IEEE Signal Processing Society Seasonal School
28.03.2022 – 01.04.2022

Networked Federated Learning

Theory, Algorithms and Applications

Alex Jung (Aalto University)

Aalto University

IEEE Signal Processing Society

Organizing Committee:

Dr. Alex Jung	Dr. Simo Särkkä	Dr. T.S.N. Murthy
Ass. Prof. at Aalto U.	Assoc. Prof. at Aalto U.	Ass. Prof. JNTUK Vizianagaram
IEEE Finland Chapter SP/CAS	IEEE Finland Chapter CSS/RAS/SMCS	IEEE Vizag Bay Chapter COMSOC/SPS

also offered as Aalto course (2 credits):

CS-E407508 - Special course in Machine learning and Data science: Networked Federated Learning, Lectures, 28.3.2022-3.6.2022

CS-E407508 –
Special course in Machine learning and Data science:
Networked Federated Learning

- grading based on Exercises.
- exam: discussion of exercises (in person or zoom).

About Me.

- MSc (2008) and Ph.D. (2012) in EE, TU Vienna
- since 2015 Ass. Prof. for Machine Learning at Aalto/CS
- leading group “Machine Learning for Big Data”
- 2020- Associate Editor for IEEE Signal Processing Letters
- 2019-2022: Chair of IEEE Finland Jt. Chapter SP/CAS

RA1: Networked Federated Learning.

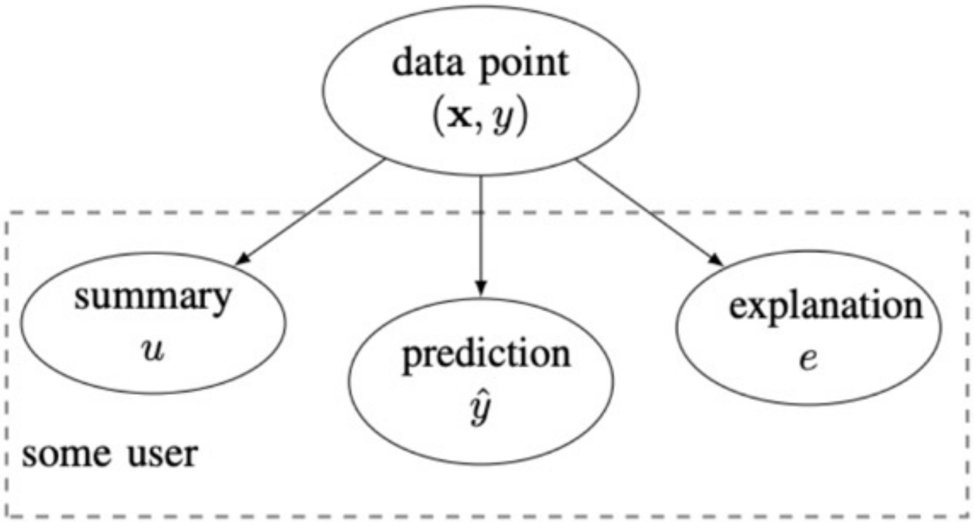
High-Precision Management of Pandemics

Y. Sarcheshmehpour, M Leinonen and AJ, “Federated Learning From Big Data Over Networks”, IEEE ICASSP, 2021.

AJ, “Networked Exponential Families for Big Data Over Networks,” in IEEE Access, 2020, doi: 10.1109/ACCESS.2020.3033817.

AJ, N. Tran, “Localized Linear Regression in Networked Data,” in IEEE SPL, 2019, doi: 10.1109/LSP.2019.2918933.

RA2: Explainable Machine Learning.



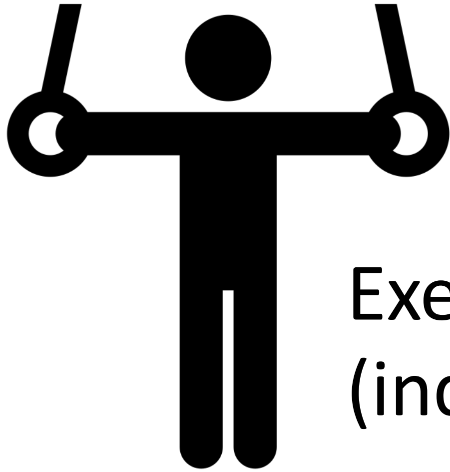
explanation can be:
- relevant example of training set
- subset of features
- counterfactuals
- a free text explanation
- court sentence

L. Zhang, G. Karakasidis, A. Odnoblyudova, L. Dogruel and AJ., "Explainable Empirical Risk Minimization", arXiv, 2022. weblink

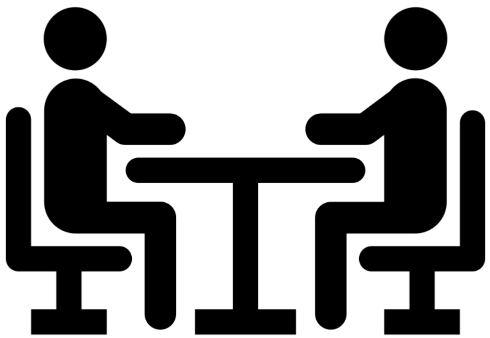
AJ and P. H. J. Nardelli, "An Information-Theoretic Approach to Personalized Explainable Machine Learning," in IEEE SPL, 2020, doi: 10.1109/LSP.2020.2993176.

School Format.

Lectures
(via zoom, recorded)



Exercises
(independent work)



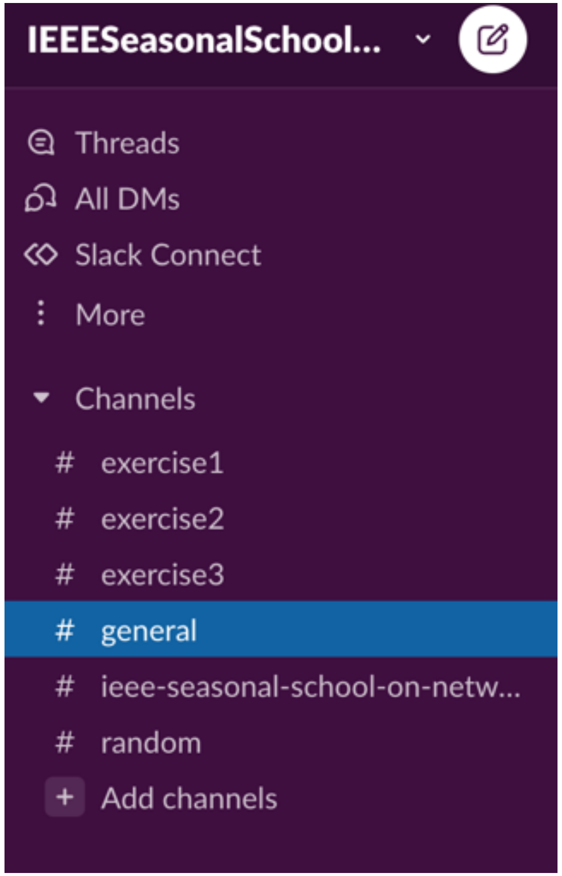
Discussion Forum
[click here to join]

all times EET/local Helsinki time)

hall: https://aalto.zoom.us/j/64034033878?pwd=ayt4aDNyaEVSSGt1eHNNZFVFR2RoZz09

- Tue, 11:00-13:00, Basics of spectral graph theory, Prof. Avratchenkov
- We, 10:00-11:00, Parallel/distributed methods for state-space models, Prof. Särkkä
- We, 12:30-13:30, Compact and Efficient Neural Networks, Steps Towards Communication Efficient Federated Learning, Dr. H. Tavakoli
- Thu., 11:00-12:00, Machine Learning applications in meteorological forecasting, Dr. Schicker
- Thu, 13:00-14:00, Towards Communication-Efficient and Personalized Federated Learning, Dr. Samek
- Thu, 14:30-15:30, Communication-Computation Efficient Distributed Machine Learning , Prof. Fischione
- Fr, 11:00-12:00, Tackling the problem of “bad” explanations with the Human-in-the-Loop principle, Dipl.-Ing. Saranti

Discussion Forum („Slack“)



can be used via web-browser or with an installed app;

link to join is on school site https://ieeespcasfinland.github.io/

post your questions there in suitable channel, e.g., “exercise1”
don’t hesitate to answer other’s questions

Support for Exercises

Dr. Yu Tian

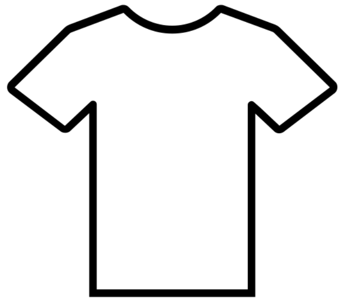
MSc. Dick Carrillo Melgarejo

Dr. Shamsiiat (Shamsi) Abdurakhmanova

Exercises.

