Machine Learning

2025 (ML-2025)
Lecture 2. Linear models

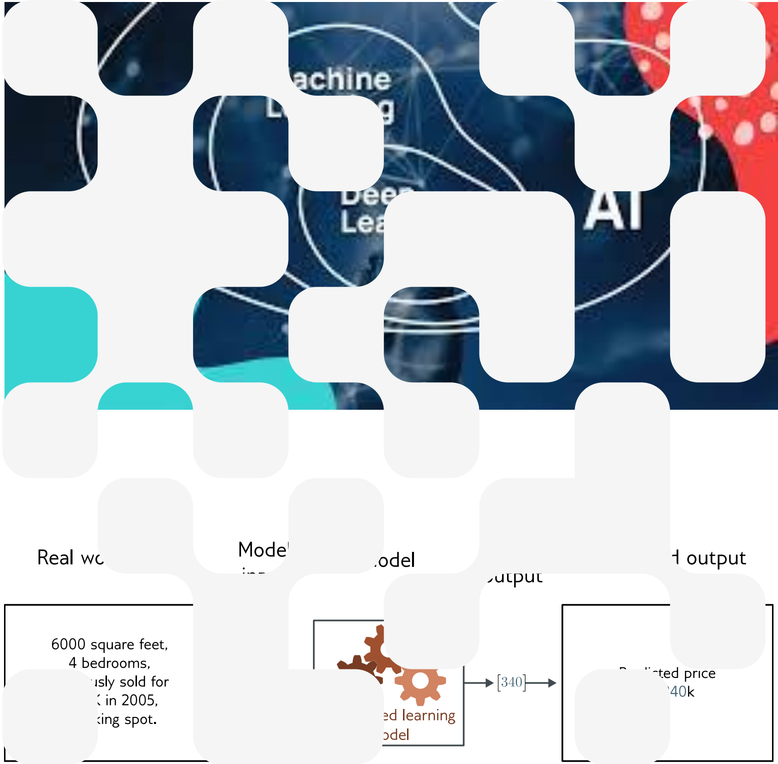
by Alexei Kornaev, Dr. Sc., Assoc. Prof., Robotics and CV, Innopolis University
Researcher at the RC for AI, National RC for Oncology n.a. NN Blokhin

Agenda

I. **Linear Regression and its Generalization**

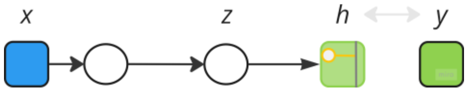
II. Logistic Regressioin and its Generalization

III. Setting of the models



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

Linear Regression



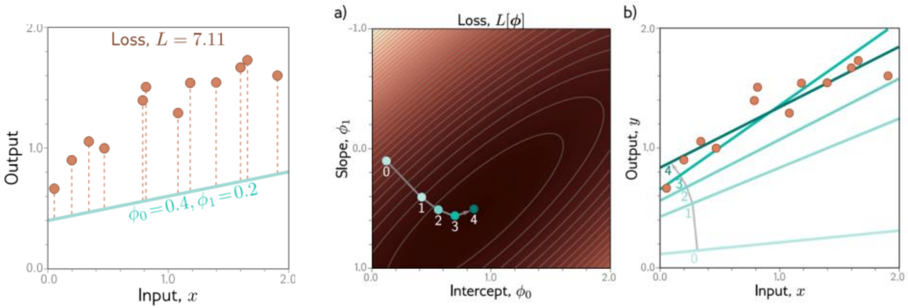
Model predicts output *h* given input *x*

$h = \phi_0 + \phi_1 x$

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 $y^{(i)}$.

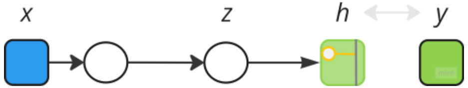
$$\boldsymbol{x} = \begin{bmatrix} x^{(1)} \\ \dots \\ x^{(m)} \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}, \boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}.$$

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



Supervised learning intuition: S. J. Prince. Understanding Deep Learning. MIT Press, 2023. URL http://udlbook.com.

