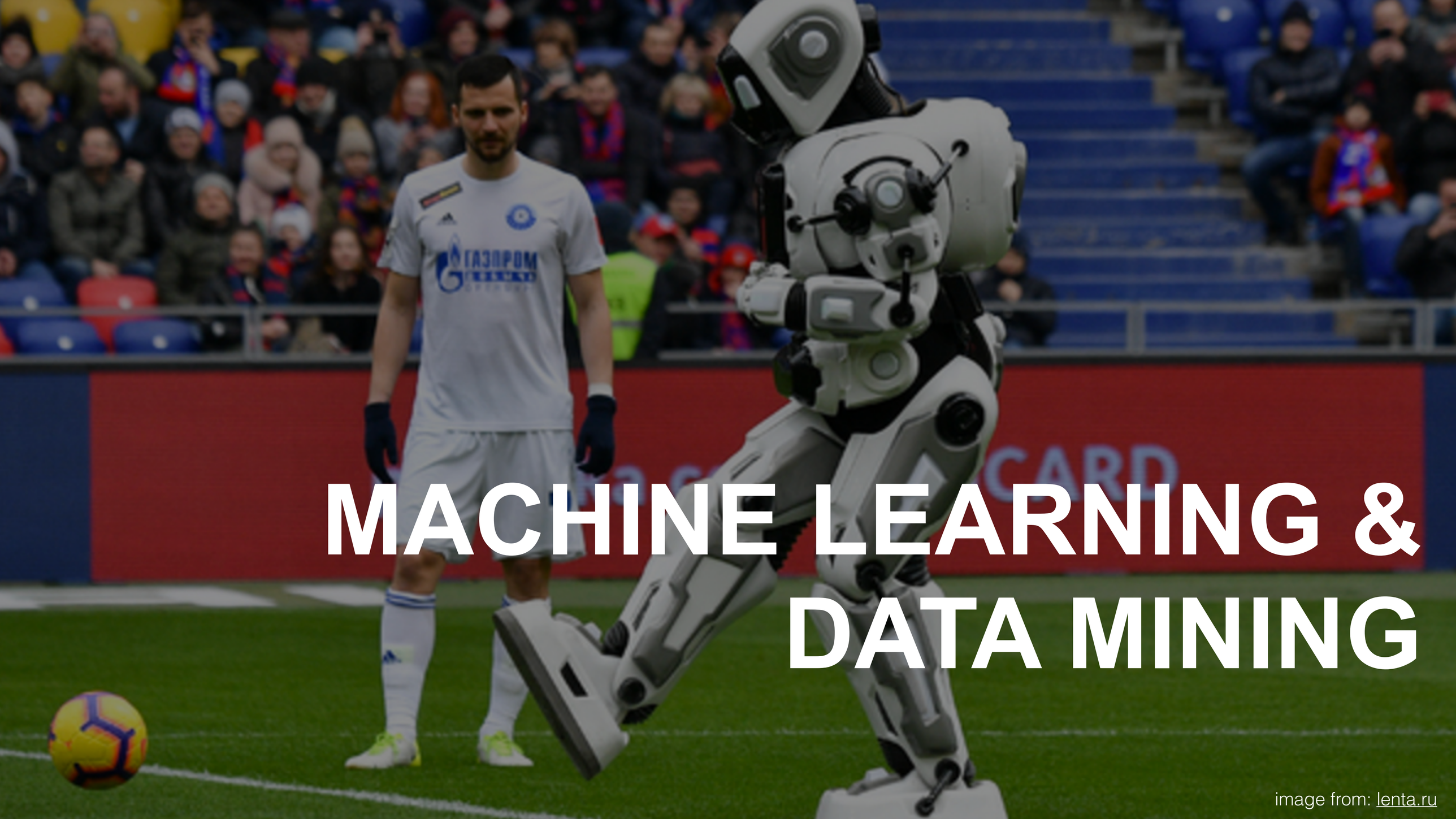
