Deep Learning

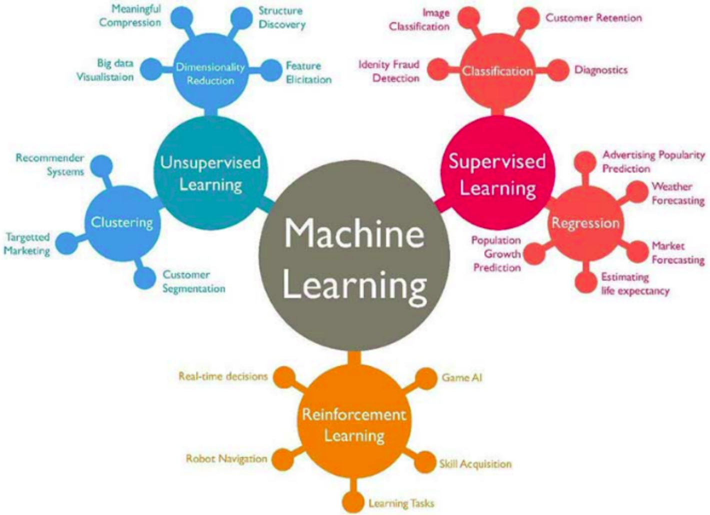
2. Main concepts of supervised learning. Linear regression.

Viacheslav Dudar

Taras Shevchenko National University of Kyiv

2018

Machine learning chapters



Supervised learning

Task of learning a function that maps an input to an output based on example input-output pairs.

Training data: $\{(input_1, output_1), \ldots, (input_n, output_n)\}$
Aim: build a function $f : input \rightarrow output$

- Regression problem: outputs are continuous
- Classification problem: outputs are dicrete

Types of possible inputs

- Single real number: $input \in R$
- Multiple real numbers $input \in R^m$ (vector, image)
- Sequence of varying length (text, sound, video)
- Binary / Disrete elements
- Combination of elements above

How to find mapping

Approach: consider parametric mapping from input to output.
Find parameters during training.

$$f(input, parameters) \rightarrow output$$

Find parameters such that:

$$f(input_1, parameters) \approx output_1$$

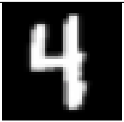
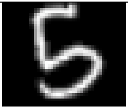
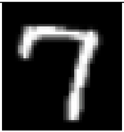
$$f(input_2, parameters) \approx output_2$$

$$\dots$$

$$f(input_n, parameters) \approx output_n$$

Example 1: Classification

MNIST training dataset:

Number	Input	Output
1		4
2		5
...	...	...
60000		7

Input: image $28 * 28$ (we can view it as $R^{28*28} = R^{784}$)

Output: digit label $\{0, 1, 2...9\}$

Example 1: Classification

Aim: find parameters of the function f such that:

$f($$) = 8$

for the new examples (that are not in the training set)

How to test performance?

Training (learning) stage: find parameters w_* such that

$$f(input_i, w_*) \approx output_i$$

To check performance we need separate test set:
Check that $f(new\_input, w_*) = correct\_output$
Ability to classify new inputs correctly is called **generalization**

Overall process overview

We have:

- **Training set** (many input-output pairs)
- **Test set** (input-output pairs, separate from training set)

Build a **model** with unknown **parameters** w: $f(input, w)$
Learning: Find parameters w_* such that $f(input, w_*)$ is close to outputs of training set.
Evaluating the model: check if $f(input, w_*)$ is close to outputs of the test set.
Testing stage: find $f(input, w_*)$ for inputs we do not know correct outputs.

What remains?

- Specify the model $f(input, w)$
- Find a way to search for unknown coefficients w
- Find ways to improve generalization

Simplest model: linear regression

Linear model of inputs:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

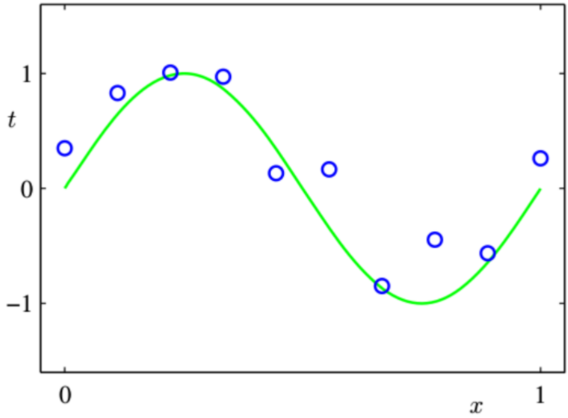
Extension using nonlinear **basis** functions:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