Linear Regression



Model predicts output *h* given input *x*

Consider a model $\boldsymbol{f} = \left[x^{(i)}, \boldsymbol{\phi}\right]$ 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 $y^{(i)}$.

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

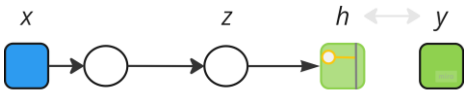
$$\boxed{L(\boldsymbol{\phi}) = \frac{1}{2m} \sum_{i=1}^{m} \left(h^{(i)} - y^{(i)}\right)^2 \Rightarrow \min.}$$

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

Training algorithm.

1. Initialize the *weights* $\boldsymbol{\phi}$ with a random seed
2. Calculate the *hypothesis* matrix $\boldsymbol{h} = \boldsymbol{x\phi}$ 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): $\boldsymbol{\phi}$.

Linear Regression. Generalization (multiple var., polynomial)



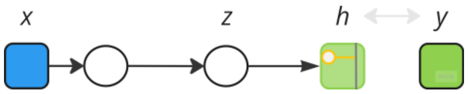
Model predicts output *h* given input *x*

Consider a model $\boldsymbol{f} = \left[x^{(i)}, \boldsymbol{\phi}\right]$ 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 $y^{(i)}$.

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

Training algorithm.
1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{h} = \boldsymbol{x}\boldsymbol{\phi}$ and $\nabla L = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y})$
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): $\boldsymbol{\phi}$.

Linear 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 $y^{(i)}$.

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

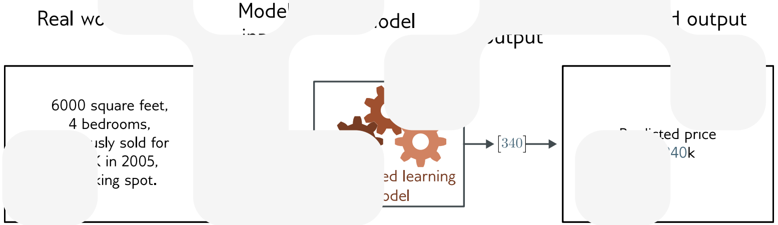
Training algorithm.
1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{h} = \boldsymbol{x}\boldsymbol{\phi}$ and $\nabla L = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y})$
3. Update $\boldsymbol{\phi}^{next} = \boldsymbol{\phi}^{prev} - \alpha\nabla L$
4. Repeat pp. 2-3 $L^{next}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

Agenda

I. Linear Regression and its Generalization

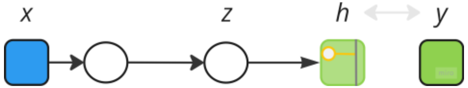
II. Logistic Regressioin and its Generalization

III. Setting of the models



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

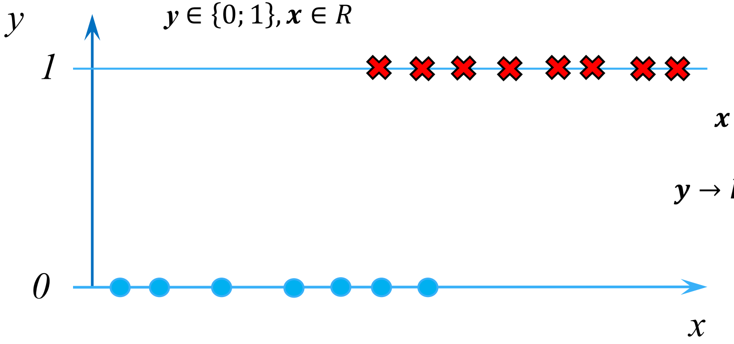
Logistic Regression



Model predicts output *h* given input *x*

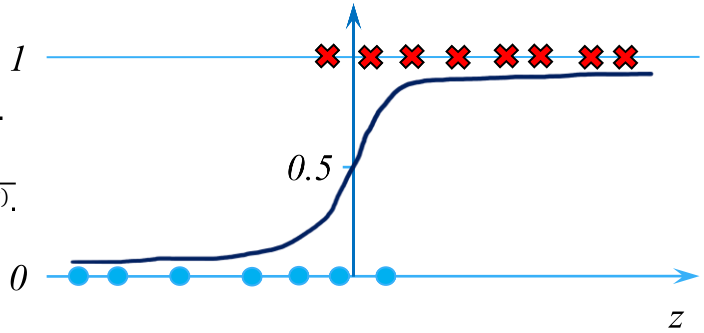
Consider a model $\boldsymbol{f} = [x^{(i)}, \boldsymbol{\phi}]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