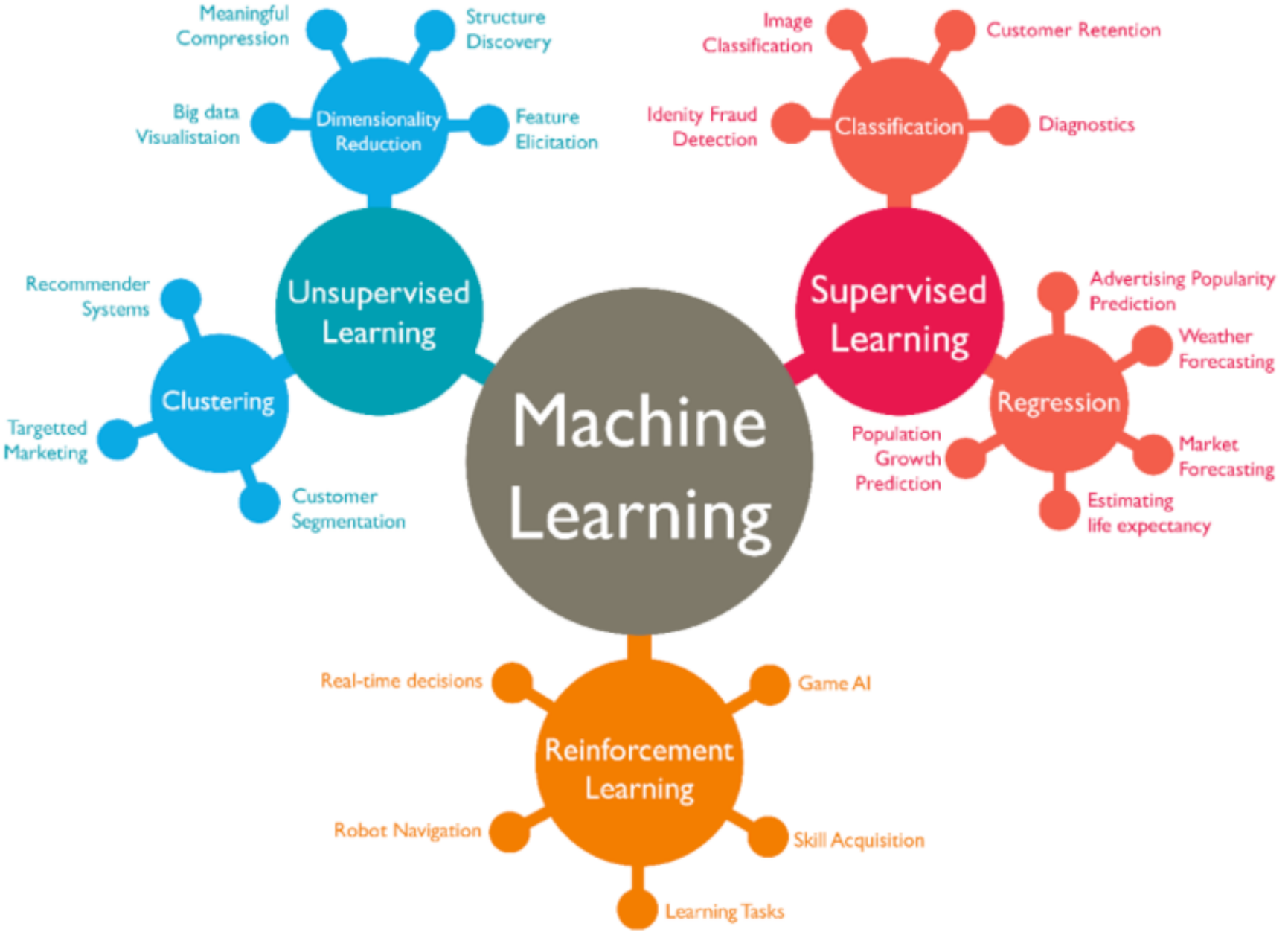