- three exercises: Exercise 1, 2 and 3
- each exercise consists of a Python notebook
- notebook contains starter code and explanations
- tasks at the end of notebooks

Running Toy Application



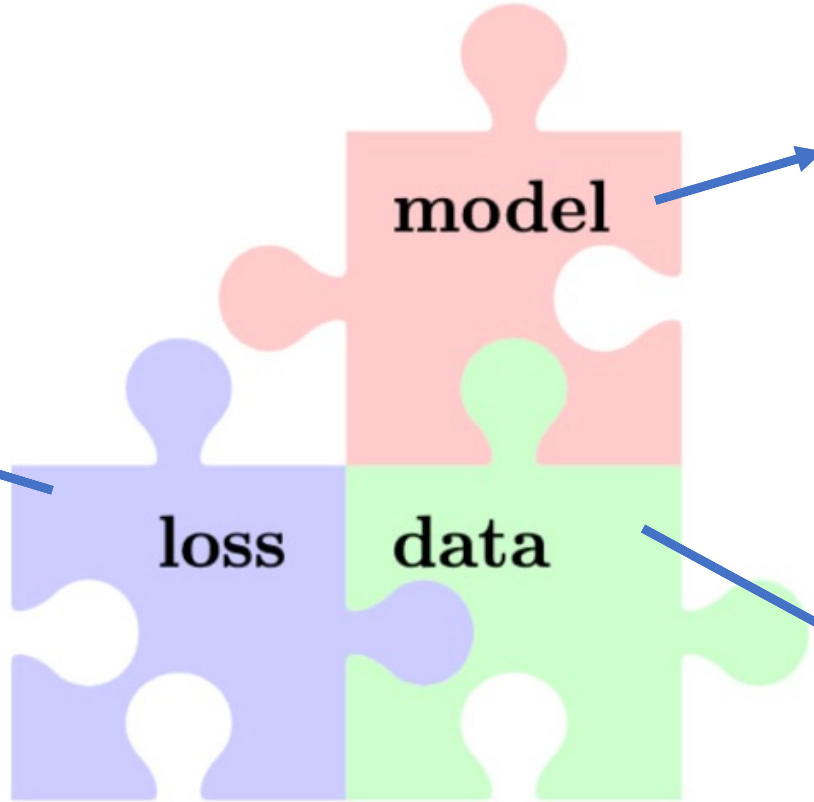
morning temperature = - 10 (minimum daytime temp.)
maximum daytime temperature ?

Exercise 1.

- basic components of ML: data, model and loss
- principle of empirical risk minimization (ERM)
- solve ERM using gradient descent
- read in datapoints from a csv file

Three Components of ML

- squared error
- logistic loss
- focal loss
- ....



- linear space
- decision tree
- neural nets
- ....

- numeric arrays
- text
- networks
- ...

Data.

```
import pandas as pd  # the pandas package provides tools for
df = pd.read_csv('FMIData.csv') # read in data from a csv fil
df.head(5)      # show first 5 datapoints
```

$h(x)$ $\hat{y}$

	date	min_temp	max_temp	latitude	longitude	station
0	2021-04-11	$x^{(0)}$ 2.4	$y^{(0)}$ 6.1	60.30373	25.54916	Porvoo Kilpilahti satama
1	2021-04-12	4.9	9.9	60.30373	25.54916	Porvoo Kilpilahti satama
2	2021-04-13	2.8	6.5	60.30373	25.54916	Porvoo Kilpilahti satama
3	2021-04-14	0.5	6.5	60.30373	25.54916	Porvoo Kilpilahti satama
4	2021-04-15	0.7	9.4	60.30373	25.54916	Porvoo Kilpilahti satama

Feature Matrix and Label Vector.

	date	min_temp	max_temp
0	2021-04-11	2.4	6.1
1	2021-04-12	4.9	9.9
2	2021-04-13	2.8	6.5
3	2021-04-14	0.5	6.5
4	2021-04-15	0.7	9.4

$\Rightarrow y \in \mathbb{R}^5$

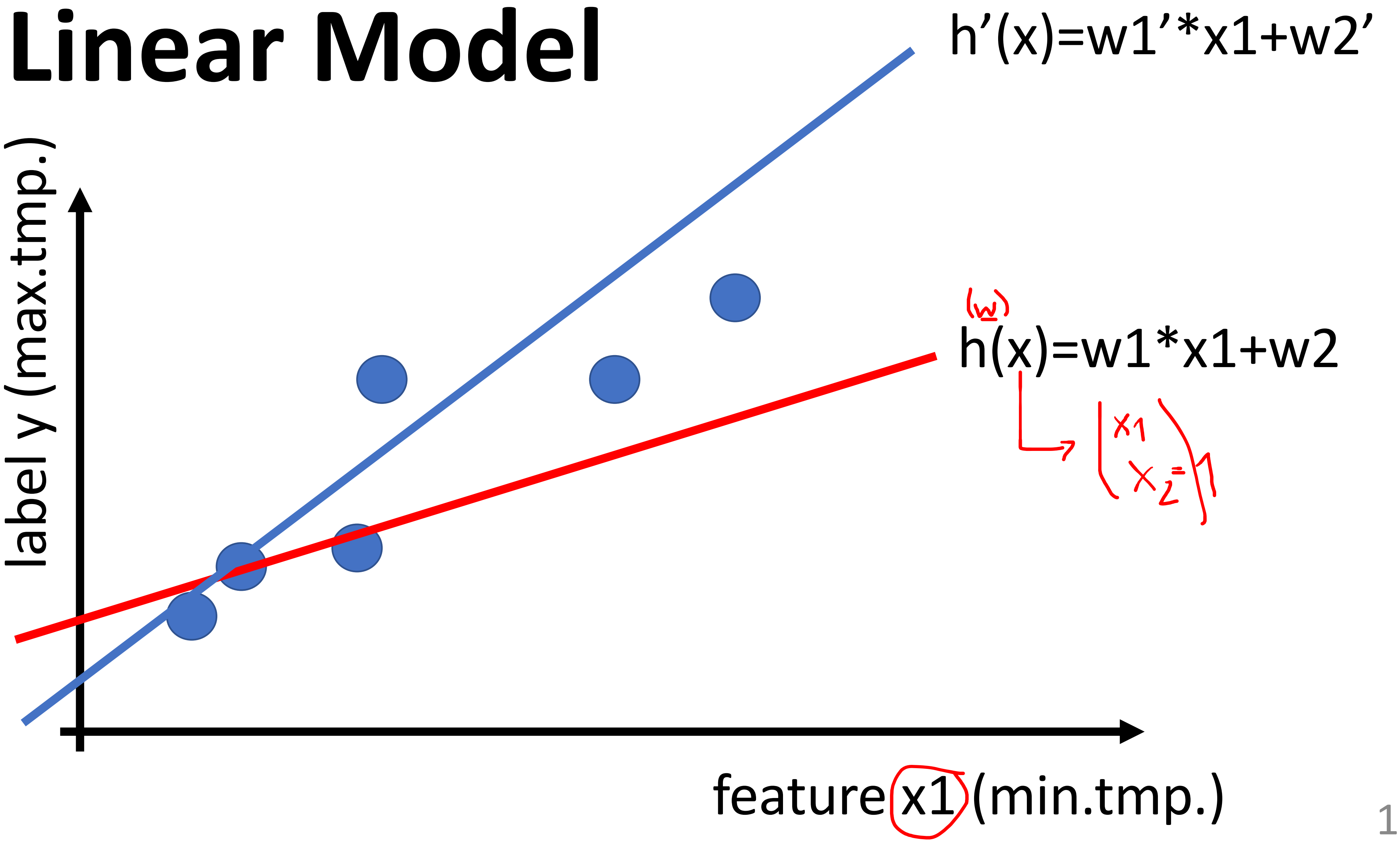
$$\underline{X} = \begin{pmatrix} 2.4 & 1 \\ 4.9 & 1 \\ \vdots & \vdots \\ 0.7 & 1 \end{pmatrix}$$

