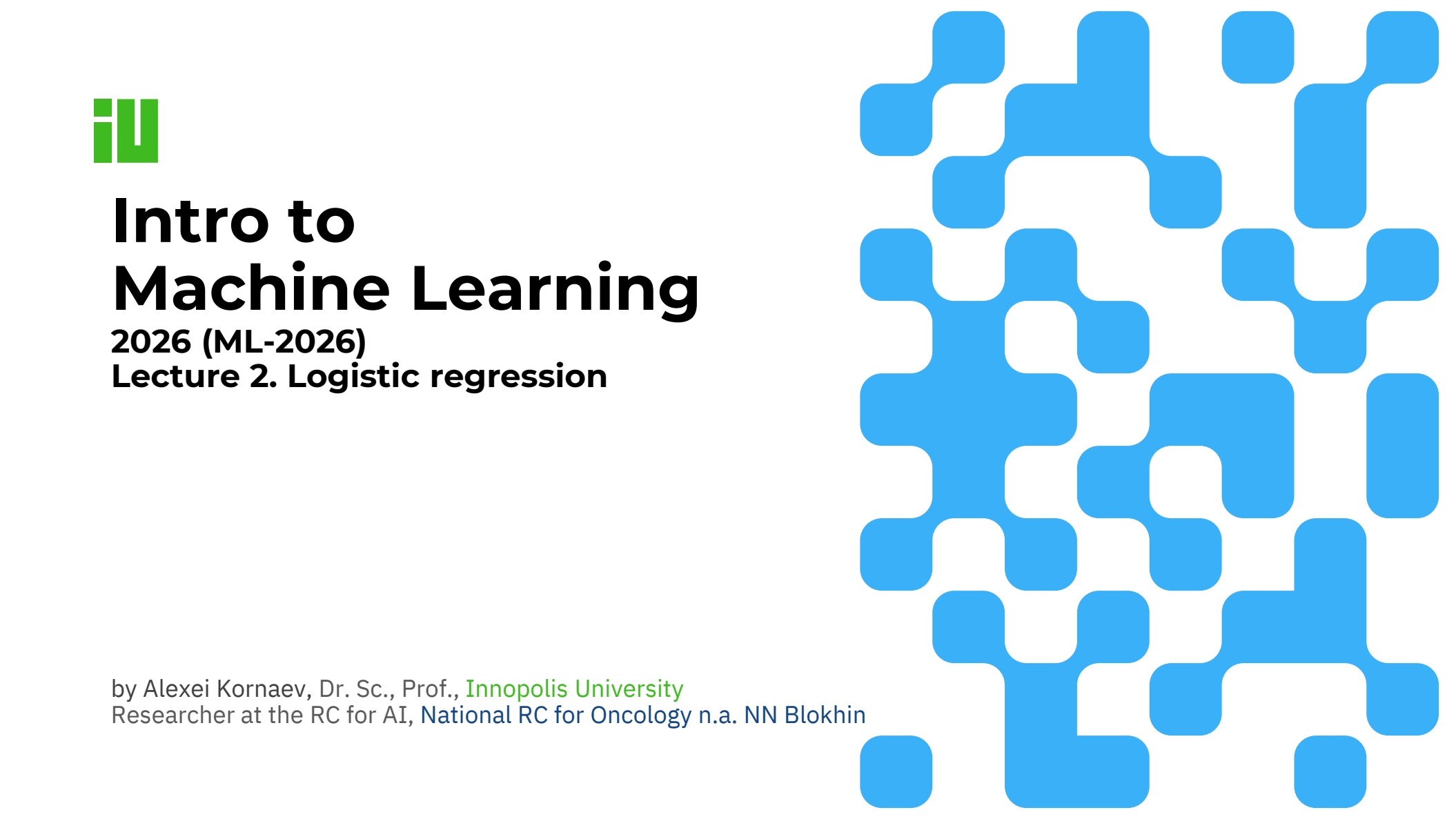


Intro to Machine Learning

2026 (ML-2026)

Lecture 2. Logistic regression

by Alexei Kornaev, Dr. Sc., Prof., Innopolis University
Researcher at the RC for AI, National RC for Oncology n.a. NN Blokhin

Agenda

I. Exploratory data analysis (EDA)

II. Logistic regressioin and its generalization

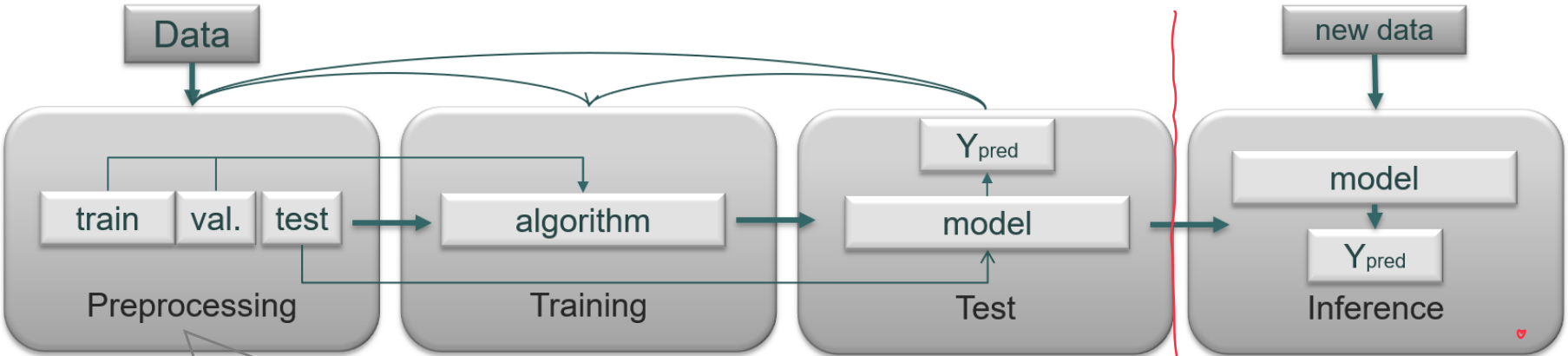
III. Metrics

IV. Setting of the models

V. Loss functions

All models are wrong, but some are useful.
/George Box/

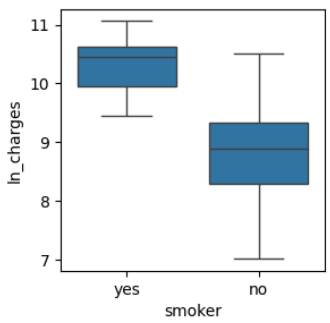
Flowchart for an ML model design (tabular data)

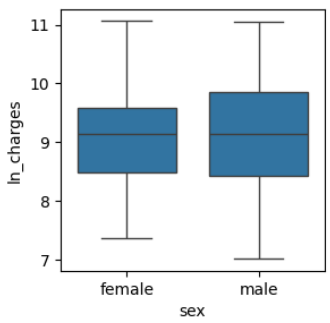


handling missing values
EDA: descriptive statistics, correlation,
encoding categorical features,
feature normalization,
dimensionality reduction of the features,
space/constructing new features,
splitting the dataset

