EE-559
Deep Learning

slido

Have you completed the practice exercises for the first two lectures?

ⓘ Start presenting to display the poll results on this slide.

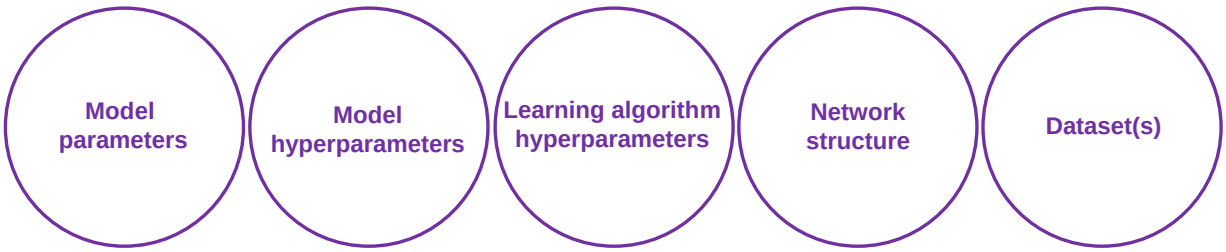
What's on today?

- The design space: the (many) choices for a model
- Training the model: how to fit the model to a dataset
- Model performance: how well a model predicts
- Classification: binary and multi-class problems
- Real-world impact: the cost of model fitting and deployment
- Exercises: the full training pipeline and the impact of data

The design space

$$y = f(x; \Theta)$$

The design space: choices for a model



$$\{x_i, y_i\}_{i=1}^{N}$$

Datasets

Training set

to determine the **model parameters**

minimize the loss

Validation set

to determine the **hyperparameters**

heuristics or search

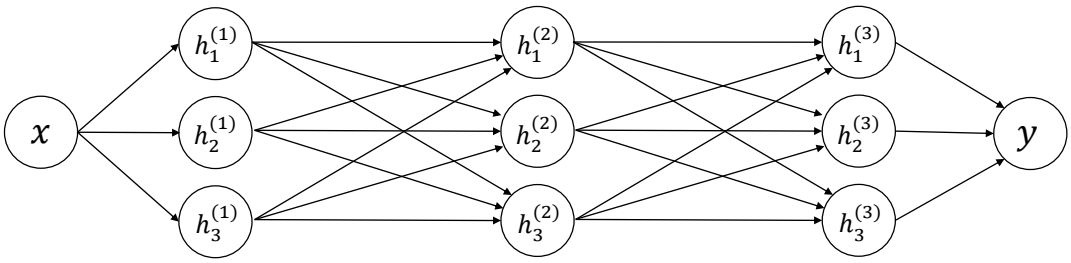
Testing set

to measure the ability **to generalize**

model performance

Training the model

Our ‘deep’ neural network



$$y = \Theta_0^{(3)} + \boldsymbol{\Theta}^{(3)} a\left[\boldsymbol{\Theta}_0^{(2)} + \boldsymbol{\Theta}^{(2)} a\left[\boldsymbol{\Theta}_0^{(1)} + \boldsymbol{\Theta}^{(1)} a\left[\boldsymbol{\Theta}_0^{(0)} + \boldsymbol{\Theta}^{(0)} x\right]\right]\right]$$

how to determine the value of the model parameters?