MACHINE LEARNING & DATA MINING

image from: lenta.ru


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks

Supervised learning setup

Dataset:

$$\mathcal{D} : \{(x_i, y_i), i = 1 \dots n\}$$

x_i – features, $x_i = \{x_i^{(1)}, \dots, x_i^{(d)}\}$, $x_i^{(m)} \in \mathbb{X}^{(m)}$

y_i – targets

$y_i \in \mathbb{R}$ – regression task

$y_i \in \{c_1, \dots, c_k\}$ – classification

Assume the samples are i.i.d.

We want to «learn» to predict y from x

Supervised learning setup

Dataset:

$$\mathcal{D} : \{(x_i, y_i), i = 1 \dots n\}$$

x_i – features, $x_i = \{x_i^{(1)}, \dots, x_i^{(d)}\}$, $x_i^{(m)} \in \mathbb{X}^{(m)}$

y_i – targets

$y_i \in \mathbb{R}$ – regression task

$y_i \in \{c_1, \dots, c_k\}$ – classification

Assume the samples are i.i.d.

We want to «learn» to predict y from x

algorithm

$$\mathcal{A} : \quad \mathcal{A}(\mathcal{D}) \to f$$

$$f : \quad f(x) \to \hat{y}$$

$$\mathcal{M}(f(\cdot), \mathcal{D}') \to max$$

metric ('score' = –'loss')

data not seen by the algorithm

Kinds of features

$x_i = \{x_i^{(1)}, \dots, x_i^{(d)}\}$, $x_i^{(m)} \in \mathbb{X}^{(m)}$

Numeric:

- $\mathbb{X}^{(m)} = \mathbb{R}$ (or e.g. $\mathbb{R}^+$ etc.) – continuous
- $\mathbb{X}^{(m)} = \mathbb{N}$ (or e.g. $\mathbb{N}^+$ etc.) – discrete

Categorical:

- $X^{(m)} = \{0, 1\}$ – binary feature
- $|X^{(m)}| < \infty$ – categorical
  - nominal – unordered (e.g. CountryOfBirth)
  - ordinal – values can be compared, though difference between values undefined (e.g. LevelOfEducation)

Linear Models

Prediction is a linear combination of features:

$$\hat{y}_i = \sum_j w_j \cdot x_i^{(j)} + b$$

We can express the data in terms of design matrix:

$$X \equiv X_{ij} = x_i^{(j)}$$

Then our estimator is:

$$\hat{y} = Xw$$

column-vector of target predictions

column-vector of parameters (weights)