Exploratory data analysis (EDA)

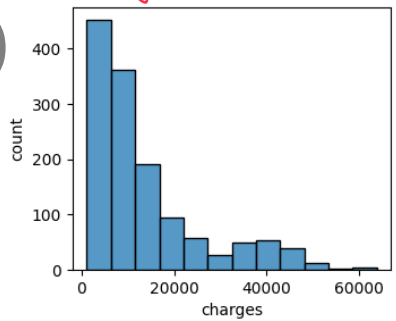
	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

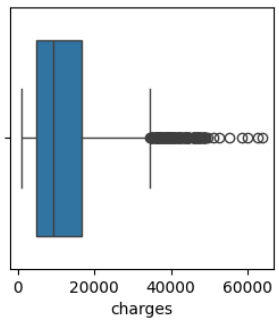


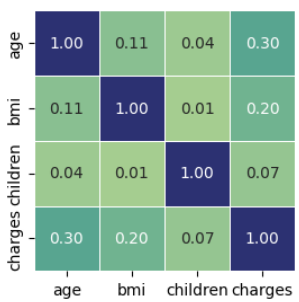


Medical insurance payout

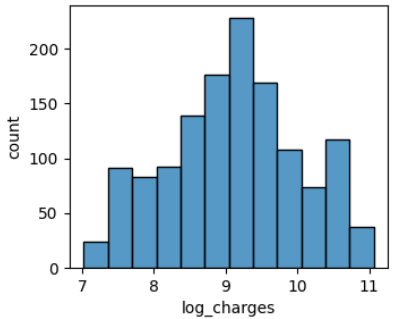
y

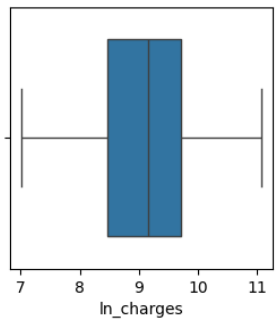


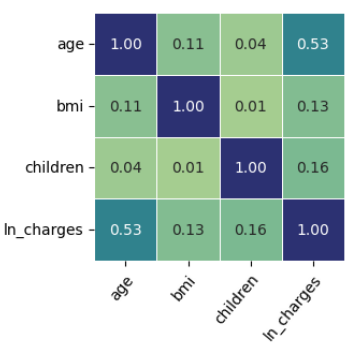




$\tilde{y} = ln(1 + y)$







Encode categorical features

```
print(data.region.unique())
['southwest', 'southeast', 'northwest', 'northeast']
```

```
pd.get_dummies(data[['region']], dtype=int)
```

region	region_northeast	region_northwest	region_southeast	region_southwest
southwest	0	0	0	1
southeast	0	0	1	0
southeast	0	0	1	0
northwest	0	1	0	0
northwest	0	1	0	0

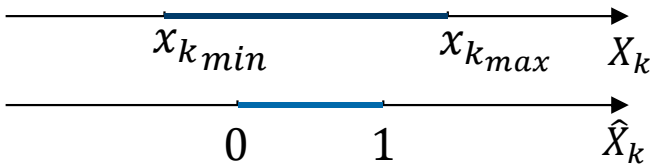
⚠ warning: $x_4 = 1 - \sum_{i=1}^{3} x_i$

```
pd.get_dummies(data[['region']], dtype=int,
drop_first=True)
```

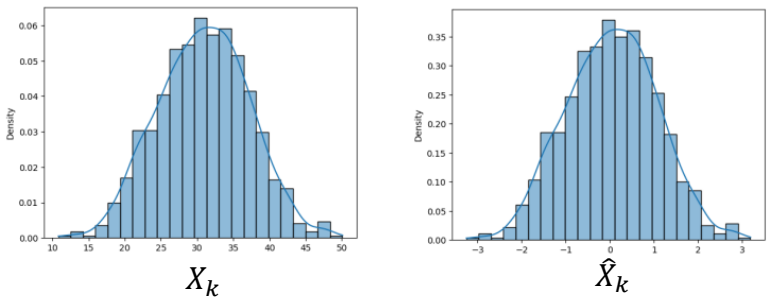
region_northwest	region_southeast	region_southwest
0	0	1
0	1	0
0	1	0
1	0	0
1	0	0

Normalize features

1. MinMaxScaler: $\hat{X}_k = \frac{X_k - x_{k_{min}}}{x_{k_{max}} - x_{k_{min}}}$



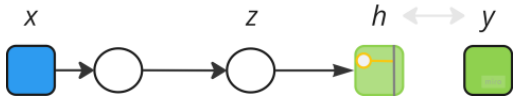
2. StandardScaler: $\hat{X}_k = \frac{X_k - \bar{x}_k}{S_k}$




```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_num_norm = scaler.fit_transform(data[['age', 'bmi','children']])
```

	sex_male	smoker_yes	region_northwest	region_southeast	region_southwest	age_norm	bmi_norm	children_norm	ln_charges
0	0.0	1.0	0.0	0.0	1.0	0.021739	0.321227	0.0	9.734236
1	1.0	0.0	0.0	1.0	0.0	0.000000	0.479150	0.2	7.453882
2	1.0	0.0	0.0	1.0	0.0	0.217391	0.458434	0.6	8.400763
3	1.0	0.0	1.0	0.0	0.0	0.326087	0.181464	0.0	9.998137
4	1.0	0.0	1.0	0.0	0.0	0.304348	0.347592	0.0	8.260455

Recap: Linear Regression



Model predicts output *h* given input *x*

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

$$\boldsymbol{x} = \begin{bmatrix} x^{(1)} \\ \dots \\ x^{(m)} \end{bmatrix} \rightarrow \begin{bmatrix} 1\ x^{(1)} \\ \dots\ \dots \\ 1\ x^{(m)} \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}, \boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}.$$

$$L(\boldsymbol{\phi}) = \frac{1}{2m}\sum_{i=1}^{m}\left(h^{(i)} - y^{(i)}\right)^2 \Rightarrow \min.$$

$$\nabla L = \left[\frac{\partial L}{\partial \phi_j}\right] = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y}).$$

Training algorithm.

1. Initialize the *weights* $\boldsymbol{\phi}$ with a random seed
2. Calculate the *hypothesis* matrix $\boldsymbol{h} = \boldsymbol{x}\boldsymbol{\phi}$ and the *loss gradient*: $\nabla L = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y})$
3. For the given ϕ_j components (annotated with idex 'prev', ϕ_j^{prev}) calculate the newer ones ϕ_j^{next} moving towards the direction, which is opposite to the loss gradient vector, with steps which are proportional to the *learning rate* α:
$\boldsymbol{\phi}^{next} = \boldsymbol{\phi}^{prev} - \alpha\nabla L$, or in the scalar form $\phi_0^{\text{next}} = \phi_0^{prev} - \alpha\frac{\partial L}{\partial \phi_0}$, $\phi_1^{\text{next}} = \phi_1^{prev} - \alpha\frac{\partial L}{\partial \phi_1}$
4. Repeat pp. 2-3 until the minimum of the loss function L is reached, based on the condition of small changes in its value over several neighboring iterations or based on the condition of reaching the maximum number of iterations : $L^{next} - L^{prev} < \delta$, #iter. > max # of iter
5. Save the trained model (model weights): $\boldsymbol{\phi}$.