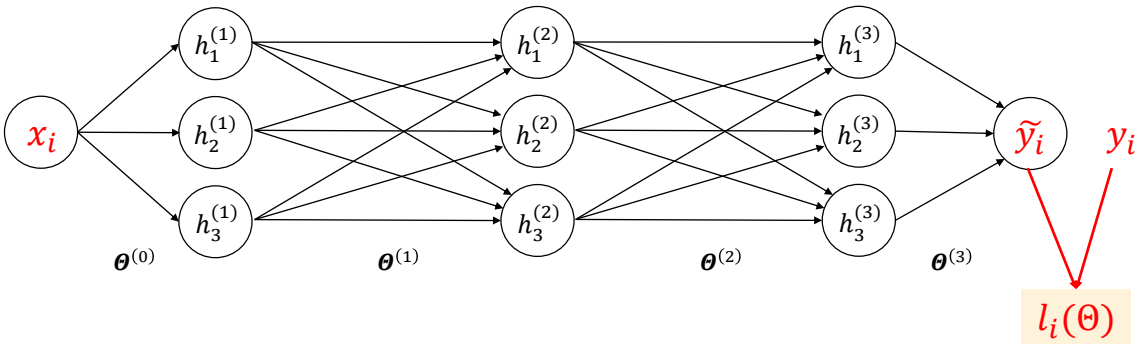
Datasets

to determine the **model parameters**

to determine the **hyperparameters**

to measure the ability **to generalize**

Computing the loss, the training signal



$\{x_i, y_i\}_{i=1}^N$ $L(\Theta) = \sum_{i=1}^{N} l_i(\Theta)$ e.g. $L(\Theta) = \sum_{i=1}^{N} (f(x_i; \Theta) - y_i)^2$

Determining the model parameters

initialize parameters → changes that decrease $L(\Theta)$ on average

randomly

based on $\frac{\partial L(\Theta)}{\partial \Theta}$

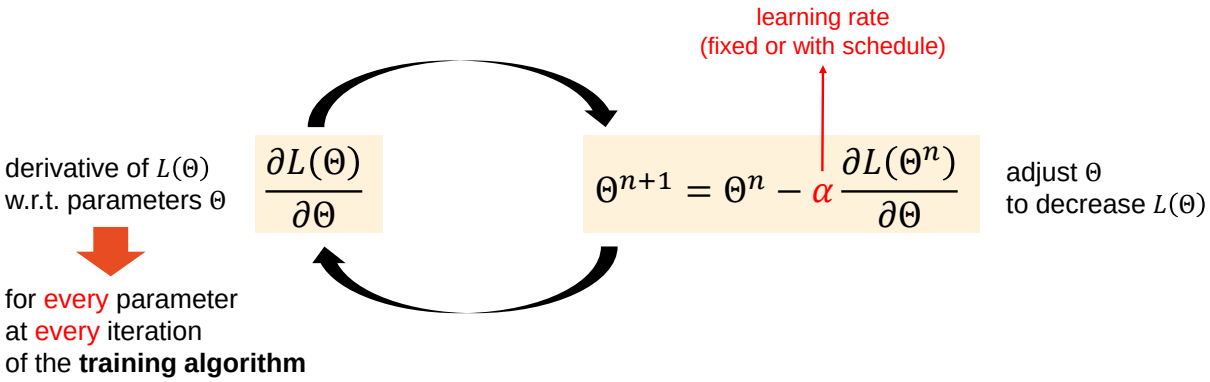
$$\frac{\partial L(\Theta)}{\partial \Theta} = \begin{bmatrix} \frac{\partial L(\Theta)}{\partial \Theta_0} \\ \frac{\partial L(\Theta)}{\partial \Theta_1} \\ \frac{\partial L(\Theta)}{\partial \Theta_2} \\ \dots \\ \frac{\partial L(\Theta)}{\partial \Theta_P} \end{bmatrix}$$

gradient calculated from $\{x_i, y_i\}_{i=1}^{N}$

Concepts: Initialization, vanishing gradient, gradient explosion, gradient of the loss with respect to the parameters (at current position)

1,000,000,000,000

Gradient descent



how to calculate the gradients efficiently?