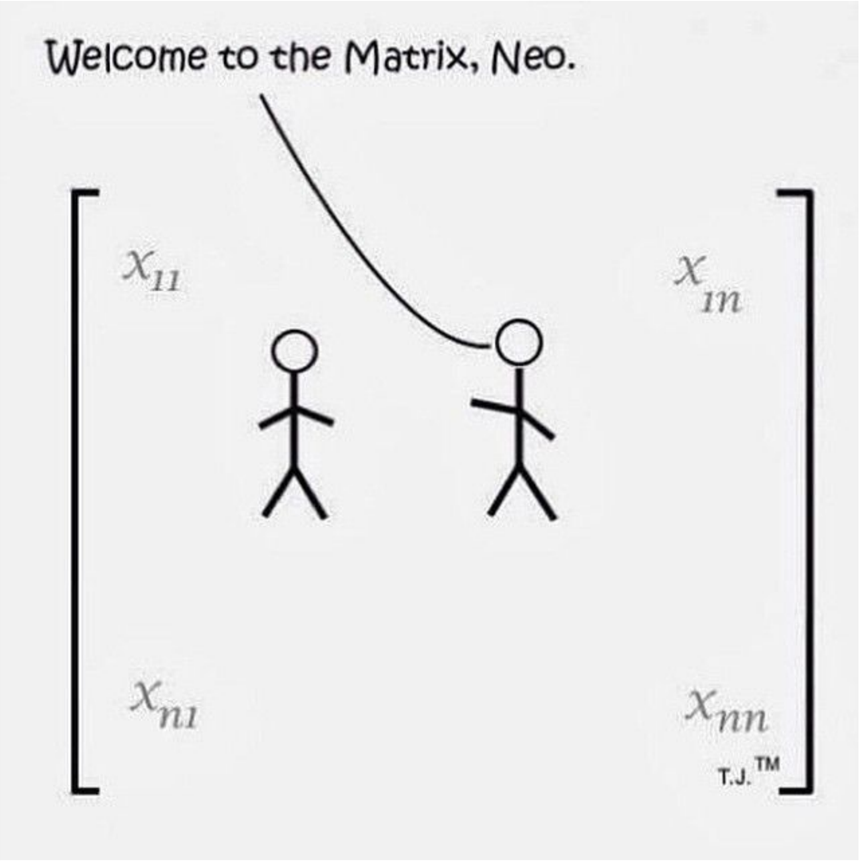


image from: mathmaniacs (Instagram)

Linear Regression

Linear model with quadratic loss:

$$\mathcal{L} = \frac{1}{n}\sum_i (y_i - \hat{y}_i)^2 = \frac{1}{n}||y - \hat{y}||^2$$

$$= \frac{1}{n}||y - Xw||^2 \to min$$

mean squared error (MSE)

Linear Regression

Analytical solution:

$$0 = \nabla_w \mathcal{L} = \nabla_w \frac{1}{n} ||y - Xw||^2$$

$$= \frac{1}{n} \nabla_w (y - Xw)^T (y - Xw)$$

$$= \frac{1}{n} \nabla_w \left[y^T y - 2y^T Xw + w^T X^T Xw \right]$$

$$= \frac{1}{n} \left[0 - 2X^T y + (X^T X + (X^T X)^T) w \right]$$

$$\Rightarrow X^T Xw - X^T y = 0$$

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

Linear Regression

Analytical solution:

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

Linear Regression

Analytical solution:

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

Problems:

- Expensive matrix inversion – $\mathrm{O}(d^3)$
- $X^T X$ may be singular or ill-conditioned

Linear Regression

Analytical solution:

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

Problems:

- Expensive matrix inversion – $\mathrm{O}(d^3)$
- $X^T X$ may be singular or ill-conditioned

Alternative approaches:

- scikitlearn implementation uses SVD method – $\mathrm{O}(nd^2)$
- gradient descent – can be used with any differentiable loss function

Going ‘non-linear’

- Say we want to fit our data with n-th degree polynomial ($d = 1$)
- Can we do so with linear regression?