Agenda

I. Exploratory data analysis (EDA)

II. Logistic regression and its generalization

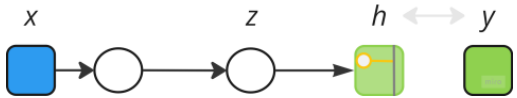
III. Metrics

IV. Setting of the models

V. Loss functions

All models are wrong, but some are useful.
/George Box/

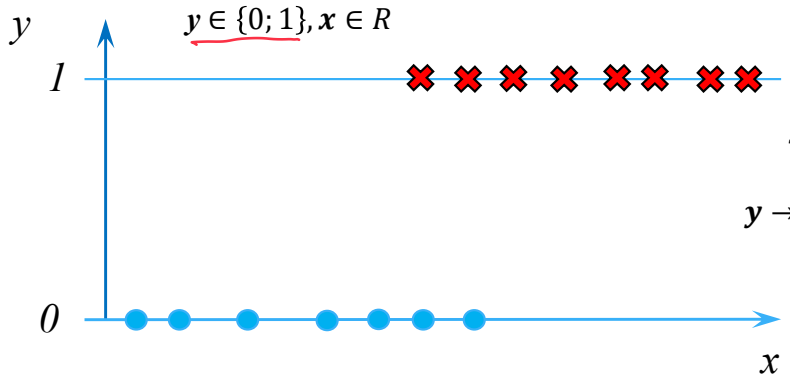
Logistic Regression



Model predicts output *h* given input *x*

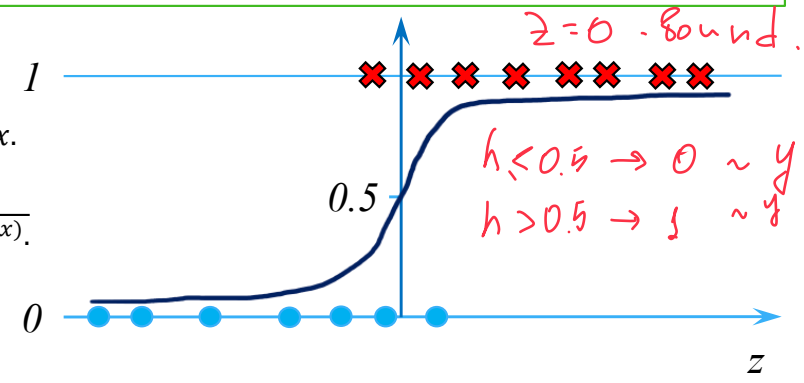
Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

$$L(\boldsymbol{\phi}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\ln(h^{(i)}) + (1-y^{(i)})(\ln(1-h^{(i)})\right) \Rightarrow \min.$$



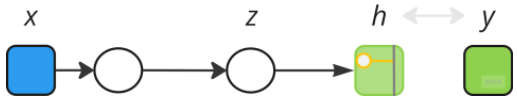
$$\boldsymbol{x} \to \boldsymbol{z}\colon\ z = \phi_0 + \phi_1 x.$$

$$\boldsymbol{y} \to \boldsymbol{h}\colon\ h(z) = \frac{1}{1+e^{-z(x)}}.$$



$$\boldsymbol{x} = \begin{bmatrix} x^{(1)} \\ \dots \\ x^{(m)} \end{bmatrix}; \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}; \to \quad \boldsymbol{x} = \begin{bmatrix} 1\ x^{(1)} \\ \dots\ \dots \\ 1\ x^{(m)} \end{bmatrix}; \boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}; \boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi}; \quad \to \quad h(z) = \frac{1}{1+e^{-z(x)}}, \text{ or } \boldsymbol{h} = \sigma(\boldsymbol{z}).$$

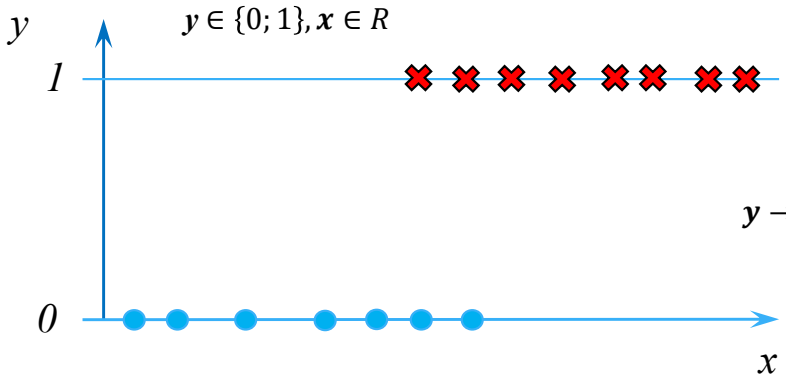
Logistic Regression



Model predicts output *h* given input *x*

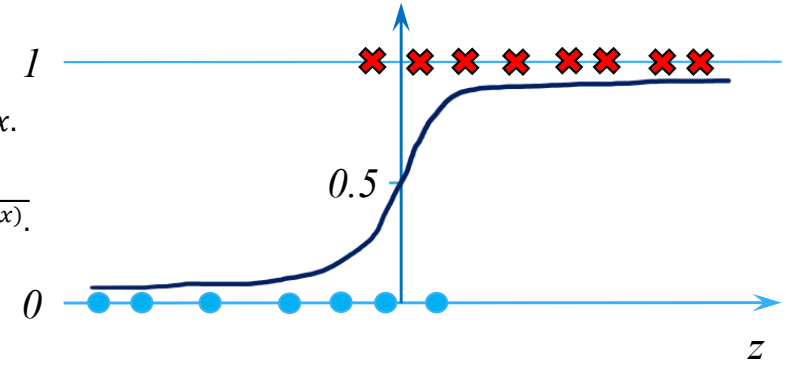
Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

$$L(\boldsymbol{\phi}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\ln(h^{(i)}) + (1-y^{(i)})(\ln(1-h^{(i)})\right) \Rightarrow \min.$$



$\boldsymbol{x} \to \boldsymbol{z}$: $z = \phi_0 + \phi_1 x.$

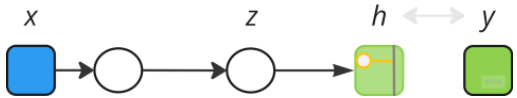
$\boldsymbol{y} \to \boldsymbol{h}$: $h(z) = \frac{1}{1+e^{-z(x)}}.$



Training algorithm.

1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi}$, $\boldsymbol{h} = \sigma(\boldsymbol{z})$, then $\nabla L = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y})$
3. Update $\boldsymbol{\phi}^{next} = \boldsymbol{\phi}^{prev} - \alpha\nabla L$
4. Repeat pp. 2-3 $L^{next} - L^{prev} < \delta$, #iter. > max # of iter
5. Save the trained model (model weights): $\boldsymbol{\phi}$.