Concepts: Global minimum, local minima, saddle points, convexity, concavity, trajectory of the gradient, exploration of the parameter space

Stochastic gradient descent (SGD)

gradient calculated from $\{x_i, y_i\}_{i=1}^N$

$$\Theta^{n+1} = \Theta^n - \alpha \sum_{i=1}^{N} \frac{\partial l_i(\Theta^n)}{\partial \Theta}$$

gradient estimate calculated from a **batch** (*random subset of data*) B^n
with $|B^n| \in \{1, 2, \dots, N\}$
where $|\,.\,|$ is the cardinality of a set

$$\Theta^{n+1} = \Theta^n - \alpha \sum_{i \in B^n} \frac{\partial l_i(\Theta^n)}{\partial \Theta}$$

set of batch indices
at iteration n

Concepts:
Batch, epoch, noise (randomness) in the gradient, deterministic descent on a changing loss, normalisation

Momentum term

$$\Theta^{n+1} = \Theta^n - \alpha\, \boldsymbol{m}^{n+1}$$

α learning rate

$$\boldsymbol{m}^{n+1} = \beta \boldsymbol{m}^n + (1-\beta) \sum_{i \in B^n} \frac{\partial l_i(\Theta^n)}{\partial \Theta}$$

$\beta \in [0,1)$ controls the degree of smoothing of the gradient over time

effective learning rate
- increases if gradient directions over time are aligned
- decreases if gradient directions over time change

Concepts:
Weighted combination of gradients, reduced oscillations at valleys, Nesterov accelerated momentum

Normalizing the gradient

$$\Theta^{n+1} = \Theta^n - \alpha \frac{\partial L(\Theta^n)}{\partial \Theta}$$

$$\Theta^{n+1} = \Theta^n - \alpha \frac{\boldsymbol{m}^{n+1}}{\sqrt{\boldsymbol{v}^{n+1}} + \epsilon}$$

$$\boldsymbol{m}^{n+1} = \frac{\partial L(\Theta^n)}{\partial \Theta}$$

$$\boldsymbol{v}^{n+1} = \left(\frac{\partial L(\Theta^n)}{\partial \Theta}\right)^2$$

Concept:
Point-wise operations

Adaptive moment estimation (Adam)

$$\Theta^{n+1} = \Theta^n - \alpha \frac{\tilde{\boldsymbol{m}}^{n+1}}{\sqrt{\tilde{\boldsymbol{v}}^{n+1}} + \epsilon}$$

$$\tilde{\boldsymbol{m}}^{n+1} = \frac{\boldsymbol{m}^{n+1}}{1 - (\beta)^{n+1}}$$

$$\boldsymbol{m}^{n+1} = \beta \boldsymbol{m}^n + (1-\beta) \sum_{i \in B^n} \frac{\partial l_i(\Theta^n)}{\partial \Theta}$$

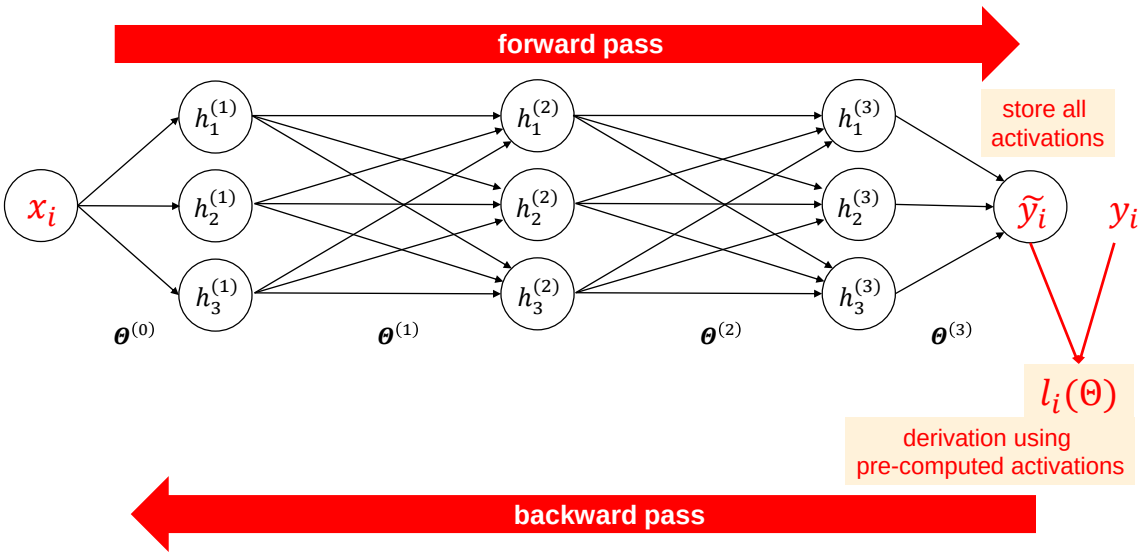
$$\tilde{\boldsymbol{v}}^{n+1} = \frac{\boldsymbol{v}^{n+1}}{1 - (\gamma)^{n+1}}$$