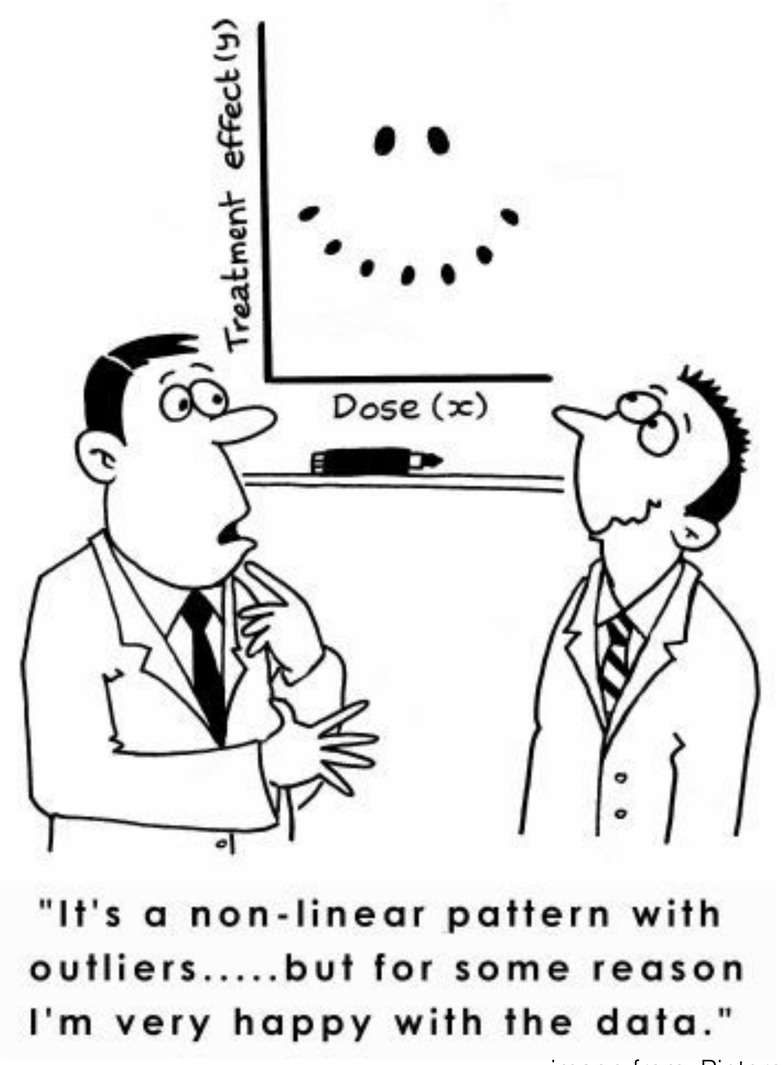


image from: Pinterest

Going ‘non-linear’

- Say we want to fit our data with n-th degree polynomial ($d = 1$)
- Can we do so with linear regression?

- Yes! Just introduce new features:

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \rightarrow X = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \dots & (x_1)^p \\ \vdots & & & & \\ 1 & x_n & (x_n)^2 & \dots & (x_n)^p \end{bmatrix}$$

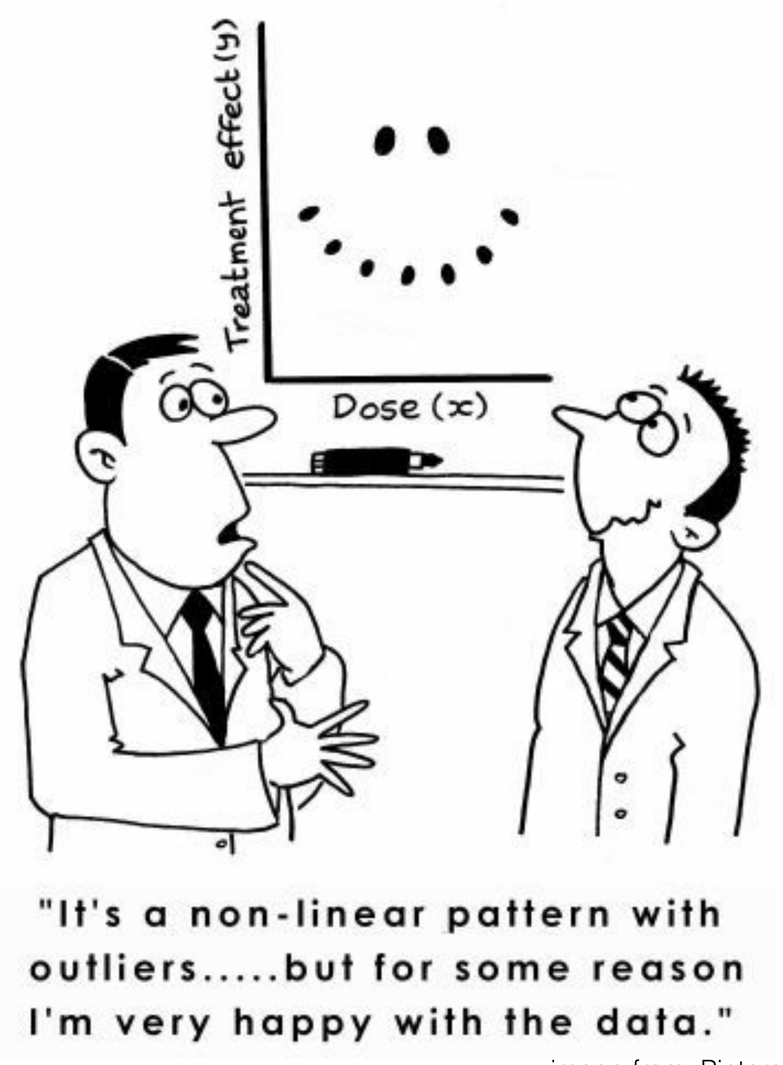


image from: Pinterest

Going ‘non-linear’

- More generally:

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \rightarrow X = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_p(x_1) \\ \vdots & & & \\ \phi_1(x_n) & \phi_2(x_n) & \dots & \phi_p(x_n) \end{bmatrix}$$

Probabilistic approach

- The i.i.d. assumption - i.e. all data objects are independently drawn from some distribution:

$$(x_i, y_i) \sim P^{\text{true}}(x, y)$$

- Now, we want to estimate conditional $P^{\text{true}}(y \mid \text{x})$ with a parametric distribution $P(y \mid \text{x}, \theta)$ using maximum likelihood approach

$$L = \Pi_{i=1}^{n} P(y_i | x_i, \theta) \rightarrow_{\theta} max$$

Probabilistic approach

- Let's assume $P^{\text{true}}(y \mid \mathbf{x})$ is a linear dependence plus normally distributed noise. Our estimator is thus:

$$P\left(y|x,\theta\right) = \mathcal{N}(y|w^T x, \sigma^2)$$

$$\theta \equiv (w, \sigma)$$

$$L \to max \Longleftrightarrow -logL \to min$$

Probabilistic approach

$$
\begin{aligned}
-logL &= -\sum log\left(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}}\right) \\
&= C + n\,log\sigma + \sum \frac{(y_i - w^T x_i)^2}{2\sigma^2} \rightarrow min
\end{aligned}
$$

Probabilistic approach

$$-logL = -\sum log\left(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}}\right)$$

$$= C + n\,log\sigma + \sum \frac{(y_i - w^T x_i)^2}{2\sigma^2} \to min$$

$$\Updownarrow$$

$$\sum (y_i - w^T x_i)^2 \to min$$

Probabilistic approach

$$-logL = -\sum log\left(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}}\right)$$

$$= C + n\,log\sigma + \sum \frac{(y_i - w^T x_i)^2}{2\sigma^2} \to min$$

Note: same logic would apply for any deterministic function + noise

$$\Updownarrow$$

$$\sum (y_i - w^T x_i)^2 \to min$$

Probabilistic approach

$$-logL = -\sum log\left(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y_i - w^T x_i)^2}{2\sigma^2}}\right)$$

$$= C + n\,log\sigma + \sum \frac{(y_i - w^T x_i)^2}{2\sigma^2} \rightarrow min$$

Note: same logic would apply for any deterministic function + noise

$$\Updownarrow$$

$$\sum (y_i - w^T x_i)^2 \rightarrow min$$

Using MSE (or RMSE) loss is equivalent to assuming normal distribution of errors in the data in probabilistic approach

Other loss functions

$$\mathrm{MAE} = \frac{1}{n}\sum_i |y_i - \hat{y}_i|$$

«mean average error»

Question: what error distribution would lead to this loss with the max. likelihood approach?

Other loss functions

$$\mathrm{MAPE} = \frac{1}{n}\sum_i \left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

«mean average percentage error»

$$\mathrm{MSLE} = \frac{1}{n}\sum_i \left(\log(y_i + 1) - \log(\hat{y}_i + 1)\right)^2$$

«mean squared logarithmic error»

A little Bayes

- In Bayesian approach we treat model parameters as random variables as well
- Then, prior distribution $P(\theta)$ reflects our beliefs of what values of θ are likely to be (before we observe any data)
- These beliefs are updated as we get new experience (training data):

likelihood

posterior

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)}$$

prior

Maximum a posteriori

- Let's consider a linear model with a prior on weights $P(w) = \mathcal{N}(0, \sigma_w^2 I)$
- Assume normal distribution of errors (as we did with max. likelihood approach)
- Then our posterior is:

$$P(w \mid X, y) \sim \mathcal{N}(y \mid Xw, \sigma^2 I)\, \mathcal{N}(w \mid 0, \sigma_w^2 I)$$