$x^{(1)} = (x_1^{(1)}\ x_2^{(1)})$

$x^{(2)}$

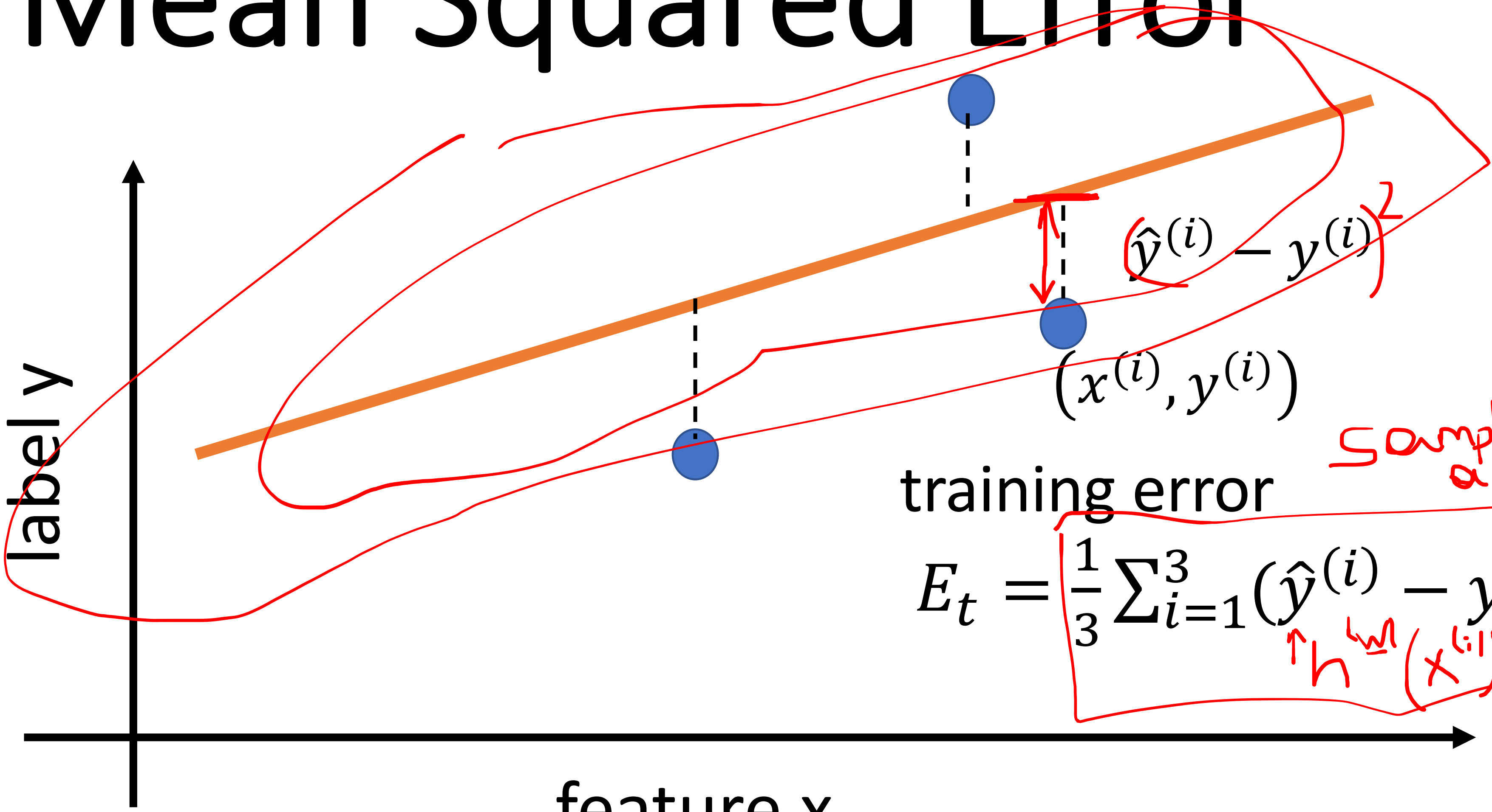
```
X = np.hstack((df["min_temp"].to_numpy().reshape(m,1),np.ones((m,1))))
y = df["max_temp"].to_numpy().reshape(m,)
```

Linear Model



which hypothesis $h(x)$, or model parameter w, is best?

Mean Squared Error



training error

sample aver.

$$E_t = \boxed{\frac{1}{3}\sum_{i=1}^{3}(\underbrace{\hat{y}^{(i)}}_{h^{(w)}(x^{(i)})} - y^{(i)})^2}$$

Empirical Risk Minimization (informal)

learn hypothesis (model parameters) such that the resulting average loss ("empirical risk") on a training set is minimal

Empirical Risk Minimization

$$\underline{X} = \begin{pmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \end{pmatrix}$$

datapoints $\left(\mathbf{x}^{(1)}, y^{(1)}\right), \ldots, \left(\mathbf{x}^{(m)}, y^{(m)}\right)$;

$$\min_{\mathbf{w}\in\mathbb{R}^n} (1/m) \sum_{i=1}^{m} \Big(\underbrace{\mathbf{w}^T \mathbf{x}^{(i)}}_{h(\mathbf{x}^{(i)})} - y^{(i)} \Big)^2 .$$

$h^{(w)}$

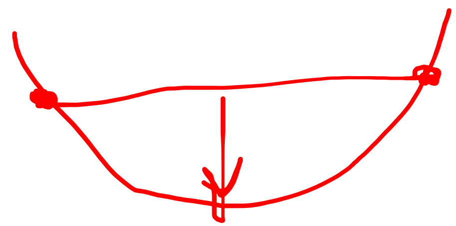
Matrix/Vector Notation

$$\min_{\mathbf{w}} f(\mathbf{w}) \text{ , with } f(\mathbf{w}) := (1/m)\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

feature matrix $\mathbf{X} = \left(\left(\mathbf{x}^{(1)}\right)^T, \ldots, \left(\mathbf{x}^{(m)}\right)^T\right)^T$

label vector $\mathbf{y} = \left(y^{(1)}, \ldots, y^{(m)}\right)^T.$

Gradient Descent



$$\min_{\mathbf{w}} f(\mathbf{w}) \text{ , with } f(\mathbf{w}) := (1/m)\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

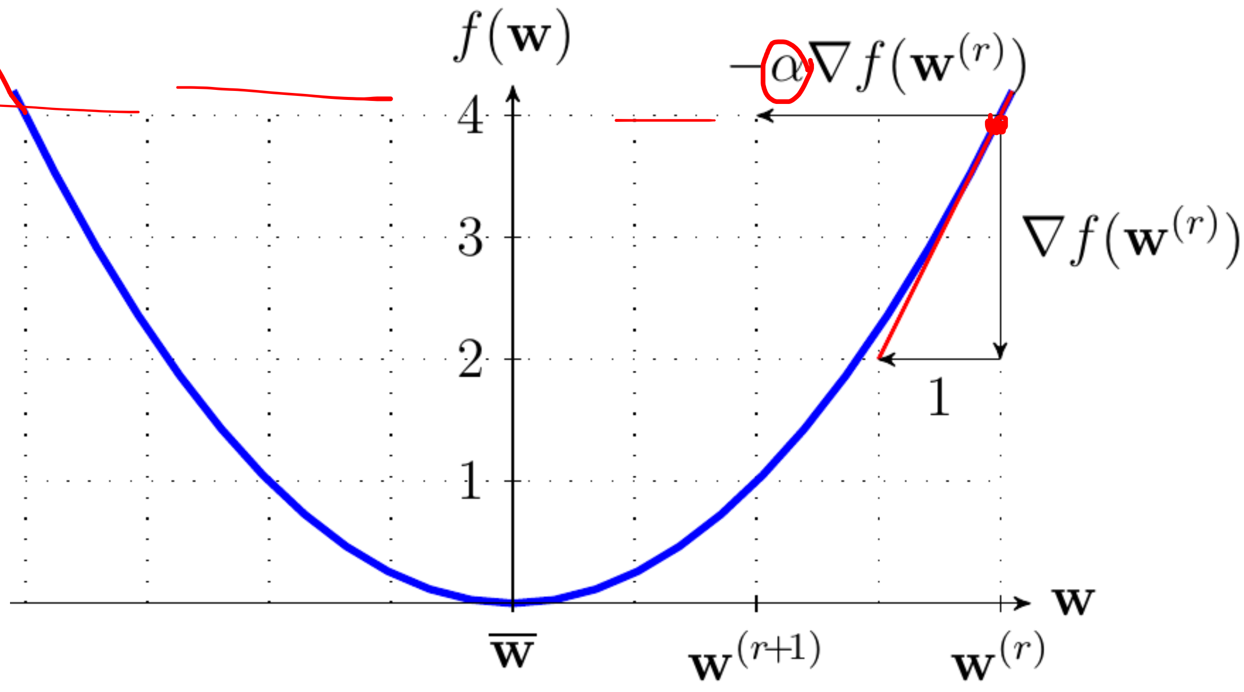
start with some initial guess $\boldsymbol{w}^{(0)}$ and iteratively improve by GD steps

$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \alpha \nabla f\left(\mathbf{w}^{(r)}\right)$$

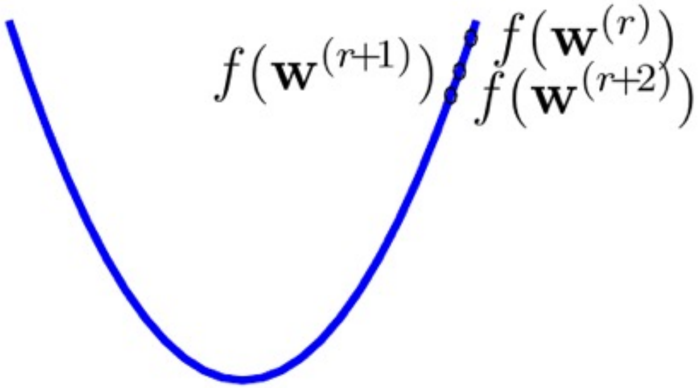
with step-size or "learning rate" α

Gradient Descent Step

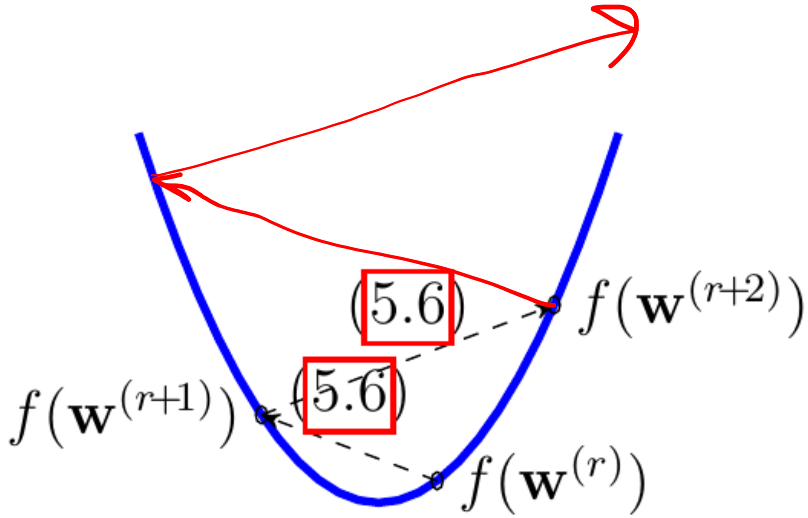
$$\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \alpha \nabla f\left(\mathbf{w}^{(r)}\right)$$



Effect of Learning-Rate.