$$\boldsymbol{v}^{n+1} = \gamma \boldsymbol{v}^n + (1-\gamma) \sum_{i \in B^n} \left(\frac{\partial l_i(\Theta^n)}{\partial \Theta} \right)^2$$

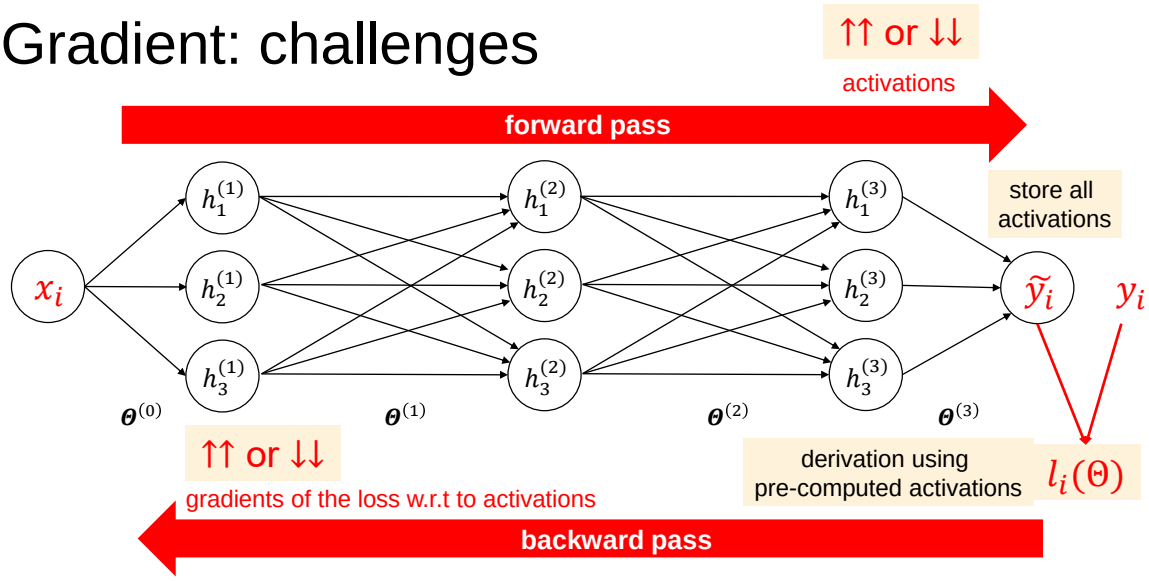
$\gamma, \beta \in [0,1)$

SGD is a special case of Adam ($\gamma = \epsilon = 0$)

Backpropagation algorithm



Gradient: challenges



Concepts: Different starting point for the optimization for each different initialization, gradients of the activations may "vanish" or "explode" (worsens with backward pass), problem worsens with depth

Exploding or vanishing gradient

$$y = \theta_0^{(3)} + \boldsymbol{\theta}^{(3)} a\left[\boldsymbol{\theta}_0^{(2)} + \boldsymbol{\theta}^{(2)} a\left[\boldsymbol{\theta}_0^{(1)} + \boldsymbol{\theta}^{(1)} a\left[\boldsymbol{\theta}_0^{(0)} + \boldsymbol{\theta}^{(0)} x\right]\right]\right]$$

$$\boldsymbol{f}^{(n)} = \boldsymbol{\theta}_0^{(n)} + \boldsymbol{\theta}^{(n)} \boldsymbol{h}^{(n)}$$
$$= \boldsymbol{\theta}_0^{(n)} + \boldsymbol{\theta}^{(n)} a\left[\boldsymbol{f}^{(n-1)}\right]$$