Logistic Regression. Generalization (multiple var., polynomial)



Model predicts output *h* given input *x*

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

$$L(\boldsymbol{\phi}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\ln(h^{(i)}) + (1-y^{(i)})(\ln(1-h^{(i)})\right) \Rightarrow \min.$$

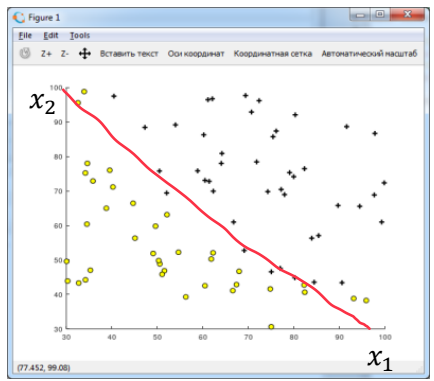
$$\boldsymbol{x} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ \dots & \dots \\ x_1^{(m)} & x_2^{(m)} \end{bmatrix}; \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}; \quad \rightarrow \quad \boldsymbol{x} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ \dots & \dots & \dots \\ 1 & x_1^{(m)} & x_2^{(m)} \end{bmatrix}; \boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \end{bmatrix}; \boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi}; \quad \rightarrow \quad h(z) = \frac{1}{1+e^{-z(x)}}, \text{ or } \boldsymbol{h} = \sigma(\boldsymbol{z}).$$

$$\varphi_0 + \varphi_1 x_1 + \varphi_2 x_2 = 0$$

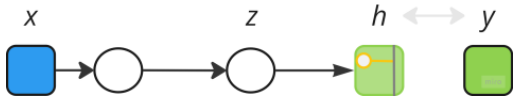
$$x_2 = -\frac{\varphi_0}{\varphi_2} - \frac{\varphi_1}{\varphi_2} x_1$$

Training algorithm.

1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi}$, $\boldsymbol{h} = \sigma(\boldsymbol{z})$, then $\nabla L = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y})$
3. Update $\boldsymbol{\phi}^{next} = \boldsymbol{\phi}^{prev} - \alpha\nabla L$
4. Repeat pp. 2-3 $L^{next} - L^{prev} < \delta$, #iter. > max # of iter
5. Save the trained model (model weights): $\boldsymbol{\phi}$.



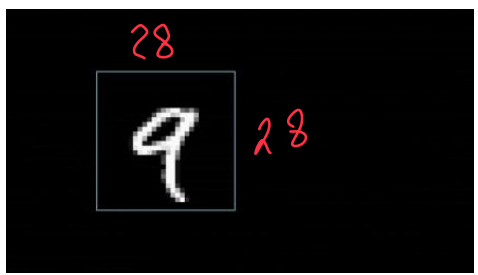
Logistic Regression. Generalization (multiple var., polynomial)



Model predicts output *h* given input *x*

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

$$L(\boldsymbol{\phi}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\ln(h^{(i)}) + (1-y^{(i)})(\ln(1-h^{(i)})\right) \Rightarrow \min.$$



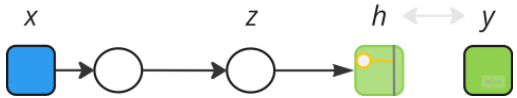
Gray scale picture of "Nine"

$$\boldsymbol{x} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ \dots & \dots & \dots & \dots \\ x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix}; \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}; \quad \rightarrow \boldsymbol{x} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix}; \boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \dots \\ \phi_n \end{bmatrix}; \boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi};\ \boldsymbol{h} = \sigma(\boldsymbol{z}).$$

Training algorithm.

1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi}$, $\boldsymbol{h} = \sigma(\boldsymbol{z})$, then $\nabla L = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y})$
3. Update $\boldsymbol{\phi}^{next} = \boldsymbol{\phi}^{prev} - \alpha\nabla L$
4. Repeat pp. 2-3 $L^{next} - L^{prev} < \delta$, #iter. > max # of iter
5. Save the trained model (model weights): $\boldsymbol{\phi}$.

Logistic Regression. Generalization (multiple var., polynomial)



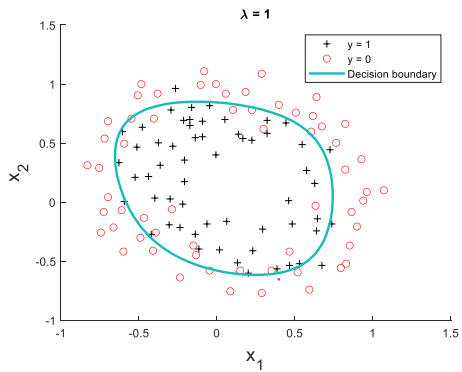
Model predicts output *h* given input *x*

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

$$L(\boldsymbol{\phi}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\ln(h^{(i)}) + (1-y^{(i)})(\ln(1-h^{(i)})\right) \Rightarrow \min.$$

$$\boldsymbol{x} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ \dots & \dots \\ x_1^{(m)} & x_2^{(m)} \end{bmatrix}; \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}; \quad \rightarrow \quad X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_1^{(1)}x_2^{(1)} & x_1^{(1)2} & x_2^{(1)2} & \dots \end{bmatrix}$$

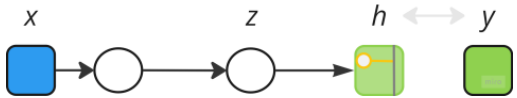
$z = 0$



Training algorithm.

1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi}$, $\boldsymbol{h} = \sigma(\boldsymbol{z})$, then $\nabla L = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y})$
3. Update $\boldsymbol{\phi}^{next} = \boldsymbol{\phi}^{prev} - \alpha\nabla L$
4. Repeat pp. 2-3 $L^{next} - L^{prev} < \delta$, #iter. > max # of iter
5. Save the trained model (model weights): $\boldsymbol{\phi}$.

Logistic Regression. Generalization (multiple classes)



Model predicts output *h* given input *x*

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

$$L(\boldsymbol{\phi}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\ln(h^{(i)}) + (1-y^{(i)})(\ln(1-h^{(i)}))\right) \Rightarrow \min.$$

$$\boldsymbol{x} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & \dots & \dots \\ 1 & x_1^{(m)} & x_2^{(m)} \end{bmatrix}; \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \rightarrow$$

$$\varphi = \begin{bmatrix} \varphi_{00} & \varphi_{01} & \varphi_{02} \\ \varphi_{10} & \varphi_{11} & \varphi_{12} \\ \varphi_{20} & \varphi_{21} & \varphi_{22} \end{bmatrix}$$

$z_1 = \varphi_{00} + \varphi_{10}x_1 + \varphi_{20}x_2$

$z_2 = \dots$

$z_3 = \dots$

$z_k \rightarrow h_k \rightarrow y_k$

$$h_k = \frac{e^{z_k}}{\sum_i e^{z_i}} \rightarrow \text{softmax}$$

$$\sum_k h_k = 1$$

Training algorithm.
1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi}$, $\boldsymbol{h} = \sigma(\boldsymbol{z})$, then ∇L
3. Update $\boldsymbol{\phi}^{next} = \boldsymbol{\phi}^{prev} - \alpha\nabla L$
4. Repeat pp. 2-3 $L^{next} - L^{prev} < \delta$, #iter. > max # of iter
5. Save the trained model (model weights): $\boldsymbol{\phi}$.