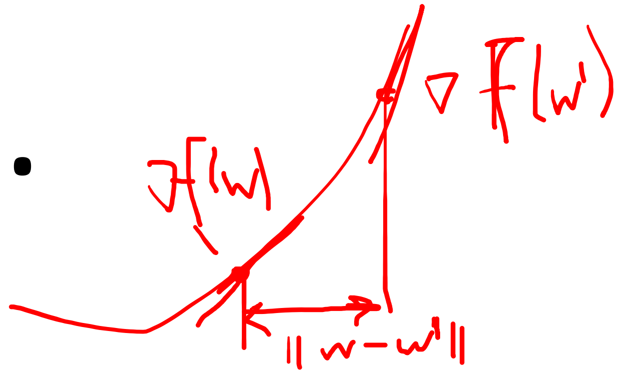


α too small: GD steps make only very little progress!



α too large: GD steps depart from optimum!

A Sufficient Condition.

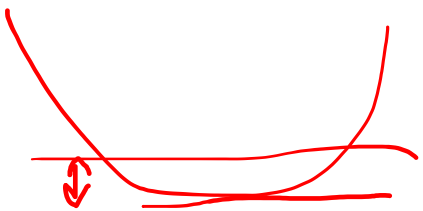


assume objective function is "β-smooth"

$$\|\nabla f(\mathbf{w}) - \nabla f(\mathbf{w}')\| \le \beta \|\mathbf{w} - \mathbf{w}'\|.$$

GD converges when using learning rate

$$\alpha = 1/\beta$$



proof: https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/lec6.pdf

Computing Gradient.

to implement GD step $\mathbf{w}^{(r+1)} = \mathbf{w}^{(r)} - \alpha \nabla f\left(\mathbf{w}^{(r)}\right)$

we need to compute gradient $\nabla f\left(\mathbf{w}^{(r)}\right)$

for $f(\mathbf{w}) := (1/m)\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$ we obtain

$$\nabla f(\mathbf{w}) = -(2/m)\mathbf{X}^T\left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)$$

see A.4 of S. Boyd and L. Vandenberghe, "Convex Optimization", CUP, 2004

GD for Linear Regression.

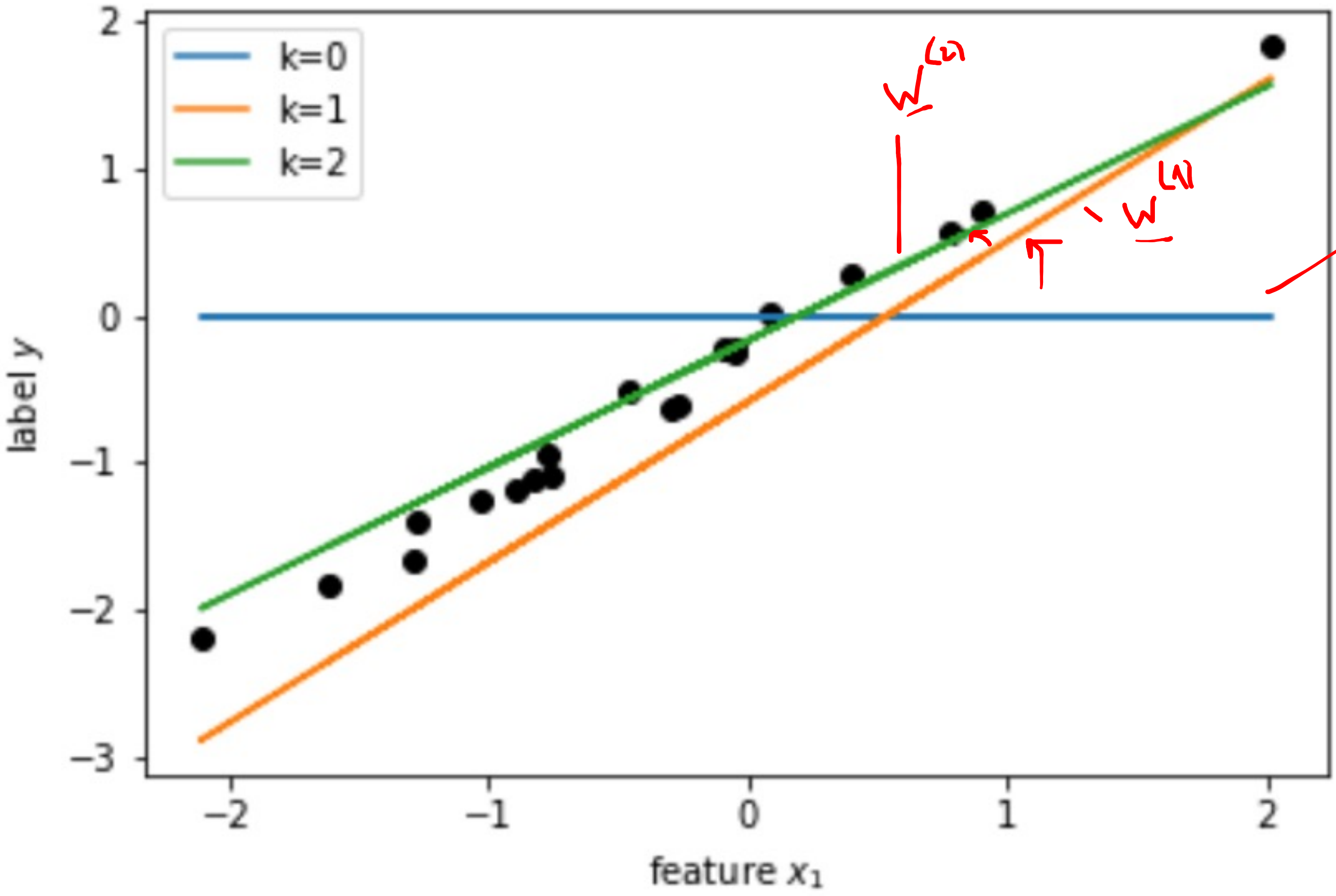
$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \alpha(2/m)\mathbf{X}^T\left(\mathbf{y} - \mathbf{X}\mathbf{w}^{(k)}\right)$$

```
# compute the gradient of f(w) at the current weight vector (ob
gradient = -(2/m) * X.T.dot(y - X.dot(current_weights))
# update the current weight vector via the GD step
current_weights = current_weights - (learning_rate * gradient)
```

When to stop ?

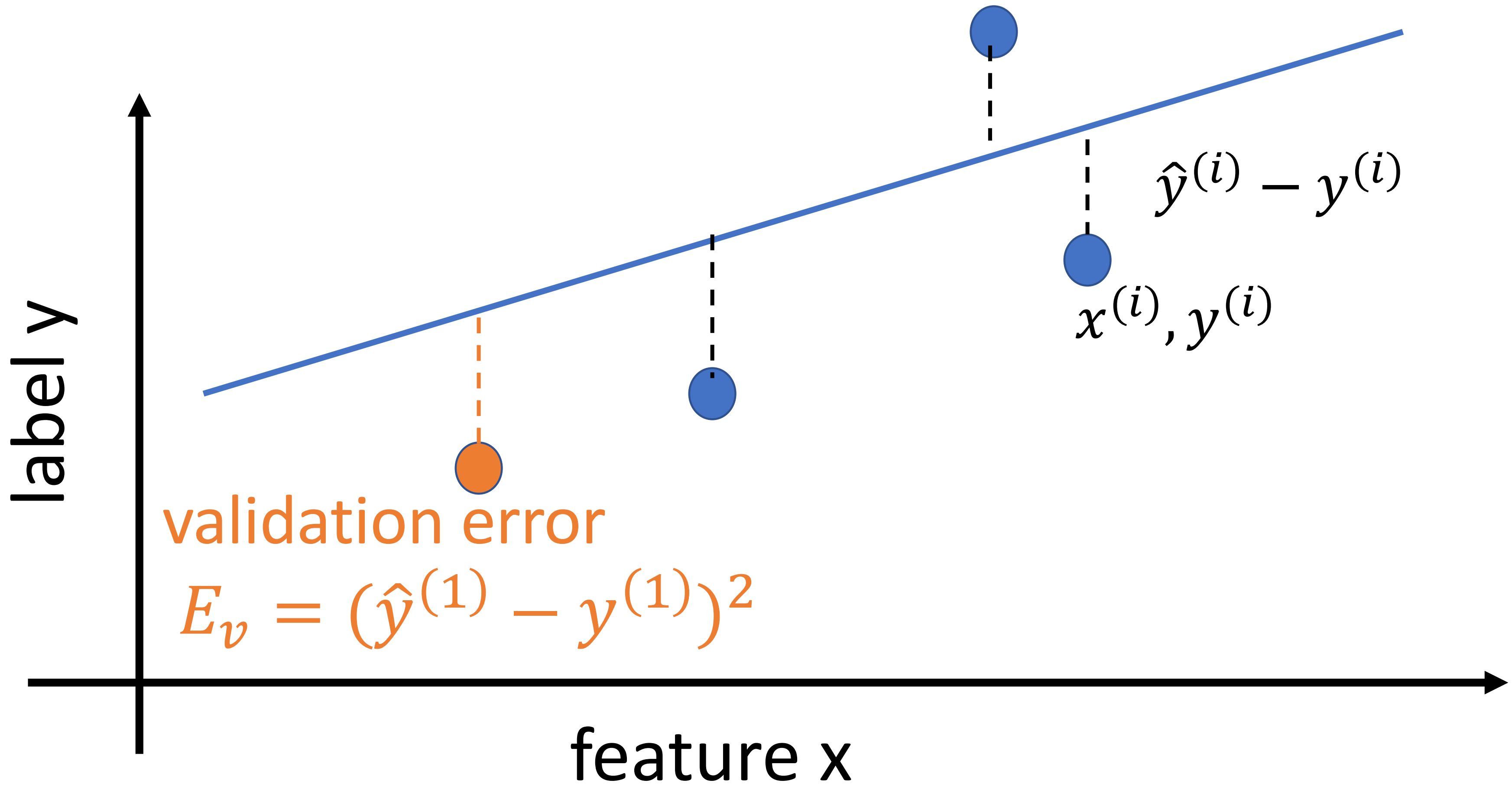
- nr. of steps required to ensure sub-optimality (”theory”)
- check for sufficiently large decrease $f\left(\boldsymbol{w}^{(k+1)}\right)$ - $f\left(\boldsymbol{w}^{(k)}\right)$
- try $h(x) = \boldsymbol{w}^{(k)}x$ on validation set (“early stopping”)