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



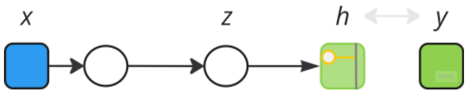
$$\boldsymbol{x} \to \boldsymbol{z}:\ z = \phi_0 + \phi_1 x.$$

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



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

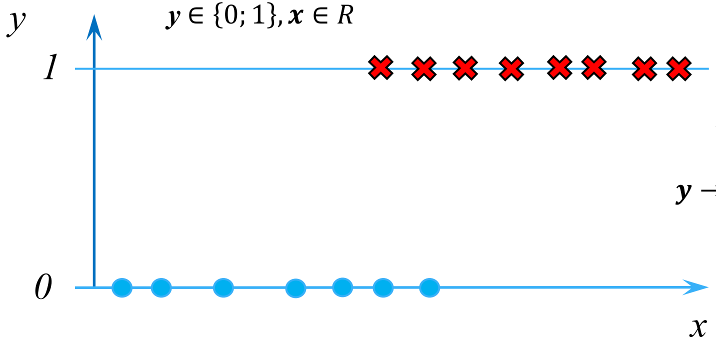
Logistic Regression



Model predicts output *h* given input *x*

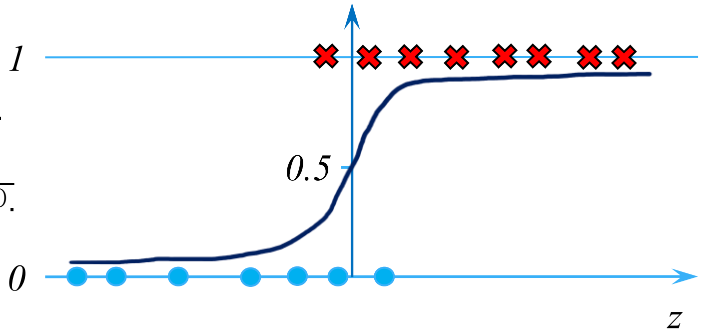
Consider a model $\boldsymbol{f} = \left[x^{(i)}, \boldsymbol{\phi}\right]$ parameterized with weights $\boldsymbol{\phi}$ that maps each i-th input sample $x^{(i)}$ into the output $z^{(i)}$ which then transforms into the hypothesis $h^{(i)}$ that should be close to the label $y^{(i)}$.

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



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

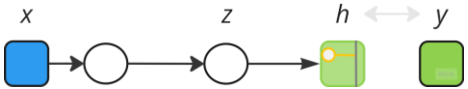
$\boldsymbol{y} \rightarrow \boldsymbol{h}$: $h(z) = \dfrac{1}{1 + e^{-z(x)}}.$



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)

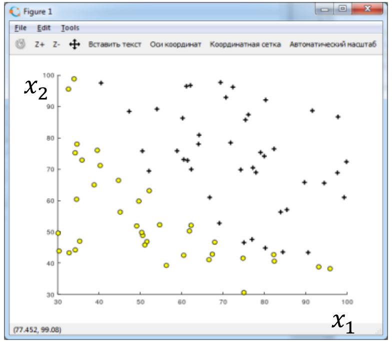


Model predicts output *h* given input *x*

Consider a model $\boldsymbol{f} = \left[x^{(i)}, \boldsymbol{\phi}\right]$ 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 $y^{(i)}$.

