_N \rangle\}$

- Find hypothesis h that *best* approximates f

Linear Models for Regression

Linear regression: hypothesis are linear functions

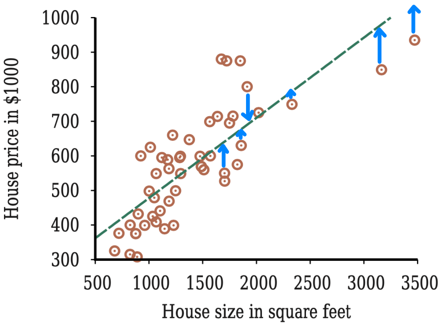
- $h(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_m x_m = \mathbf{w}^T \tilde{\mathbf{x}}$
- parameters: $\mathbf{w}$
- $\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{bmatrix}$, $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$

Example

- D contains pairs size-price for apartments
- Given size, predict price
- $price(x) = w_1 size + w_0$

↑ IS THE ERROR → $(h(x_n) - t_n)^2$

$\sum_{i=1}^{n} (h(x_i) - t_i)^2$ = CUMULATIVE ERROR



Least Squared Error

How to *best* fit the data?

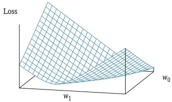
- Intuition: Measure error with quadratic distance wrt ground truth
- Loss function: squared errors (wrt D)

$$E(\mathbf{w}) = \frac{1}{2} \sum_{\langle \mathbf{x}, t \rangle \in D} (t - h(\mathbf{x}; \mathbf{w}))^2 = \frac{1}{2} \sum_{\langle \mathbf{x}, t \rangle \in D} (t - \mathbf{w}^T \tilde{\mathbf{x}})^2$$

- Minimiz error: $\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}} E(\mathbf{w})$
- (Recall: parameters $\mathbf{w}$ act as variables)

Analytical Solution (Normal Equation)

- Convex loss function: $E(\mathbf{w}) = \sum_{\langle \mathbf{x}, t \rangle \in D} (t - \mathbf{w}^T \tilde{\mathbf{x}})^2$



- Optimality condition: $\frac{\partial E}{\partial w_i}(\mathbf{w}) = 0$, for $i = 1, \ldots, m$

- Solution: $\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{t}$ (*Normal Equation*)

- $\mathbf{X} = \begin{bmatrix} 1 \; x_{11} & \cdots & x_{1m} \\ & \cdots & \\ 1 \; x_{N1} & \cdots & x_{Nm} \end{bmatrix} = \begin{bmatrix} \tilde{x}_1^T \\ \cdots \\ \tilde{x}_N^T \end{bmatrix}$ (*Design* matrix), $\mathbf{t} = \begin{bmatrix} t_1 \\ \cdots \\ t_N \end{bmatrix}$

- $\mathbf{X}^\dagger = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$ (*Pseudo-inverse* of $\mathbf{X}$)

Iterative Solution: Gradient Descent

Idea: update **w** iteratively, guided by gradient

Algorithm:

- Initialize **w** with random values
- **repeat until** (termination condition)
  - **for each** w_i **do**
    - $w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}(\mathbf{w})$

Termination conditions:

- $\#$ of iterations
- $\frac{\partial E}{\partial w_i}(\mathbf{w}) = 0$ (requires small η: slower)
- threshold on changes in $E(\mathbf{w})$

Algorithm converges for suitable small values of η.

Stochastic Gradient Descent

Every step requires scanning entire D:

- $\frac{\partial E(\mathbf{w})}{\partial w_i} = -\sum_{\langle \mathbf{x},t \rangle \in D} (t - \mathbf{w}^T \tilde{\mathbf{x}}) x_i$

Stochastic Gradient Descent:

- select random $S \subset D$
- approximate $\frac{\partial E}{\partial w_i}(\mathbf{w})$ with $\Delta_i = -\sum_{\langle \mathbf{x},t \rangle \in S} (t - \mathbf{w}^T \tilde{\mathbf{x}}) x_i$
- replace update rule with: $w_i \leftarrow w_i - \eta \Delta_i$

Effects:

- Faster convergence
- Possible lower accuracy
- Excellent performance in practice

Normal Equation vs (Stochastic) Gradient Descent

Normal Equation:

- Pros:
  - No hyper-parameters (learning rate)
- Cons:
  - $\mathbf{w}^*$ can be computed in $\mathcal{O}(Nm^2)$: slow with many features
  - computing matrix product may yield accuracy issues

(Stochastic) Gradient Descent:

- Pros:
  - better scalability on large datasets
  - better solution accuracy
- Cons:
  - hyper-parameters (learning rate), may need several runs for tuning

LSE: Probabilistic Interpretation

Squared-error loss function $E(\mathbf{w})$ may appear arbitrary

- Assume:
  - observations independent and identically distributed (i.i.d.)
  - Gaussian noise: $\epsilon \sim \mathcal{N}(0, \sigma^2)$
  - $y(\mathbf{x}) = h(\mathbf{x}; \mathbf{w}) + \epsilon = \mathbf{w}^T \tilde{\mathbf{x}} + \epsilon$
- Likelihood of $\mathbf{t}$:
  $P(\mathbf{t}|\mathbf{X}; \mathbf{w}) = \prod_{i=1}^N P(t_i|\mathbf{x}_i; \mathbf{w}) = \prod_{i=1}^N \mathcal{N}(t_i; \mathbf{w}^T \tilde{\mathbf{x}}_i, \sigma^2)$
- Log-likelihood of $\mathbf{t}$: $\ln P(\mathbf{t}|\mathbf{X}; \mathbf{w}) = \sum_{i=1}^N \ln \left(\mathcal{N}(t_i; \mathbf{w}^T \tilde{\mathbf{x}}_i, \sigma^2) \right) =$

$$-\frac{1}{\sigma^2} \underbrace{\frac{1}{2} \sum_{i=1}^N (t_i - \mathbf{w}^T \tilde{\mathbf{x}}_i)^2}_{E(\mathbf{w})} - \frac{N}{2} \ln(2\pi\sigma^2)$$