GD in Action.



$h^{(0)}(x) = (w^{(0)})^T x$

Train and Validate.



Exercise 1 – Task 1.1

Generate synthetic dataset of m data points. Each datapoint characterized by feature vector $\boldsymbol{x} = (x_1, x_2)^T$ with "true" $x_1 \sim \mathcal{N}(0,1)$ and $x_2 = 1$. and numeric label $y = \bar{w}_1 x_1 + \bar{w}_2 x_2 + \sigma\varepsilon$. with some given "true weights" $\bar{w}_1, \bar{w}_2$ a given noise strength σ and Gaussian noise $\varepsilon \sim \mathcal{N}(0,1)$

Study the deviation of the learnt weight vector after a given number N of GD steps from the true weight vector $(\bar{w}_1, \bar{w}_2)$ as a function of N, σ and m.

Exercise 1 – Task 1.2

Learn the weights of a linear predictor for the maximum daytime temperature (label) from the minimum daytime temperature (feature) at some place in Finland. To achieve this goal, use the datapoints in

https://raw.githubusercontent.com/ieeespcasfinland/ieeespcasfinland.github.io/main/FMIData.csv

Questions ?

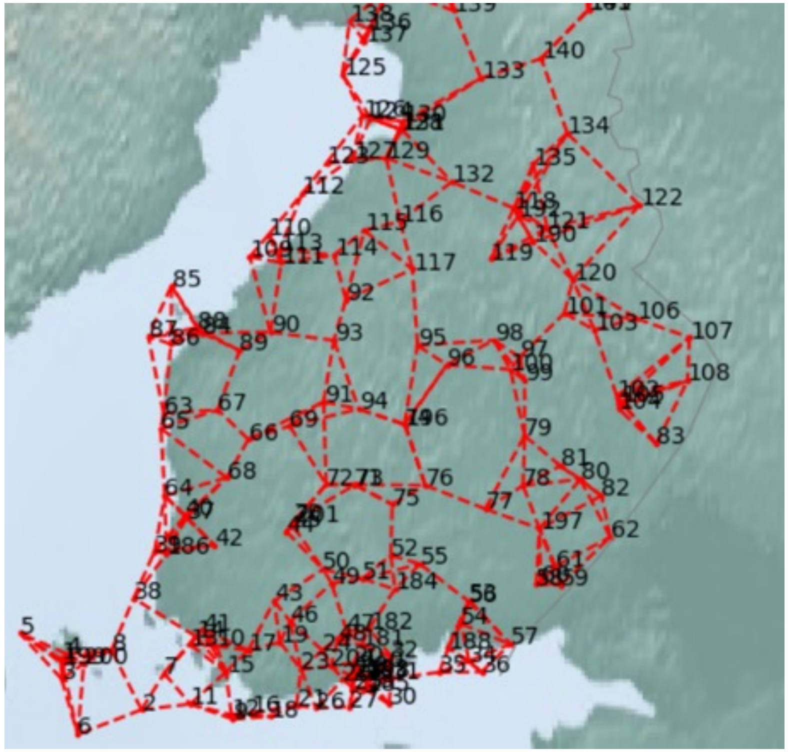
5 Min Break

11:55

Exercise 2.

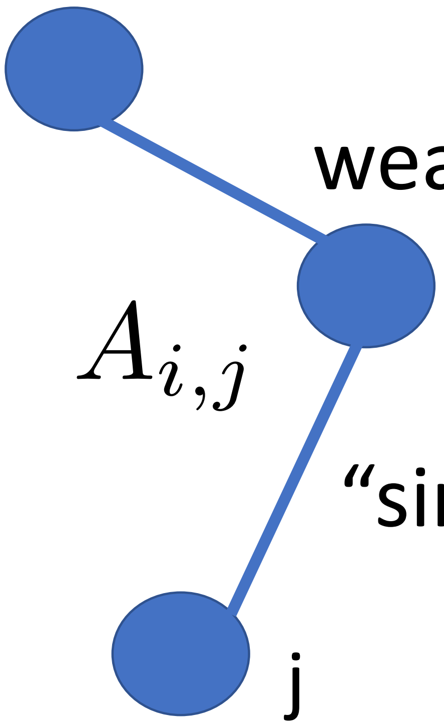
- store networked data and models using networkx.Graph object
- add local datasets and model parameters as node attributes
- combine GD with a network averaging algorithm to collaboratively linear hypothesis from a network of local datasets

Network of Weather Data



```
2021-04-24,1.2,3.8,60.30373,25.54916,Porvoo Kilpilahti satama
2021-04-25,1.0,6.5,60.30373,25.54916,Porvoo Kilpilahti satama
2021-04-26,1.4,5.1,60.30373,25.54916,Porvoo Kilpilahti satama
2021-04-27,1.5,5.3,60.30373,25.54916,Porvoo Kilpilahti satama
2021-04-28,0.2,6.5,60.30373,25.54916,Porvoo Kilpilahti satama
2021-04-29,-1.2,7.7,60.30373,25.54916,Porvoo Kilpilahti satama
2021-04-30,-1.0,9.0,60.30373,25.54916,Porvoo Kilpilahti satama
2021-04-11,-1.7,7.7,60.12735,19.90038,Jomala Maarianhamina lentoasema
2021-04-12,3.9,7.5,60.12735,19.90038,Jomala Maarianhamina lentoasema
2021-04-13,-0.2,5.2,60.12735,19.90038,Jomala Maarianhamina lentoasema
2021-04-14,-3.3,6.0,60.12735,19.90038,Jomala Maarianhamina lentoasema
2021-04-15,-4.4,9.0,60.12735,19.90038,Jomala Maarianhamina lentoasema
2021-04-16,-0.2,11.5,60.12735,19.90038,Jomala Maarianhamina lentoasema
2021-04-17,0.3,14.5,60.12735,19.90038,Jomala Maarianhamina lentoasema
```

The Empirical Graph.