With ReLU, range of $h^{(n)}$ is half of range of $f^{(n-1)}$ (clipping below 0)

$\boldsymbol{\theta}_0^{(n)} = \mathbf{0}$
$\boldsymbol{\theta}^{(n)} \sim \mathrm{N}(\mu = 0, \sigma^2)$

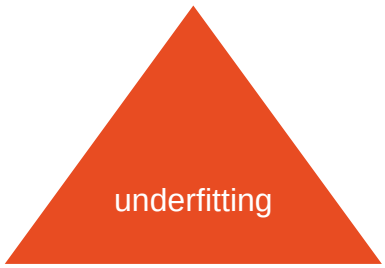
If $\sigma^2 \ll 1$ → much smaller magnitude than the input
If $\sigma^2 \gg 1$ → much larger magnitude than the input

Objective: to maintain the variance of $\boldsymbol{h}^{(n)}$ the same across every layer

Concepts: magnitudes of the gradient decrease (increase) uncontrollably during the backward pass, updates to the model become vanishingly small (vanishing) or unstable ("exploding")

Model performance

Under and overfitting



underperforming on the training set

Model is too simple
Model needs more training time and/or data
High bias

underperforming on the test set

Low-quality and/or small size training data
Model capacity used to model training data noise
High variance

Model performance: cause of errors

Choice of the model → unable to describe the true function → bias ↑

Choice of the training data → limited, noisy → variance ↑

Concepts: Sampling of the input space, choice of the hyperparameters, stochastic learning algorithm may not converge to the same solution each time

How to reduce bias and variance?

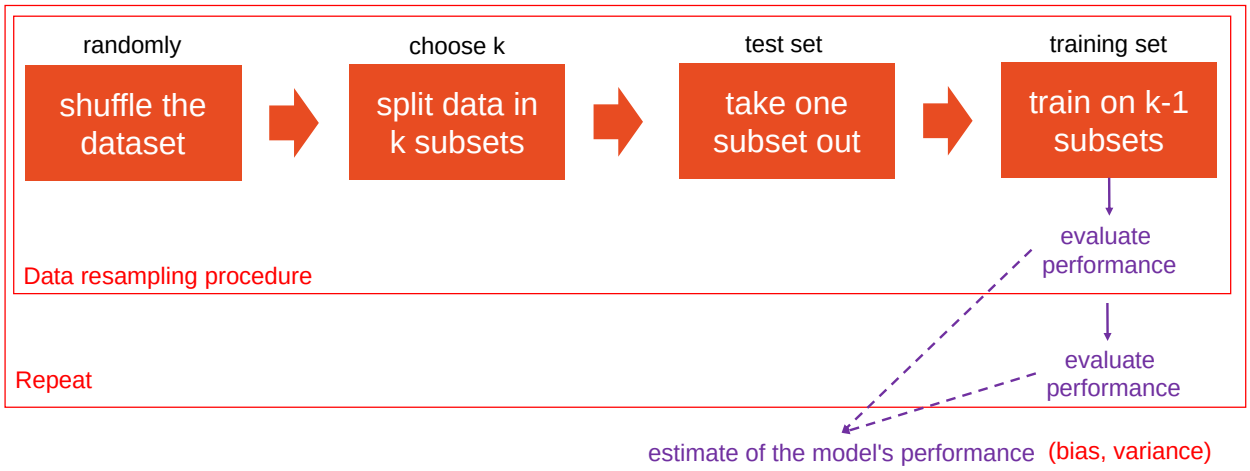
add hidden units and/or hidden layers — $D \uparrow \Rightarrow$ bias $\downarrow$

increase quantity and/or quality of training data — $N \uparrow \Rightarrow$ variance $\downarrow$