LSE: Probabilistic Interpretation

$$
\begin{aligned}
\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w}} P(\mathbf{t}|\mathbf{X};\mathbf{w}) = \operatorname*{argmax}_{\mathbf{w}} \ln P(\mathbf{t}|\mathbf{X};\mathbf{w}) = \\
\operatorname*{argmax}_{\mathbf{w}} \left(-\frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{N}(t_i - \mathbf{w}^T\tilde{\mathbf{x}}_i)^2 - \frac{N}{2}\ln(2\pi\sigma^2) \right) = \\
\operatorname*{argmax}_{\mathbf{w}} \left(-\frac{1}{2}\sum_{i=1}^{N}(t_i - \mathbf{w}^T\tilde{\mathbf{x}}_i)^2 \right) = \\
\operatorname*{argmin}_{\mathbf{w}} \left(\frac{1}{2}\sum_{i=1}^{N}(t_i - \mathbf{w}^T\tilde{\mathbf{x}}_i)^2 \right) = \\
\operatorname*{argmin}_{\mathbf{w}} E(\mathbf{w})
\end{aligned}
$$

- LSE equivalent to ML (assuming zero-mean Gaussian noise)

Feature Maps

Consider

- Univariate target function: $f : \mathbb{R} \rightarrow \mathbb{R}$
- $D = \{\langle x_1, t_1 \rangle, \ldots, \langle x_N, t_N \rangle\}$

Add features to x (as functions of x)

- $\phi(x) = \langle x, x^2, x^3 \rangle$
- In general, function $\phi : R^m \rightarrow R^n$ (*feature map*) s.t.:
  - $\phi(\mathbf{x}) = \langle \phi_0(\mathbf{x}), \ldots, \phi_n(\mathbf{x}) \rangle$
  - $\phi_0(\mathbf{x}) = 1$
- Observe: ϕ non-linear function of input $\mathbf{x}$

Feature Maps

- $\mathbf{X} = \begin{bmatrix} x_1^T \\ \cdots \\ x_N^T \end{bmatrix}, \mathbf{t} = \begin{bmatrix} t_1 \\ \cdots \\ t_N \end{bmatrix}, \Phi(\mathbf{X}) = \begin{bmatrix} \phi_0(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_n(x_N) \end{bmatrix}$

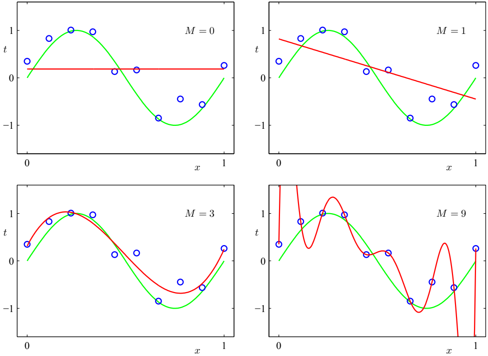
- Example: $\Phi(\mathbf{X}) = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & x_N^3 \end{bmatrix}$

Can still fit $\Phi(\mathbf{X}), \mathbf{t}$ with linear hypothesis:

- $h(\mathbf{x}; \mathbf{w}) = w_0\phi_0(\mathbf{x}) + \cdots + w_n\phi_n(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$
- $\mathbf{w}^T = \begin{bmatrix} w_0 & \cdots & w_m \end{bmatrix}$
- Linear in parameters $\mathbf{w}$

Example: Polynomial curve fitting

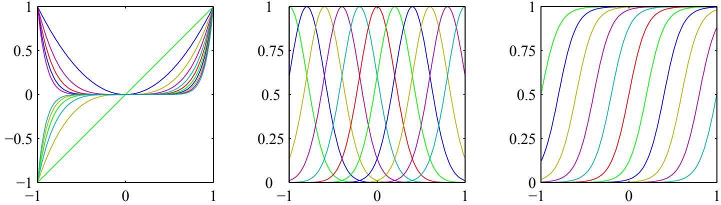
$$h(\mathbf{x}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$



Warning: overfitting!!!

Linear Regression Basis Functions

Examples of basis functions



Polynomial Radial Sigmoid / Tanh

Regularization

Regularization:

- Technique to mitigate over-fitting
- Idea: penalize complex hypotheses

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} E(\mathbf{w}) + \lambda R(\mathbf{w})$$

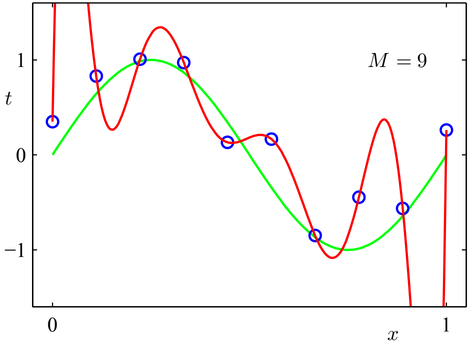
- $R(\mathbf{w})$: *regularization function*
- $\lambda > 0$: *regularization factor*

Common choices (penalize large components of w):

- $R(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$
- $R(\mathbf{w}) = \sum_{i=0}^{m} |w_i|^q$ (for some q)

Linear Regression - Regularization

$$\underset{\mathbf{w}}{\operatorname{argmin}}\, E(\mathbf{w})$$



Linear Regression - Regularization

$$\underset{\mathbf{w}}{\operatorname{argmin}} \; E(\mathbf{w}) + \lambda \frac{1}{2} \mathbf{w}^T \mathbf{w}$$