- Our estimator is maximum of this expression over w

Maximum a posteriori

$$
\begin{aligned}
\mathrm{MAP} &= \arg\max_w \left[\frac{1}{(\sqrt{2\pi}\sigma)^n} e^{-\frac{||y-Xw||^2}{2\sigma^2}} \cdot \frac{1}{(\sqrt{2\pi}\sigma_w)^d} e^{-\frac{||w||^2}{2\sigma_w^2}} \right] \\
&= \arg\min_w \left[n \log \sigma + \frac{||y-Xw||^2}{2\sigma^2} + d \log \sigma_w + \frac{||w||^2}{2\sigma_w^2} \right] \\
&= \arg\min_w \left[||y-Xw||^2 + \frac{\sigma^2}{\sigma_w^2} ||w||^2 \right]
\end{aligned}
$$

Maximum a posteriori

$$
\begin{aligned}
\mathrm{MAP} &= \arg\max_w \left[\frac{1}{(\sqrt{2\pi}\sigma)^n} e^{-\frac{||y-Xw||^2}{2\sigma^2}} \cdot \frac{1}{(\sqrt{2\pi}\sigma_w)^d} e^{-\frac{||w||^2}{2\sigma_w^2}} \right] \\
&= \arg\min_w \left[n\log\sigma + \frac{||y-Xw||^2}{2\sigma^2} + d\log\sigma_w + \frac{||w||^2}{2\sigma_w^2} \right] \\
&= \arg\min_w \left[\underbrace{||y-Xw||^2}_{\text{MSE}} + \underbrace{\frac{\sigma^2}{\sigma_w^2}||w||^2}_{\text{regularization}} \right]
\end{aligned}
$$

Ridge regression

$$\min_w \left[||y - Xw||^2 + \alpha ||w||^2 \right]$$

- Regularizes ill defined optimization problem by penalizing large weight values
  - Ensures the uniqueness of minimum (strict convexity)