Given a fixed N
if $D \uparrow \Rightarrow$ bias $\downarrow$ but variance $\uparrow \Rightarrow$ trade-off

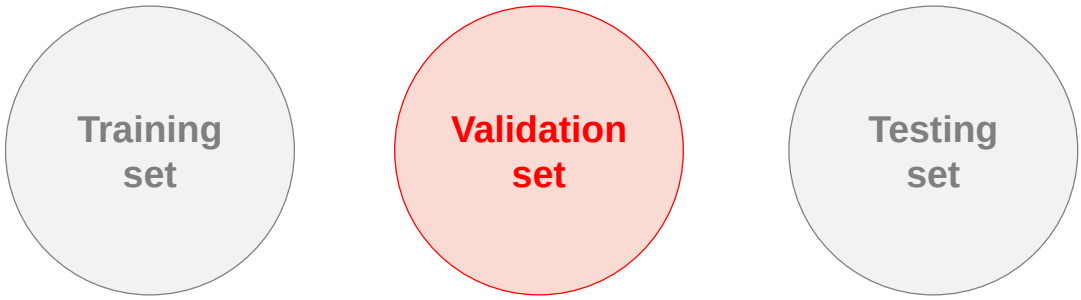
Concepts:
Bias-variance trade-off, a model capacity increase does not (necessarily) imply a test error reduction

k-fold cross validation



Concepts:
Aggregation of performance results, leave-p-out cross validation, nested cross-validation

Datasets

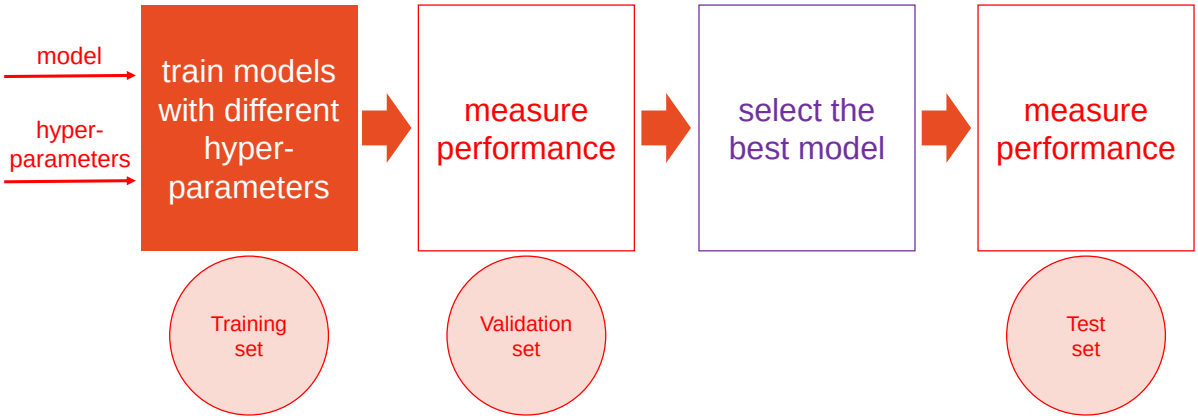


to determine the **model parameters**

to determine the **hyperparameters**

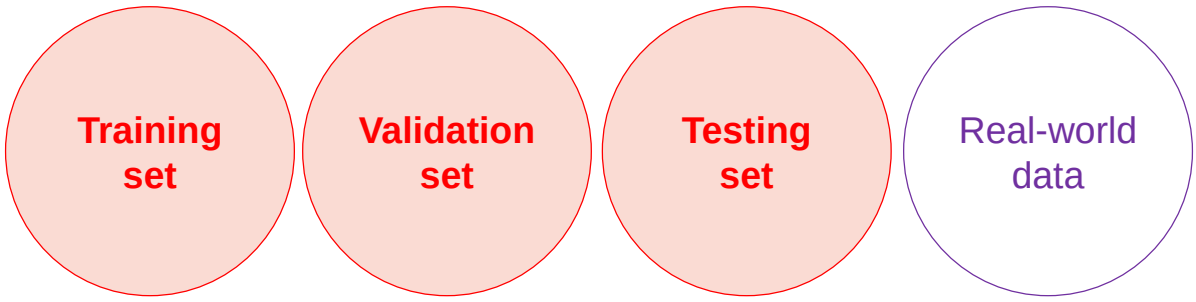
to measure the ability **to generalize**