To get rid of bias we introduce additional function:
$\phi_0(\mathbf{x}) = 1$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

Illustration



$$y(x) = sin\left(2\pi x\right) + \varepsilon$$

Model

$$\boldsymbol{x} = (x_1, ..., x_N)^T$$

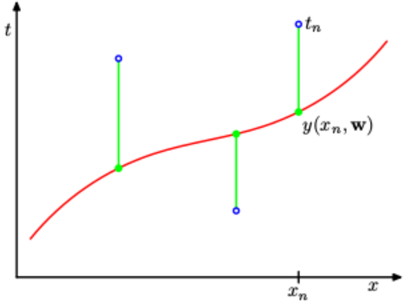
$$\boldsymbol{t} = (t_1, ..., t_N)^T$$

$$y(x, w) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M$$

Error function

Error function to minimize:

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2.$$



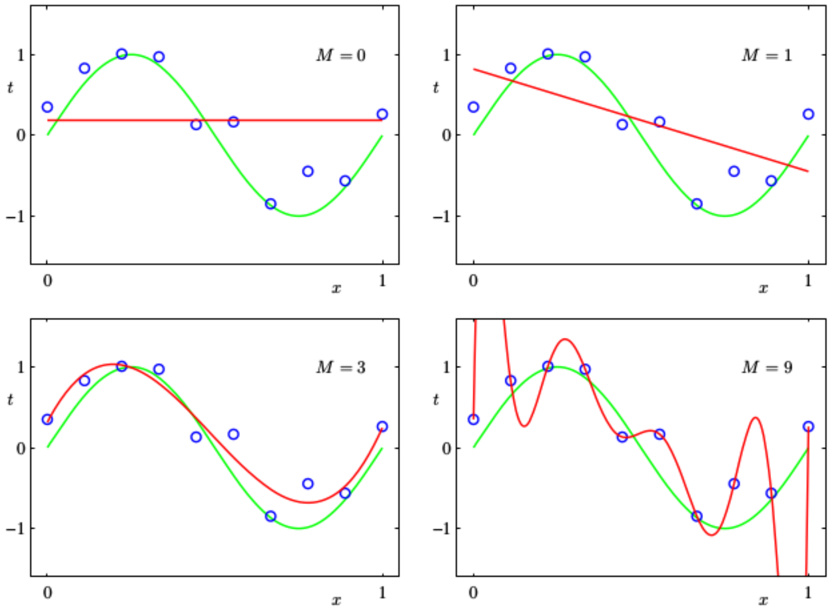
Solution

$$t = (t_1, ..., t_N)$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

Solutions for different models

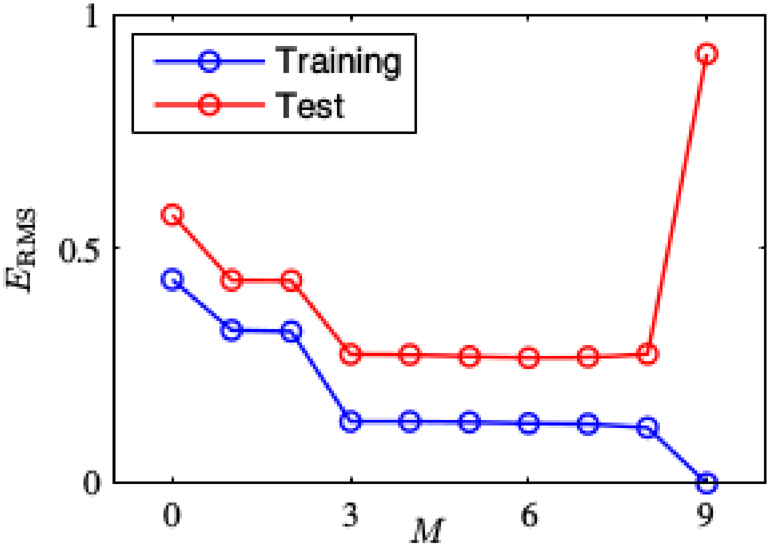


Solutions for different models

- M=0 and M=1: model is not compicated enough
- M=3: model is ok
- M=9: model is too complicated. It is not generalizing to the new points

Model M=9 is **overfitting** the data (has low training error but high test error).

Train and test errors:



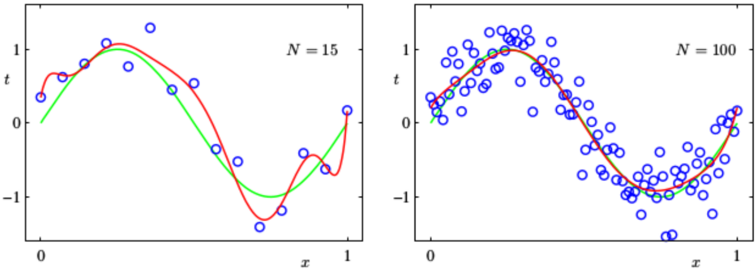
How to deal with overfitting?