Agenda

I. Exploratory data analysis (EDA)

II. Logistic regression and its generalization

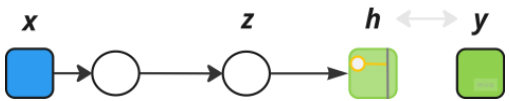
III. Metrics

IV. Setting of the models

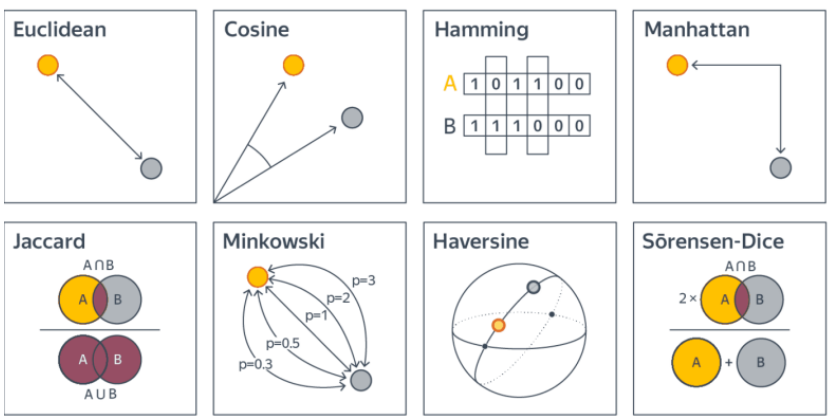
V. Loss functions

All models are wrong, but some are useful.
/George Box/

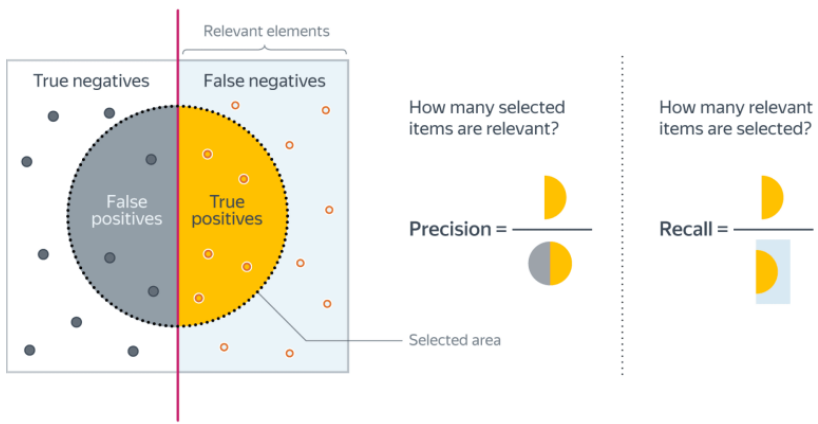
Metrics can be used both in learning (metric learning) and in estimating a model's quality



Model predicts output ***h*** given input ***x***

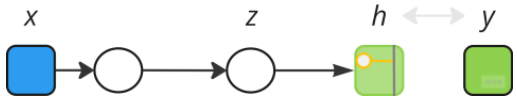


[Metrics from the Yandex Handbook]



[Classification metrics from the Yandex Handbook]

Metrics that estimate model's quality. Binary classification



Model predicts output *h* given input *x*

{0, 1}

1. Accuracy: $acc(\boldsymbol{h},\boldsymbol{y}) = \frac{1}{m}\sum_{i=1}^{m}\left(h^{(i)} == y^{(i)}\right)$, or (and) error rate: $error\ rate = 1 - \text{acc}(\boldsymbol{h},\boldsymbol{y})$

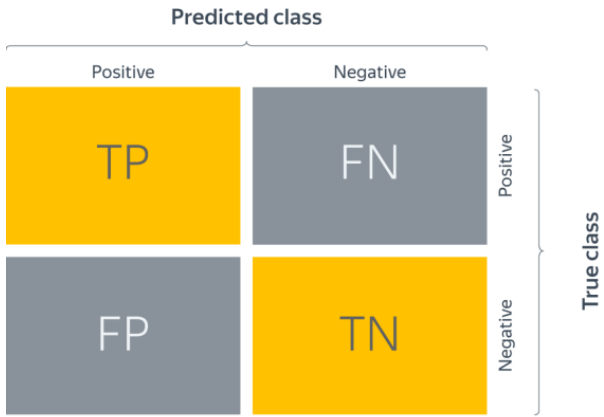
M1 $acc(\boldsymbol{h},\boldsymbol{y}) = \frac{TP+TN}{TP+TN+FP+FN}$. M2

$acc = 8/10$

Predicted	True
1	0
0	0
1	1
0	0
1	0
0	0
0	0
1	1
0	0
0	0

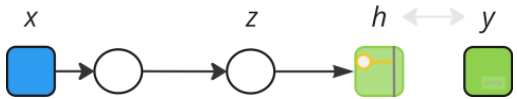
$acc = 8/10$

Predicted	True
0	1
0	0
0	0
0	0
0	1
0	0
0	0
0	0
0	0
0	0



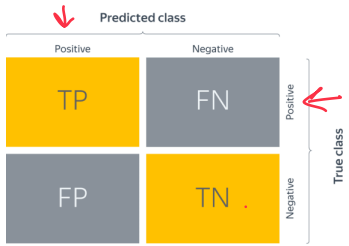
Confusion matrix

Metrics that estimate model's quality. Binary classification



Model predicts output *h* given input *x*

2. $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, $F_1 = 2\frac{Recall \cdot Precision}{Recall+Precision}$.