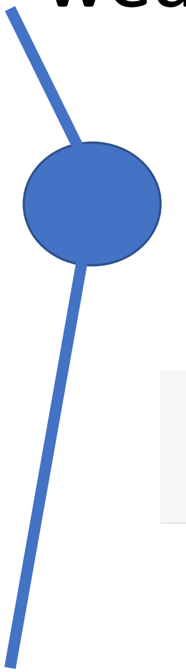

```
G_FMI = nx.Graph()
G_FMI.add_nodes_from(range(0,num_stations))
```

```
G_FMI = add_edges(G_FMI,total_neigh=4)
```

edge weights $A_{i,j}$ quantify
“statistical similarities”

Attaching Local Datasets to Nodes

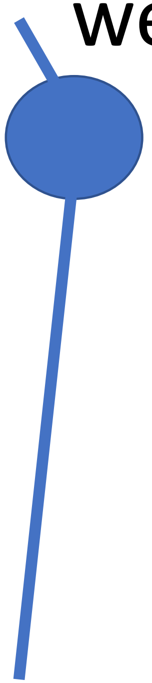
weather station i



$$\mathbf{X}^{(i)} = \left(\mathbf{x}^{(i,1)}, \ldots, \mathbf{x}^{(i,m_i)}\right)^T \text{, and } \mathbf{y}^{(i)} = \left(y^{(i,1)}, \ldots, y^{(i,m_i)}\right)^T.$$

```
G_FMI.nodes[i]['X'] = df.min_temp.to_numpy().reshape(-1,1)
G_FMI.nodes[i]['y'] = df.max_temp.to_numpy()  # label vecto
```

Attaching Linear Model to Nodes



weather station i

$$\mathbf{X}^{(i)} = \left(\mathbf{x}^{(i,1)}, \ldots, \mathbf{x}^{(i,m_i)}\right)^T \text{, and } \mathbf{y}^{(i)} = \left(y^{(i,1)}, \ldots, y^{(i,m_i)}\right)^T.$$

$$h^{(i)}(\mathbf{x}) = \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}$$

```
G.nodes[iter_node]["w"] = np.zeros(X.shape[1])
```

```
G_FMI.nodes[i]['X'] = df.min_temp.to_numpy().reshape(-1,1)
G_FMI.nodes[i]['y'] = df.max_temp.to_numpy()  # label vect
```

Learning a Global Model

$$\mathbf{w}^{(0)} = \mathbf{w}^{(1)} = \ldots = \mathbf{w}^{(m-1)} = \mathbf{w}$$

learn weight vector **w** via GD applied to linear regression with pooling all local datasets

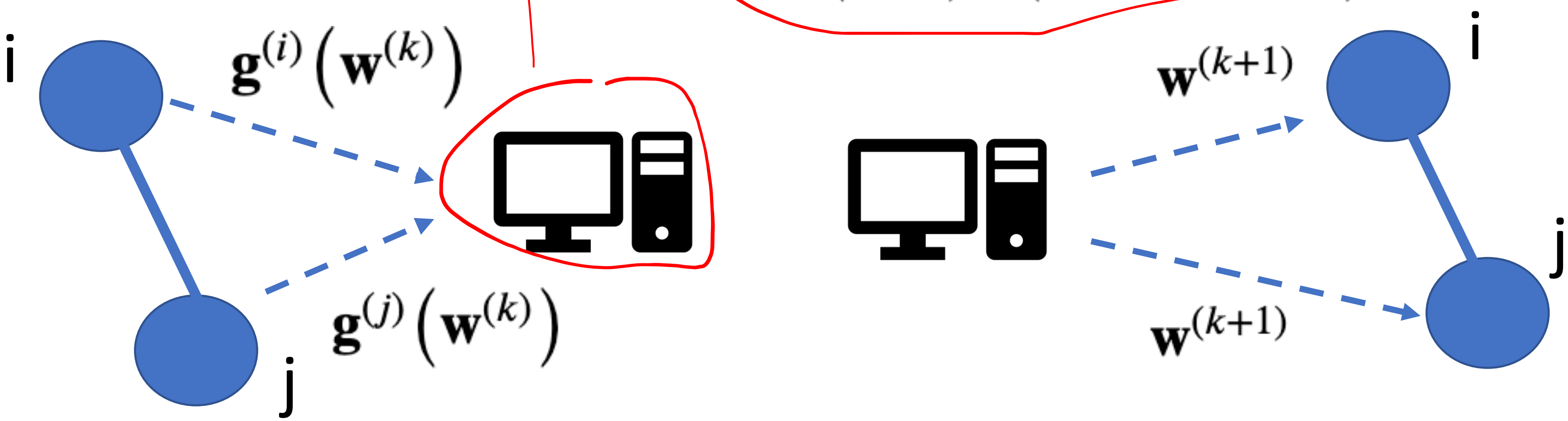
$$\mathbf{X} = \left(\left(\mathbf{X}^{(0)}\right)^T, \ldots, \left(\mathbf{X}^{(m-1)}\right)^T\right)^T, \mathbf{y} = \left(\left(\mathbf{y}^{(0)}\right)^T, \ldots, \left(\mathbf{y}^{(m-1)}\right)^T\right)^T$$

Centralized Federated Learning

GD step for linear regression on pooled local datasets:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \alpha(1/m)\sum_{i=0}^{m-1} g^{(i)}\left(\mathbf{w}^{(k)}\right)$$

local gradient $\quad g^{(i)}(\mathbf{w}) := -2\left(\mathbf{X}^{(i)}\right)^{T}\left(\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}\right)$



Distributed Federated Learning

GD step for linear regression on pooled local datasets:

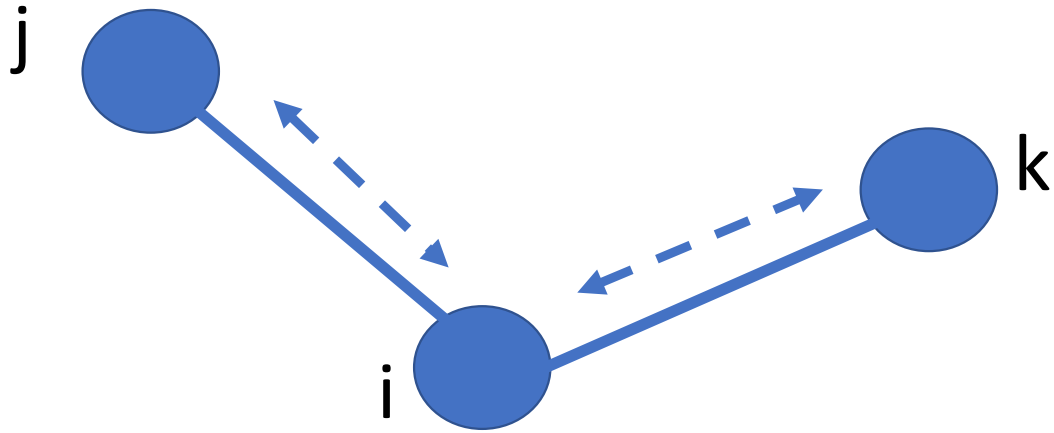
$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \alpha(1/m)\sum_{i=0}^{m-1} g^{(i)}\left(\mathbf{w}^{(k)}\right)$$

local gradient $g^{(i)}(\mathbf{w}) := -2\left(\mathbf{X}^{(i)}\right)^T\left(\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}\right)$

use network averaging to (approximately) compute

$$(1/m)\sum_{i=0}^{m-1} g^{(i)}\left(\mathbf{w}^{(k)}\right)$$

Network Averaging



$$\mathbf{g}^{(i)}(r+1) = W_{i,i}\mathbf{g}^{(i)}(r) + W_{i,j}\mathbf{g}^{(j)}(r) + W_{i,k}\mathbf{g}^{(k)}(r)$$

for suitable choice of weights $W_{i,j}$,

$$\lim_{r\to\infty} \mathbf{g}^{(i)}(r+1) = (1/m)\sum_{i'=0}^{m-1} \mathbf{g}^{(i')}(0)$$

Algorithm 1

1. init weights to zero at all nodes

```
G.nodes[iter_node]["w"] = np.zeros(X.shape[1])
```

2. repeat for N_GD times:

   2.1 compute local gradients at all nodes

```
# compute local gradient for current weight vector at node iter_node
G.nodes[iter_node]["g"] =  -2*X.T.dot(y - X.dot(G.nodes[iter_node]["w"]))
```

   2.2 N_AV iterations of network averaging

   2.3 do a GD step at each node

```
for iter_node in G.nodes(data=False):
    G.nodes[iter_node]["w"] = G.nodes[iter_node]["w"] - (learning_rate * G.nodes[iter_node]["g"] )
```

Exercise 2 – Task 2.1

Try out Algorithm 1 for a toy networked dataset consisting of two clusters/blocks

study effect of having an edge between clusters, varying number of GD steps, varying number of network averaging iterations

Exercise 2 – Task 2.2

Try out Algorithm 1 for a networked data obtained from Finnish meteorological institute (Demo code shows to load this data)

Questions ?

5 Min Break

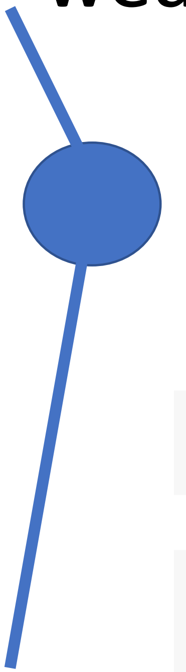
12:26

Exercise 3.