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

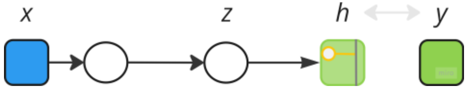
$$\boldsymbol{x} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ \dots & \dots \\ x_1^{(m)} & x_2^{(m)} \end{bmatrix}; \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix}; \quad \rightarrow \quad \boldsymbol{x} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ \dots & \dots & \dots \\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



Training algorithm.

1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{z} = \boldsymbol{x\phi}$, $\boldsymbol{h} = \sigma(\boldsymbol{z})$, 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 $\boldsymbol{f} = \left[x^{(i)}, \boldsymbol{\phi}\right]$ 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 $y^{(i)}$.

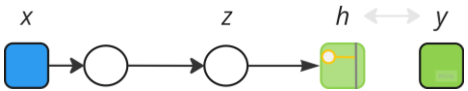
$$L(\boldsymbol{\phi}) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\ln(h^{(i)}) + (1-y^{(i)})(\ln(1-h^{(i)})\right) \Rightarrow \min.$$

[Gray scale picture of "Nine"]

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

Training algorithm.
1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{z} = \boldsymbol{x\phi}$, $\boldsymbol{h} = \sigma(\boldsymbol{z})$, 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)

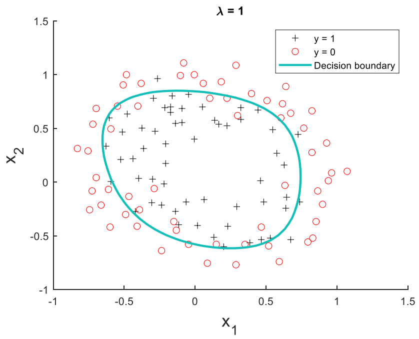


Model predicts output *h* given input *x*

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 $y^{(i)}$.

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

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



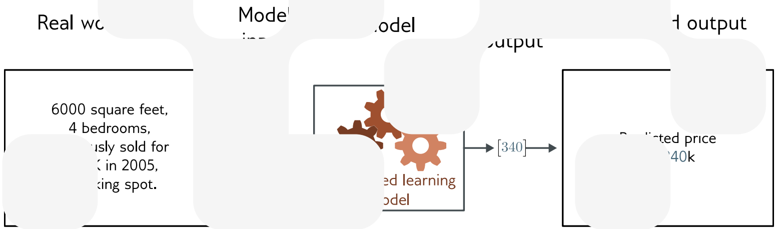
Training algorithm.
1. Initialize $\boldsymbol{\phi}$
2. Calculate $\boldsymbol{z} = \boldsymbol{x}\boldsymbol{\phi}$, $\boldsymbol{h} = \sigma(\boldsymbol{z})$, then $\nabla L = \frac{1}{m}\boldsymbol{x}^T(\boldsymbol{h} - \boldsymbol{y})$
3. Update $\boldsymbol{\phi}^{next} = \boldsymbol{\phi}^{prev}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

Agenda

I. Linear Regression and its Generalization

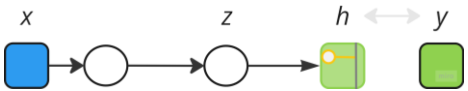
II. Logistic Regressioin and its Generalization