$pr = \frac{2}{4} : \frac{1}{2}$ $re = 1$ $F_1 = \frac{2}{3}$

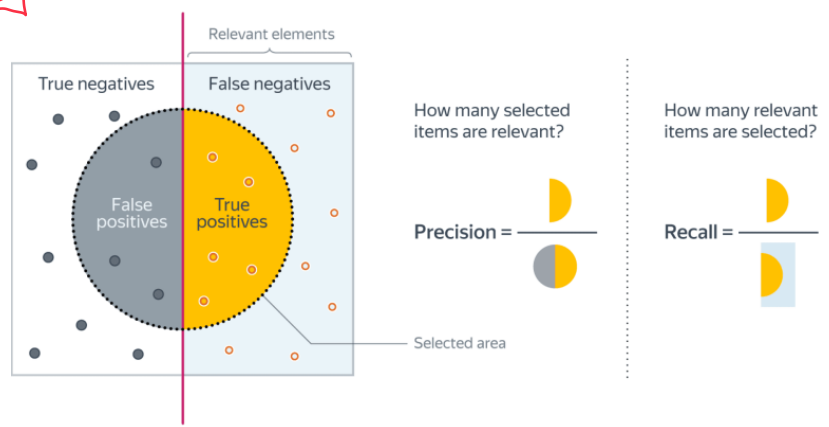
2	0
2	6

Predicted	True
1	0
0	0
1	1
0	0
1	0
0	0
0	0
1	1
0	0
0	0

$pr = 0$ $re = 0$ $F_1 = 0$

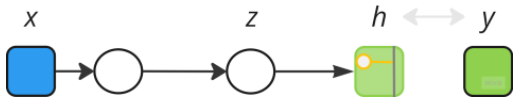
0	2
0	8

Predicted	True
0	1
0	0
0	0
0	0
0	1
0	0
0	0
0	0
0	0
0	0



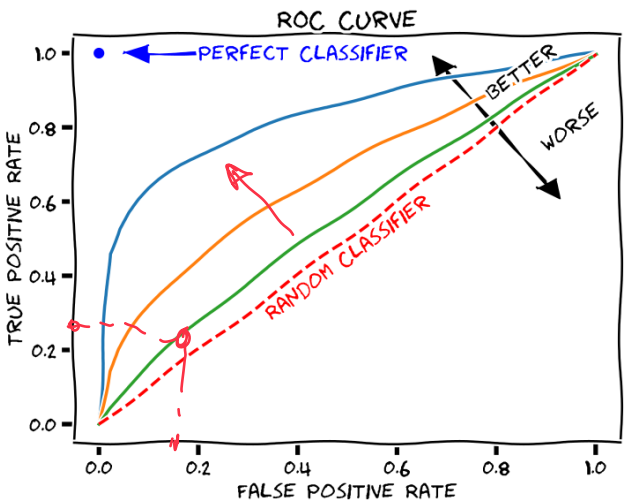
[Precision, recall, F1-score intuition]

Metrics that estimate model's quality. Binary classification



Model predicts output *h* given input *x*

1. True positive rate $TPR = Recall = \frac{TP}{TP+FN}$, false positive rate $FPR = \frac{FP}{FP+TN}$, Receiver operating characteristic (*ROC*), area under curve AUC.



[ROC curve intuition]

$tr = 0.5$

$0 \le tr \le 1$

AUC ↑

Overall, the optimization of precision and recall proceeds as follows:
- train the model on a loss function;
- obtain metric graphs depending on the threshold using real predictions on the validation set, by iterating over different thresholds from 0 to 1;
- select the desired combination of precision and recall.

Agenda

I. Exploratory data analysis (EDA)

II. Logistic regression and its generalization

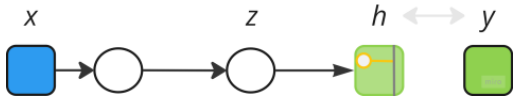
III. Metrics

IV. Setting of the models

V. Loss functions

All models are wrong, but some are useful.
/George Box/

Overfitting intuition and data splitting



Model predicts output *h* given input *x*

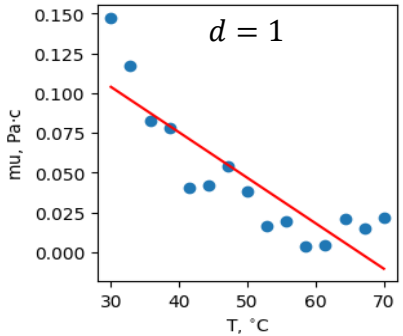
$$h = \theta_j x^j, (j = 0, .. d)$$

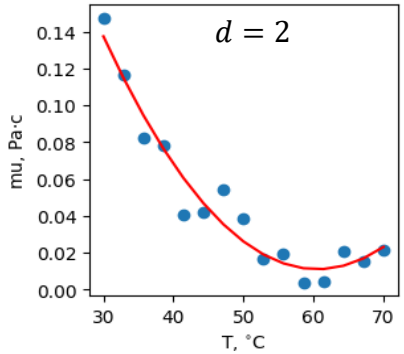
$$L = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h^{(i)} - y^{(i)}\right)^2\right] \Rightarrow \min.$$

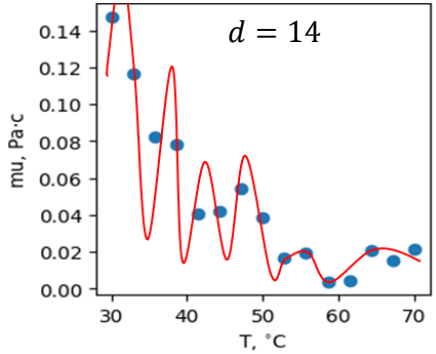
Model *parameters* are determined during the solution of the ML problem, e.g. model weights. *Hyperparameters* are set by the user, usually not in a single way, and their values affect the values of the sought parameters.

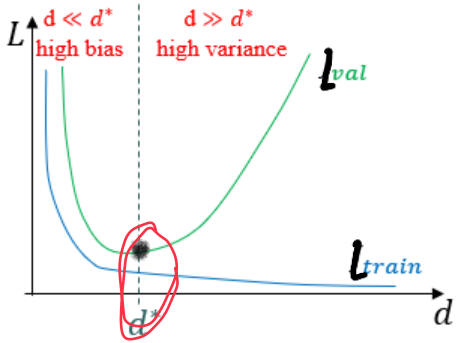


min val loss

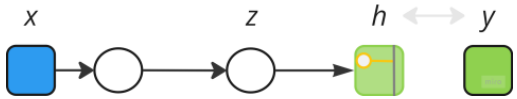








Overfitting intuition and regularization (L2)



Model predicts output *h* given input *x*

1. Feature Scaling
2. Learning Rate
3. Error and # of iterations
4. **Regularization (L2)**

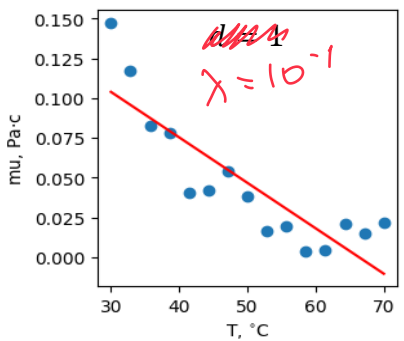
Model *parameters* are determined during the solution of the ML problem. For example, in regression problems, the parameters are the components of the matrix of weights $\boldsymbol{\phi}$. *Hyperparameters* are set by the user, usually not in a single way, and their values affect the values of the sought parameters.