- learn different local (personalized) models for nodes
- couple local models using total variation (TV)
- GD for TV regularized local linear regression

Networked Data and Models.

weather station i



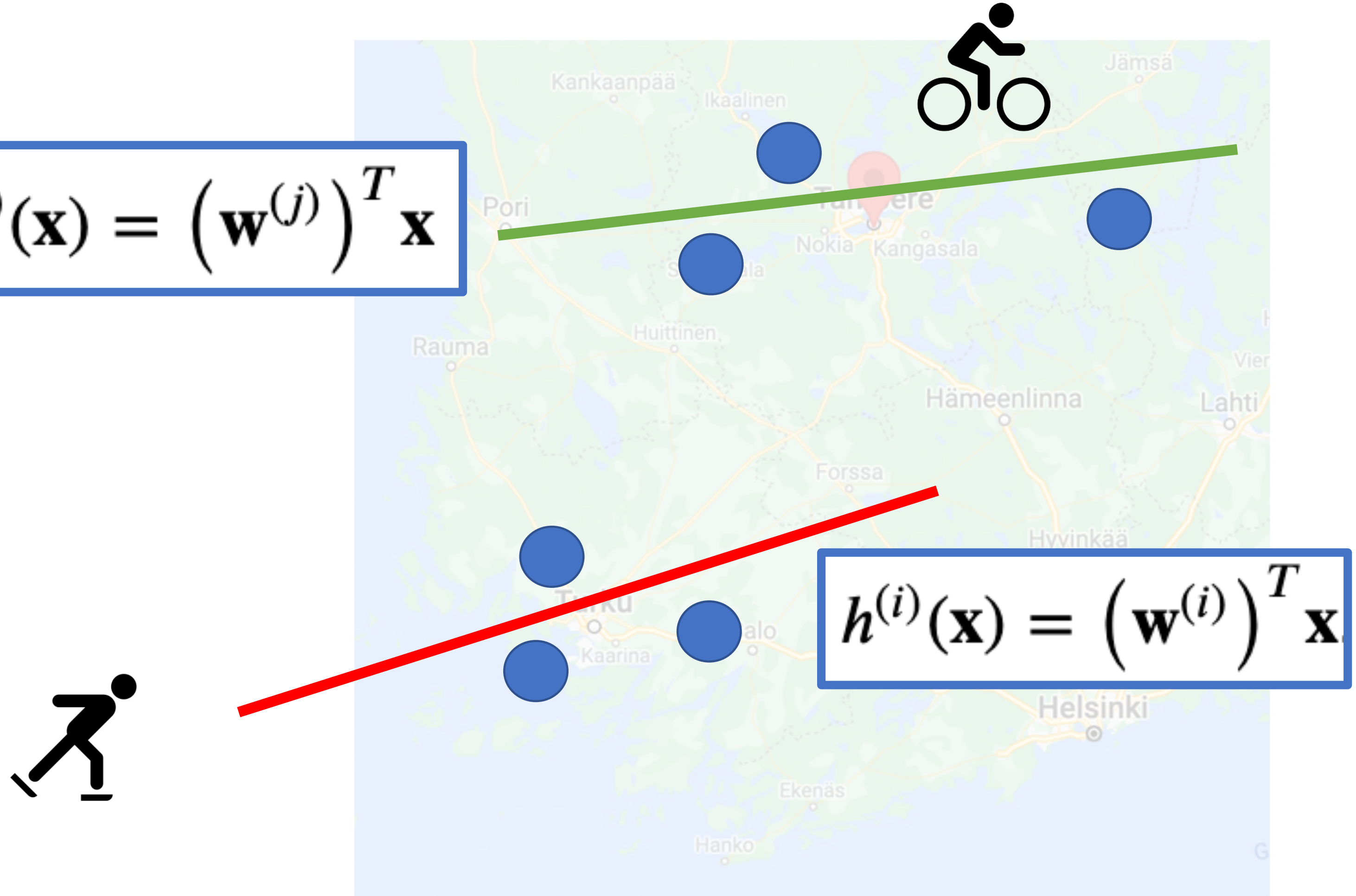
$$\mathbf{X}^{(i)} = \left(\mathbf{x}^{(i,1)}, \ldots, \mathbf{x}^{(i,m_i)}\right)^T \text{, and } \mathbf{y}^{(i)} = \left(y^{(i,1)}, \ldots, y^{(i,m_i)}\right)^T.$$

$$h^{(i)}(\mathbf{x}) = \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}$$

```
G.nodes[iter_node]["w"] = np.zeros(X.shape[1])
```

```
G_FMI.nodes[i]['X'] = df.min_temp.to_numpy().reshape(-1,1)
G_FMI.nodes[i]['y'] = df.max_temp.to_numpy()  # label vecto
```

Learn Personalized Models



$$h^{(j)}(\mathbf{x}) = \left(\mathbf{w}^{(j)}\right)^T \mathbf{x}$$

$$h^{(i)}(\mathbf{x}) = \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}$$

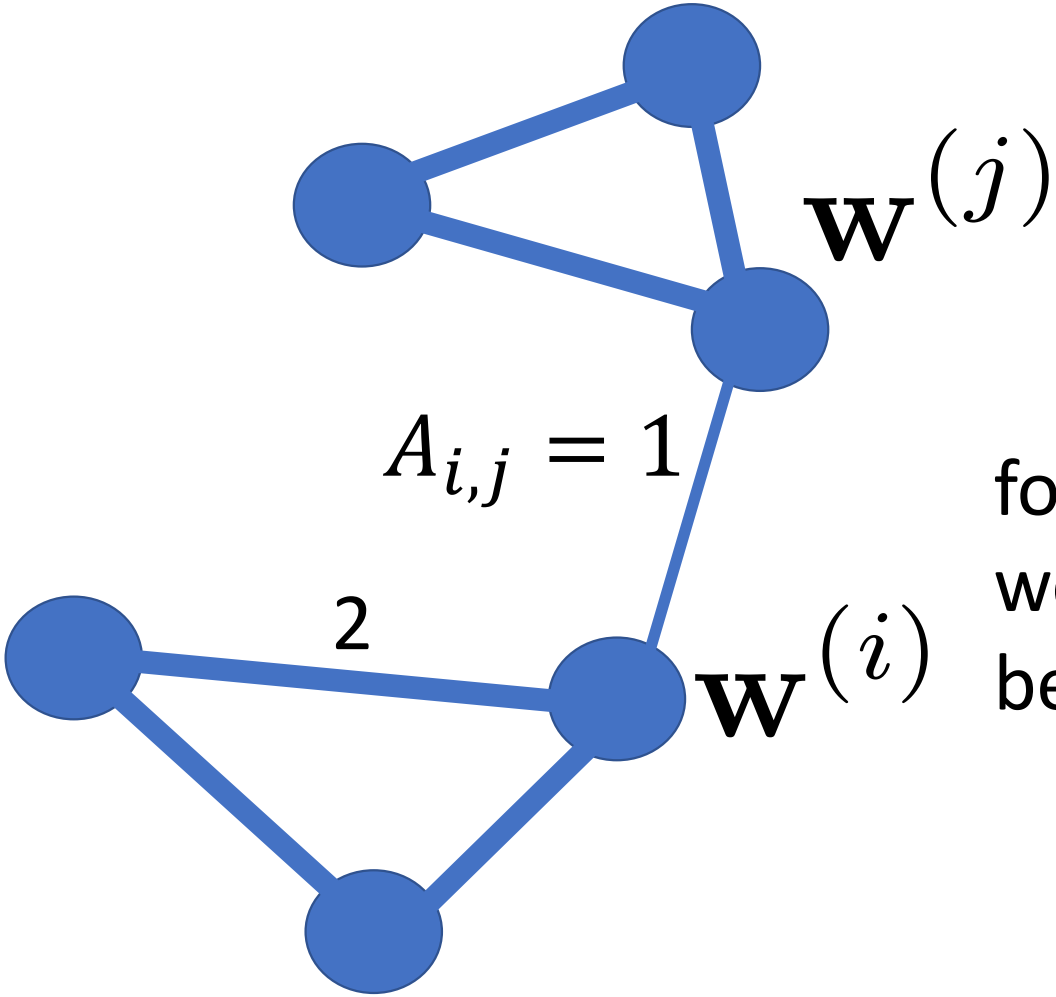
Local Linear Regression

$$\min_{\mathbf{w}^{(i)} \in \mathbb{R}^n} (1/m_i) \left\| \mathbf{y}^{(i)} - \mathbf{X}^{(i)} \mathbf{w}^{(i)} \right\|_2^2.$$

we could learn local weight vector by solving
local linear regression for each node