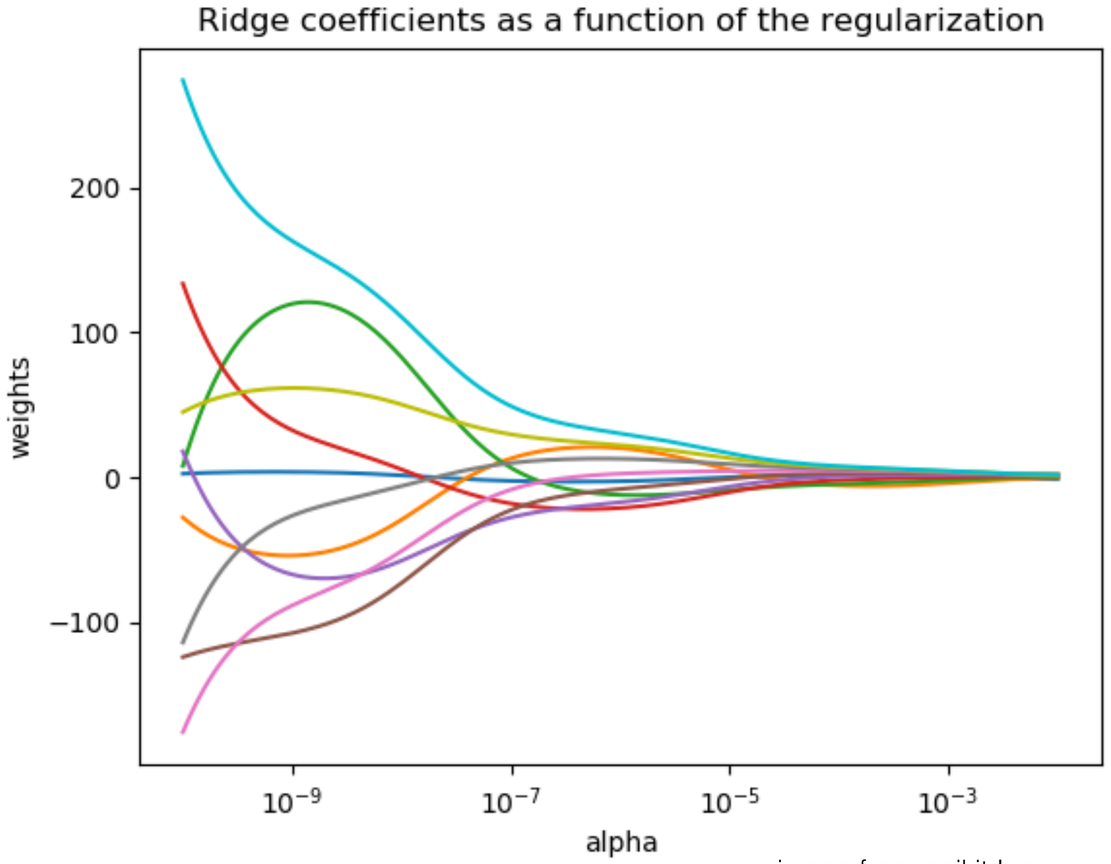


image from: scikit-learn.org

Lasso

(least absolute shrinkage and selection operator)

$$\min_{w} \left[||y - Xw||^2 + \alpha ||w||_1 \right]$$

- Regularizes by forcing some weights to zero (sparsification)

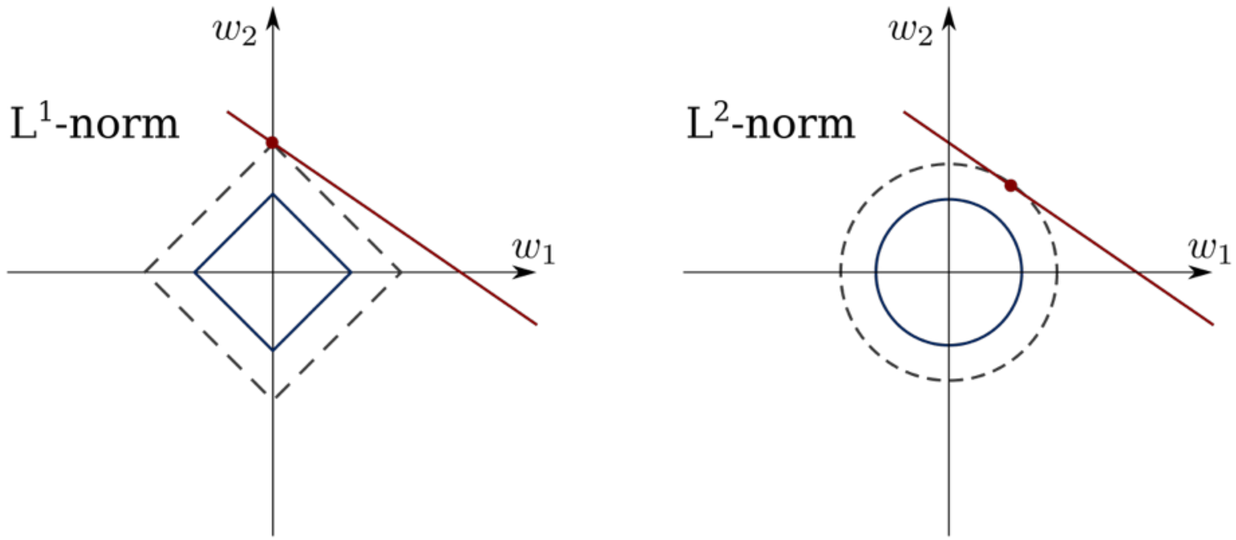


image from: wikipedia.org

Elastic Net

$$\min_w \left[||y - Xw||^2 + \alpha_1 ||w||_1 + \alpha_2 ||w||^2 \right]$$

- Combination of the two
- Ability to control the relative contribution of L1 and L2 penalties

Logistic Regression

- Regression of the probability belonging to a class
  - (so actually used for classification)
- Class probability is parametrized with a combination of linear function and logistic function:

$$P(y_i \mid x_i, w) = \frac{1}{1 + e^{-y_i \, w^T x_i}}$$

$$y_i \in \{-1, 1\}$$

Logistic Regression

- With maximum likelihood approach:

$$L = \prod \frac{1}{1 + e^{-y_i \, w^T x_i}} \longrightarrow_w \max$$

$$-\log L = \sum \log(1 + e^{-y_i \, w^T x_i}) \longrightarrow_w \min$$

logistic loss