III. Setting of the models



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

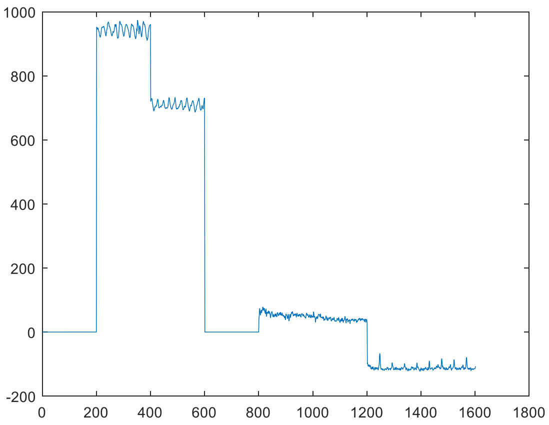
ML Settings

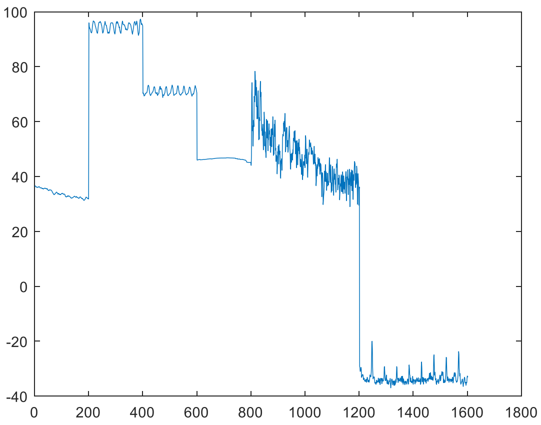


Model predicts output *h* given input *x*

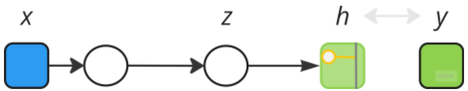
Model *parameters* are determined during the solution of the ML problem. For example, in regression problems, the parameters are the components of the matrix of weights $\boldsymbol{\phi}$. *Hyperparameters* are set by the user, usually not in a single way, and their values affect the values of the sought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

1. Feature Scaling





ML Settings

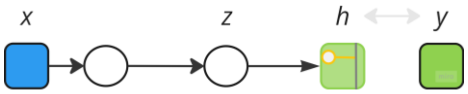


Model predicts output *h* given input *x*

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.

ML Settings



Model predicts output *h* given input *x*

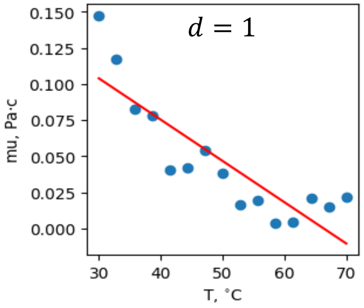
1. Feature Scaling
2. Learning Rate
3. Error and # of iterations
4. **Regularization (L2)**

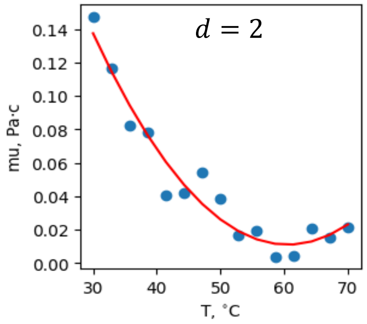
$h(x) = \theta_j x^j \quad ,(j = 0,..d)$

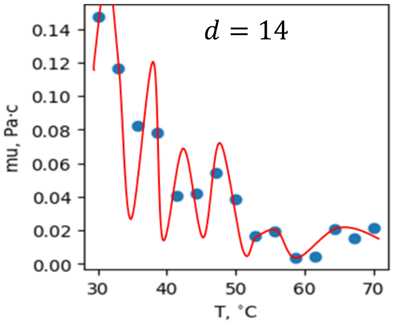
Model *parameters* are determined during the solution of the ML problem. For example, in regression problems, the parameters are the components of the matrix of weights $\boldsymbol{\phi}$. *Hyperparameters* are set by the user, usually not in a single way, and their values affect the values of the sought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

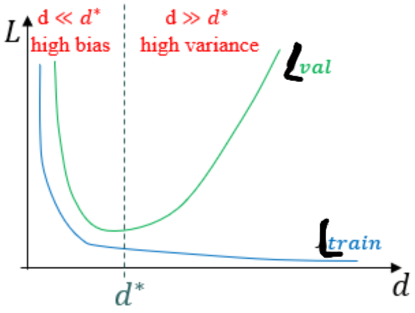
Training $\{(x_i, y_i)\}$ | validation | test

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









Just think about it

1. How can the gradient descent method be improved to find global minima instead of local ones?
2. Can the discussed linear regression problems be solved analytically without using the gradient descent method?
3. Why is the use of high-degree polynomials generally not recommended when building regression models?

Thank you for your attention!

a.kornaev@innopolis.ru, @avkornaev