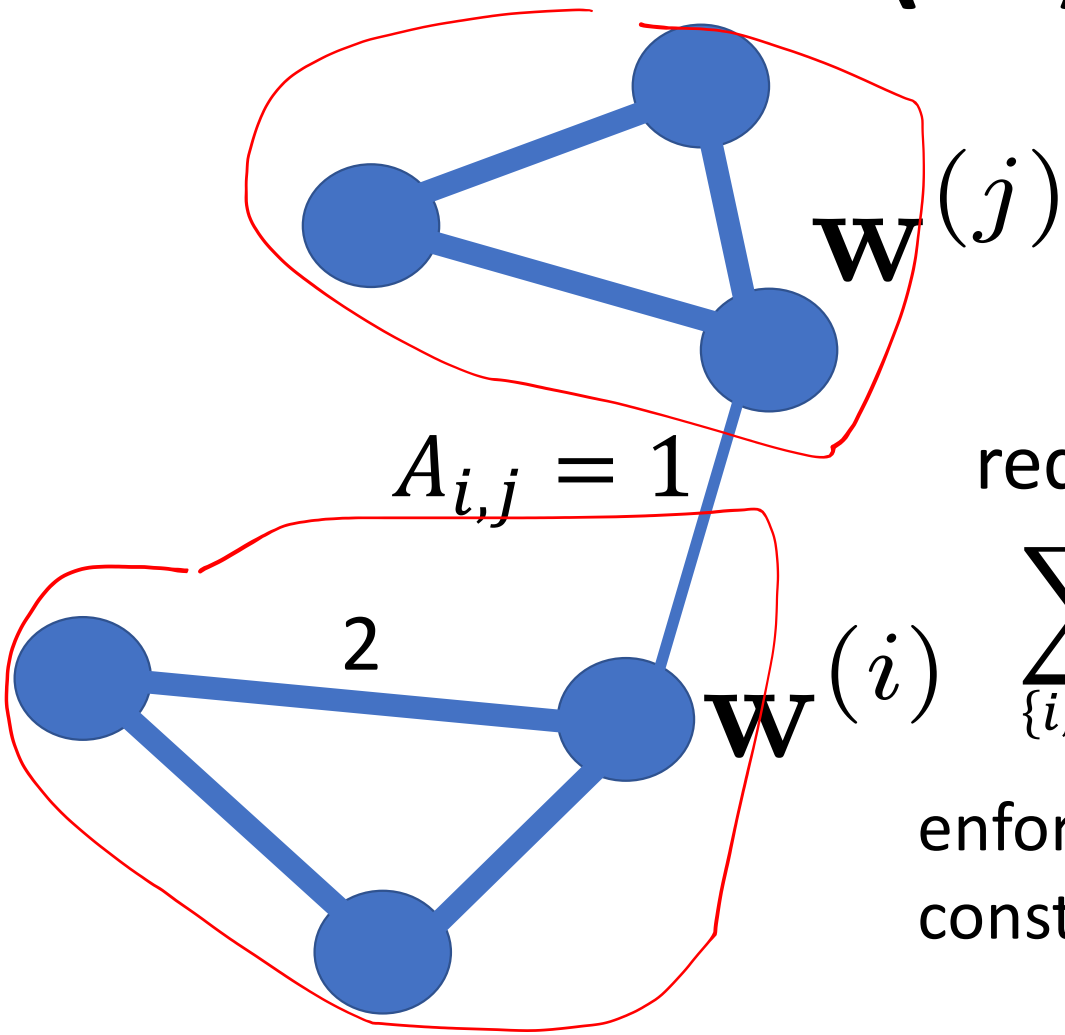
not a good idea if local dataset is too small (why?)

Clustering Assumption



force weight vectors of well connected nodes to be similar

Total Variation (TV)



requiring small TV

$$\sum_{\{i,j\}} A_{i,j} \left\| \mathbf{w}^{(i)} - \mathbf{w}^{(j)} \right\|^2$$

enforces weights to be nearly constant over clusters

TV-Regularized Linear Regression

$$\min_{\substack{\mathbf{w}^{(0)}, w^{(1)} \ldots, \\ w^{(m-1)}}} \sum_i (1/m_i) \left\| \mathbf{y}^{(i)} - \mathbf{X}^{(i)} \mathbf{w}^{(i)} \right\|_2^2 + \lambda_{\mathrm{TV}} \sum_{\{i,j\}} A_{i,j} \| \mathbf{w}^{(i)} - \mathbf{w}^{(j)} \|_2^2.$$

increasing λ_{TV}

"clusteredness"

local training errors

Gradient Descent

$$\min_{\mathbf{w}^{(0)},\ldots,} \sum_{i} (1/m_i) \left\| \mathbf{y}^{(i)} - \mathbf{X}^{(i)} \mathbf{w}^{(i)} \right\|_2^2 + \lambda_{\mathrm{TV}} \sum_{\{i,j\}} A_{i,j} \| \mathbf{w}^{(i)} - \mathbf{w}^{(j)} \|_2^2 .$$

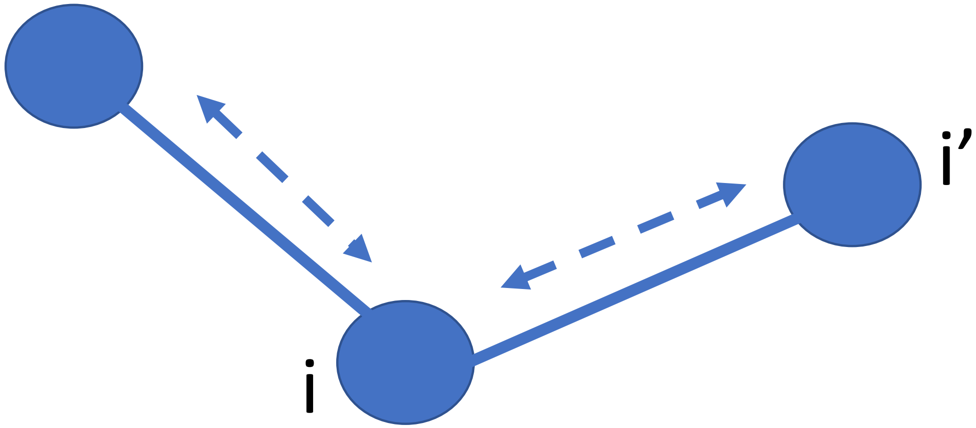
objective function is smooth and convex;

-> can be solved iteratively using GD steps

$\neq 0$

$$\mathbf{w}^{(i,k+1)} = \mathbf{w}^{(i,k)} - \alpha \left(-(2/m_i) \left(\mathbf{X}^{(i)} \right)^T \left(\mathbf{y}^{(i)} - \mathbf{X}^{(i)} \mathbf{w}^{(i,k)} \right) + 2\lambda_{\mathrm{TV}} \sum_{i' \neq i} A_{i,i'} \left(\mathbf{w}^{(i,k)} - \mathbf{w}^{(i',k)} \right) \right)$$

GD Step as Message Passing



neighb.

$$\mathbf{w}^{(i,k+1)} = \mathbf{w}^{(i,k)} - \alpha\left(-(2/m_i)\left(\mathbf{X}^{(i)}\right)^T\left(\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i,k)}\right) + 2\lambda_{\mathrm{TV}}\sum_{i' \neq i} A_{i,i'}\left(\mathbf{w}^{(i,k)} - \mathbf{w}^{(i',k)}\right)\right)$$

Algorithm 2

1. init weights to zero all nodes

```
G.nodes[iter_node]["w"] = np.zeros(X.shape[1])
```

2. repeat for N_GD times:

   update local weights at all nodes as

$$\mathbf{w}^{(i,k+1)} = \mathbf{w}^{(i,k)} - \alpha \left(-(2/m_i)\left(\mathbf{X}^{(i)}\right)^T \left(\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i,k)}\right) + 2\lambda_{\mathrm{TV}} \sum_{i' \neq i} A_{i,i'} \left(\mathbf{w}^{(i,k)} - \mathbf{w}^{(i',k)}\right) \right)$$

Exercise 3 – Task 3.1

Try out Algorithm 2 for a toy networked dataset consisting of two clusters/blocks

study the effect of varying edge weight between clusters, varying number of GD steps, varying regularization parameter λ_{TV}

Exercise 3 – Task 3.2

Try out Algorithm 2 for a networked data obtained from Finnish meteorological institute (Demo code shows to load this data)

Convergence of Alg. 2

$$\mathbf{w}^{(i,k+1)} = \mathbf{w}^{(i,k)} - \alpha \left(-(2/m_i)\left(\mathbf{X}^{(i)}\right)^T \left(\mathbf{y}^{(i)} - \mathbf{X}^{(i)}\mathbf{w}^{(i,k)}\right) + 2\lambda_{\mathrm{TV}} \sum_{i' \neq i} A_{i,i'} \left(\mathbf{w}^{(i,k)} - \mathbf{w}^{(i',k)}\right) \right)$$

is of the form $\mathbf{Q}$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - 2\alpha\left(\mathbf{Q}\mathbf{w}^{(k)} - \mathbf{q}\right)$$

convergence depends on the **eigenvalue spread of Q**

System Matrix of Alg. 2

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - 2\alpha\left(\mathbf{Q}\mathbf{w}^{(k)} - \mathbf{q}\right)$$

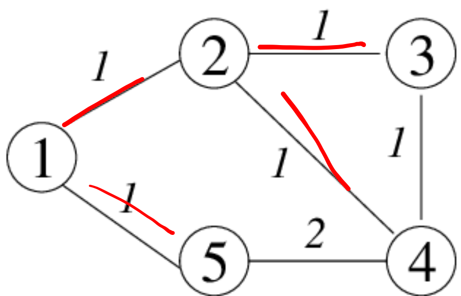
convergence depends on the **eigenvalue spread of Q**

$$\mathbf{Q} = \underbrace{\mathbf{X}^T\mathbf{X}}_{\text{data}} + \lambda_{TV} \underbrace{\mathbf{L}}_{\text{network}}$$

"block" Laplacian matrix

"stacked" feature matrix

$$\mathbf{X} = \begin{bmatrix} X^{(1)} & 0 & 0 & \cdots & 0 \\ 0 & X^{(2)} & 0 & \cdots & 0 \\ 0 & 0 & X^{(3)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & X^{(\mathcal{V})} \end{bmatrix}$$



$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 4 & -2 \\ -1 & 0 & 0 & -2 & 3 \end{pmatrix}$$

[Algorithms, Graph Theory, and Linear Equations in Laplacian Matrices](#)
D. Spielman

What's Next ?

Prof. Konstantin Avratchenkov, INRIA

Lecture: Basics of spectral graph theory

Date and Time: Tue., 29.03.2022 at 11:00- 13:00 (local Helsinki time)

zoom link: TBA

Personal website

Enjoy our Seasonal School!

Don’t hesitate to reach out to me!