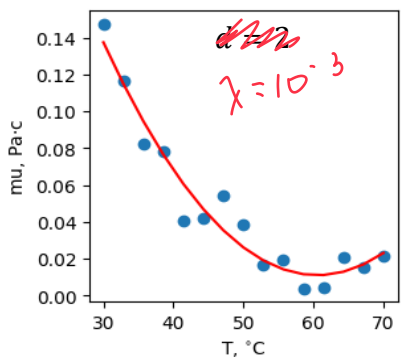


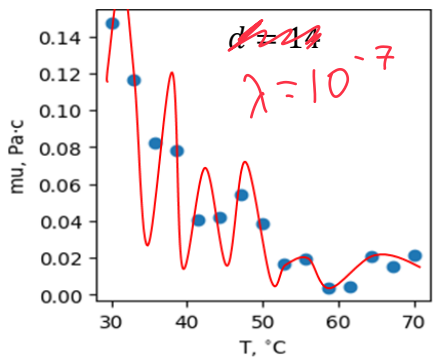
$$L = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h^{(i)} - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\phi_j^2\right] \Rightarrow \min.$$

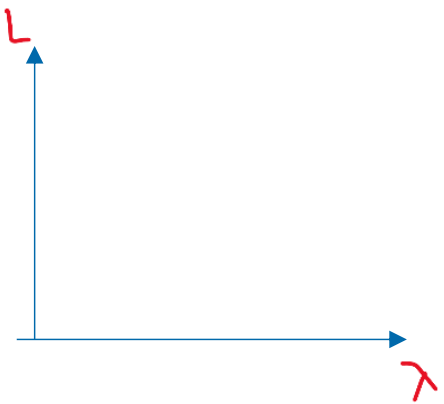
$\varphi_0\ \varphi_1\ \varphi_2\ \ldots$

$h(x) = \theta_j x^j \quad , (j = 0,..d)$









Agenda

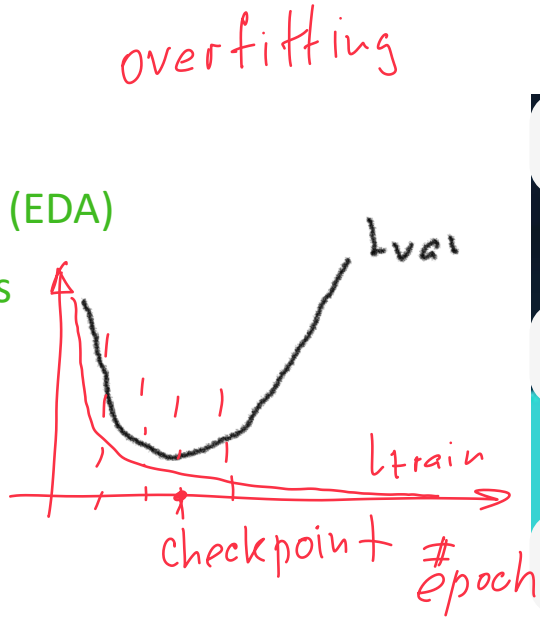
I. Exploratory data analysis (EDA)

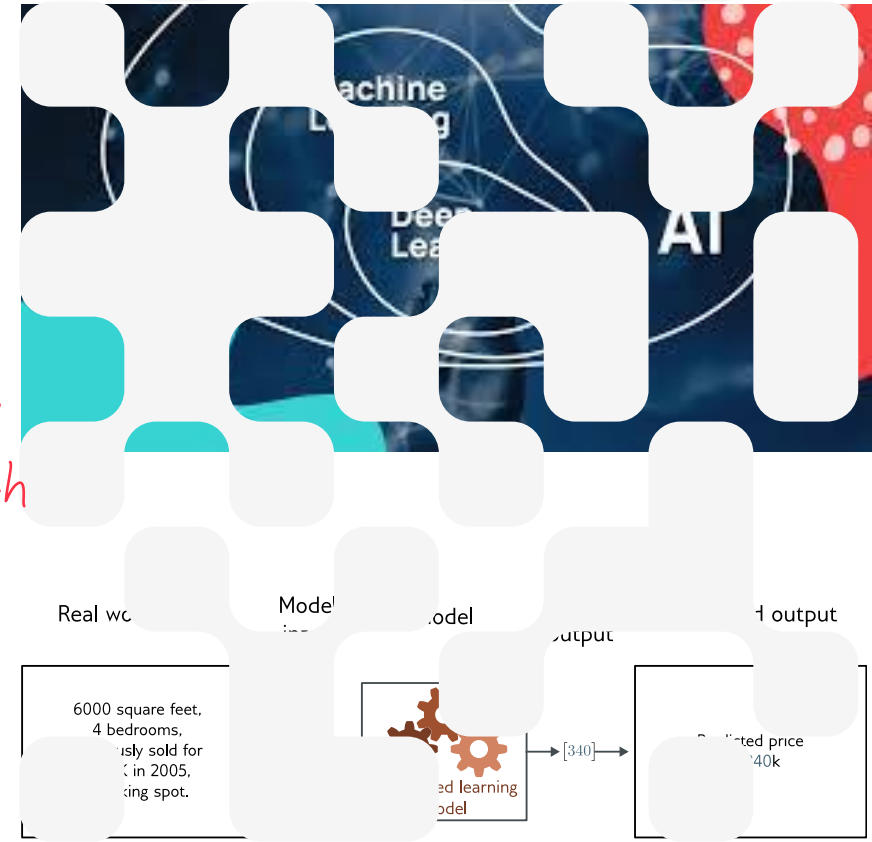
II. Logistic regression and its generalization

III. Metrics

IV. Setting of the models

V. Loss functions

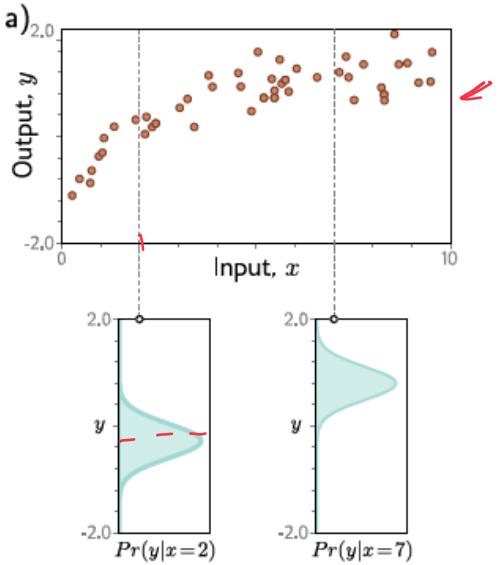




All models are wrong, but some are useful.
/George Box/

Recap: Maximum likelihood

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$. We now shift perspective and consider the model as computing a conditional probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$ or $p(Y|X)$. The loss encourages each training output to have a high probability under the distribution $p(y^{(i)}|x^{(i)})$. Suppose, the data are independent and identically distributed (i.i.d.).



$$i: \quad p(y^{(i)} | f[x^{(i)}, \varphi]) \qquad p(y^{(1)}, y^{(2)}, y^{(3)} \ldots y^{(m)} | f[x^{1}, \varphi], f[x^{2}, \varphi] \ldots)$$