In general, models with more parameters are more flexible and can approximate more complex relations, but tend to overfit more.

- Find model that has moderate complexity (so there is a balance between under and overfitting
- Get more data
- Change a complex model in such a way that it overfits less

This is where the **regularization** comes.

More data



Regularization

Coefficients for different models:

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	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

Regularized solution

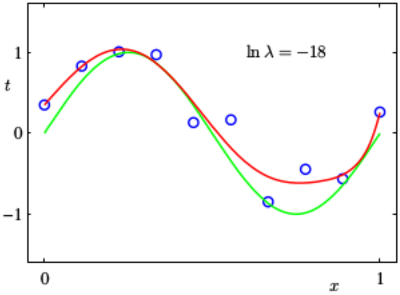
Large models have large coefficients. Can we decrease them?

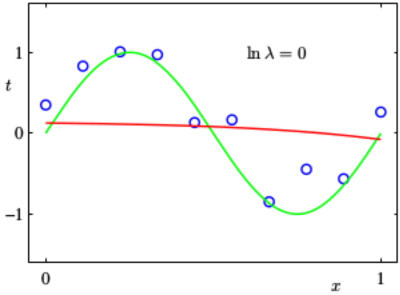
$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\mathbf{w} = \left(\lambda\mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}.$$

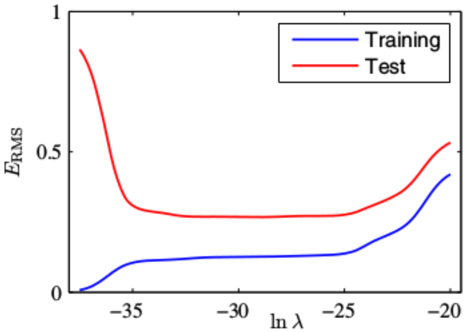
Regularized curve

Different regularization coefficients:



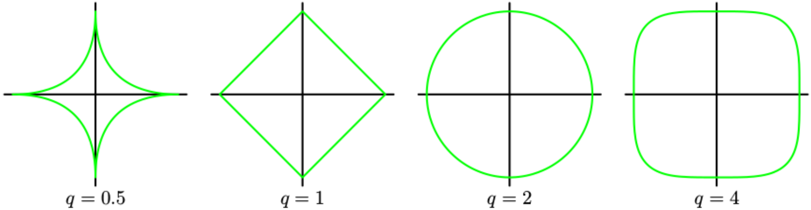


Regularization coefficient effect

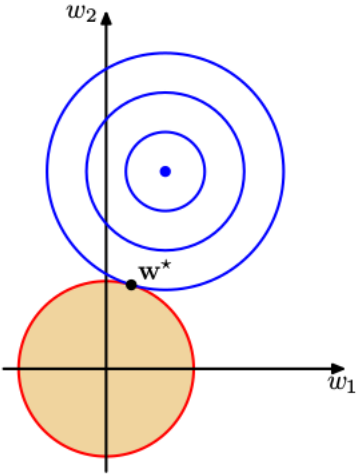


More general regularizations

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$



l1 regularization effect



Fixed basis functions limitations

Advantage of linear models: closed-form solution (not applicable in high-dimensional case).
Disadvantage: Need to find hand-crafted features. More advanced models (like neural networks) find feature vector themselves.

Bayesian approach

$$p(t|x, w, \beta) = N(y(x, w), \beta^{-1})$$

$$p(\boldsymbol{t}|\boldsymbol{x}, w, \beta) = \prod_{n=1}^{N} N\left(t_n|y(x_n, w), \beta^{-1}\right)$$

$$\ln p(\boldsymbol{t}|\boldsymbol{x}, w, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}(y(x_n, w) - t_n)^2 + \frac{N}{2}\ln(\beta) - \frac{N}{2}\ln(2\pi)$$

Maximum likelihood $\to \max \iff$ Sum of squares $\to \min$

Bayesian approach

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

α: hyperparameter

$$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha)$$

Maximum posterior, or MAP solution

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}.$$

MAP $\rightarrow \max \iff$ Regularized sum of squares with $\lambda = \frac{\alpha}{\beta} \rightarrow \min$

Bias-variance decomposition

$$
\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right] \\
&= \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}.
\end{aligned}
$$

Bias-variance decomposition