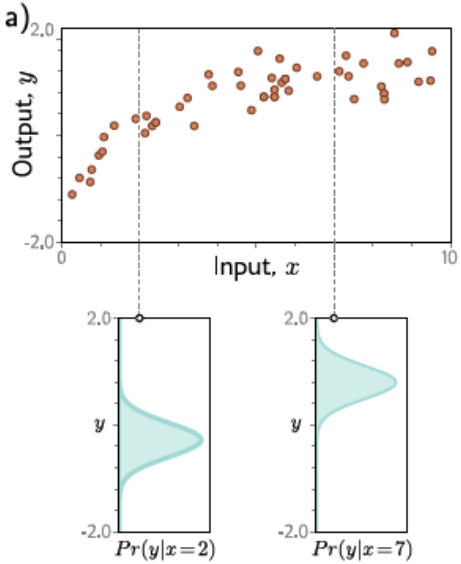
$$J = \arg\max_{\varphi} \left\{ \prod_{i} p(y^{(i)} | f[x^{(i)}, \varphi]) \right\} \Rightarrow \max$$

Maximum likelihood

$$L = -\log(J) = \arg\min_{\varphi} \left\{ \sum_{i} \log\{p(y^{i} | f[x^{i}, \varphi])\} \right\} \Rightarrow \min$$

S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

Recap: Algorithm for constructing loss funtions



The recipe for constructing loss functions for training data $\{\mathbf{x}_i, \mathbf{y}_i\}$ using the maximum likelihood approach is hence:

1. Choose a suitable probability distribution $Pr(\mathbf{y}|\boldsymbol{\theta})$ defined over the domain of the predictions $\mathbf{y}$ with distribution parameters $\boldsymbol{\theta}$.
2. Set the machine learning model $\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ to predict one or more of these parameters, so $\boldsymbol{\theta} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ and $Pr(\mathbf{y}|\boldsymbol{\theta}) = Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \boldsymbol{\phi}])$.
3. To train the model, find the network parameters $\hat{\boldsymbol{\phi}}$ that minimize the negative log-likelihood loss function over the training dataset pairs $\{\mathbf{x}_i, \mathbf{y}_i\}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big[L[\boldsymbol{\phi}]\Big] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\left[-\sum_{i=1}^{I} \log\Big[Pr(\mathbf{y}_i|\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])\Big]\right]. \tag{5.6}$$

4. To perform inference for a new test example $\mathbf{x}$, return either the full distribution $Pr(\mathbf{y}|\mathbf{f}[\mathbf{x}, \hat{\boldsymbol{\phi}}])$ or the value where this distribution is maximized.

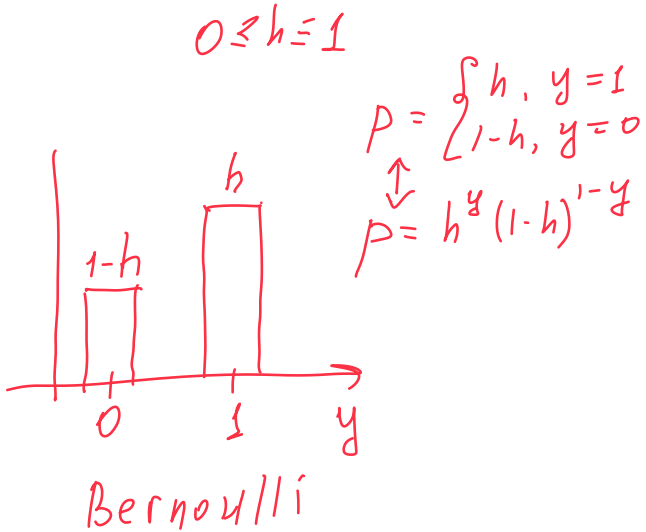
We devote most of the rest of this chapter to constructing loss functions for common prediction types using this recipe.

negative log-likelihood

S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

Maximum likelihood for Binary Cross-Entropy (BCE) loss

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$. We now shift perspective and consider the model as computing a conditional probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$ or $p(Y|X)$. The loss encourages each training output to have a high probability under the distribution $p(y^{(i)}|x^{(i)})$. Suppose, the data are independent and identically distributed (i.i.d.).



$$h = -\sum_i \log(p_i) \Rightarrow \min$$

$$-\sum_i \log\left(h_i^{y^i}(1-h_i)^{1-y^i}\right) \Rightarrow \min$$

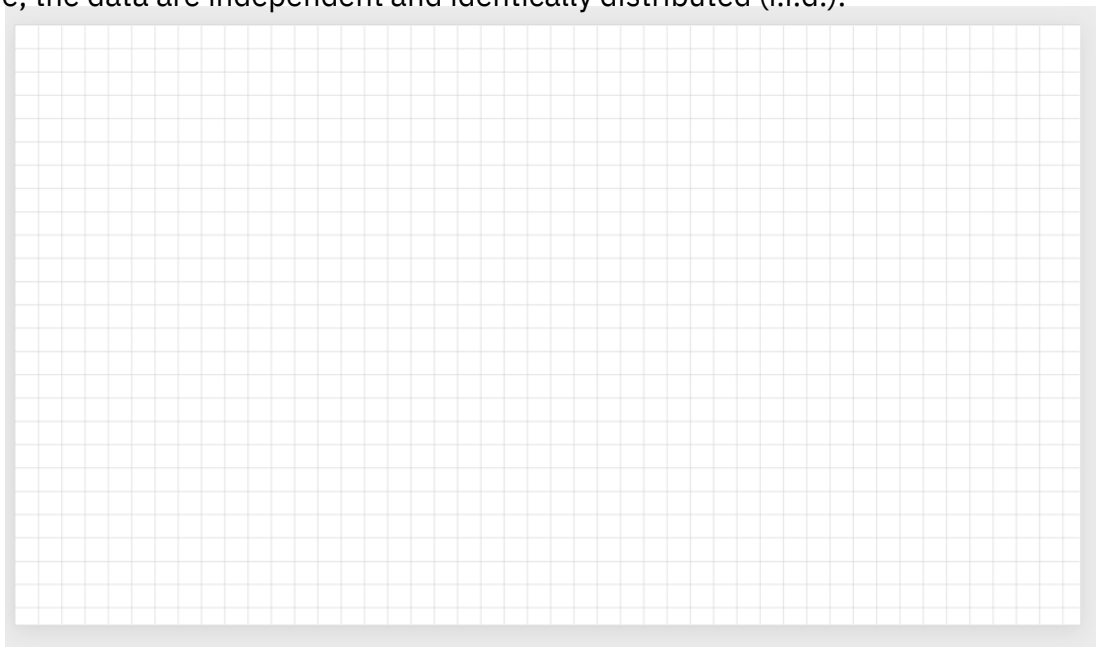
$$h = -\sum_i \left[y^i \log(h_i) + (1-y^i)\log(1-h_i)\right]$$

BCE loss

S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

Maximum likelihood for Binary Cross-Entropy (BCE) loss

Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$. We now shift perspective and consider the model as computing a conditional probability distribution $p(\boldsymbol{y}|\boldsymbol{x})$ or $p(Y|X)$. The loss encourages each training output to have a high probability under the distribution $p\left(y^{(i)}|x^{(i)}\right)$. Suppose, the data are independent and identically distributed (i.i.d.).



S. J. Prince. Understanding Deep Learning. MIT Press, 2023.

Thank you for your attention!

a.kornaev@innopolis.ru, @avkornaev