Hyperparameters



Concepts:
Hyperparameter search, empirical choice, sampling the hyperparameter space

Data

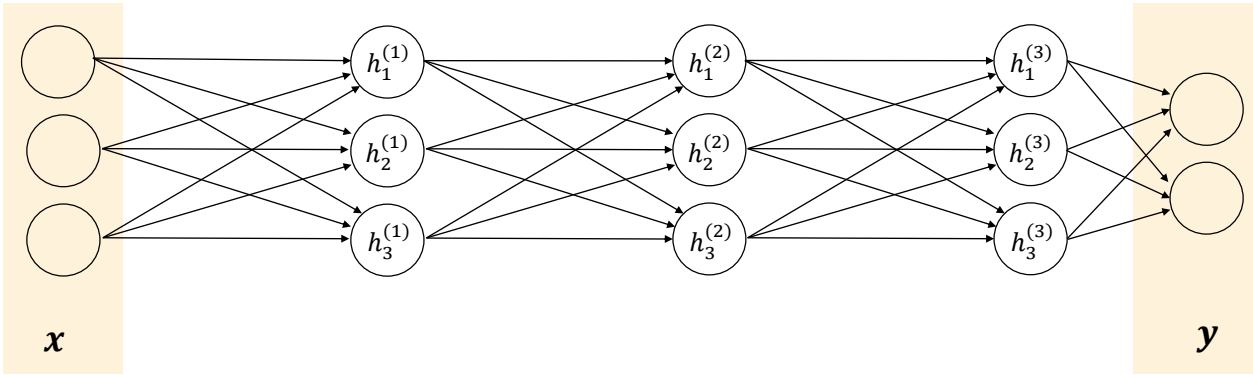


Classification

Recap: negative log-likelihood criterion

$$\Theta^* = \arg\min_{\Theta} \left[- \sum_{i=1}^{N} \log[P(y_i | f(x_i; \Theta))] \right]$$

Recap: network diagram



Binary classification

$y_i \in \{0,1\}$ 2 classes

Probability distribution (*Bernoulli*) over the output space:

$$P(y|\lambda) = \begin{cases} 1-\lambda & y_i = 0 \\ \lambda & y_i = 1 \end{cases} \qquad \lambda \in [0,1]$$

$$P(y|\lambda) = \lambda^y (1-\lambda)^{1-y}$$

Concepts:
Categories, labels

Binary cross-entropy loss

$\text{sig}(z) = \frac{1}{1 + e^{-z}}$ logistic sigmoid ($\mathbb{R} \rightarrow [0,1]$)

$P(y|x) = \text{sig}(f(x;\Theta))^{y}(1 - \text{sig}(f(x;\Theta)))^{1-y}$

$$L(\Theta) = \sum_{i=1}^{N} -y_i \log\left[\text{sig}\left(f(x_i;\Theta)\right)\right] - (1 - y_i)\log\left[1 - \text{sig}\left(f(x_i;\Theta)\right)\right]$$

inference $\hat{y} = 1$ if $\lambda > 0.5$ (and 0 otherwise)

Concepts:
Setting $f(x;\Theta)$ to predict λ of the Bernoulli distribution, negative log-likelihood of the training set

Multi-class classification

$y_i \in \{1, 2, \ldots, c, \ldots, C\}$ C classes

Probability distribution (*categorical*) over the output space:

$P(y = c) = \lambda_c$ $\lambda_1, \lambda_2, \ldots, \lambda_c, \ldots, \lambda_C$ probability of each category

$\lambda_c \in [0,1]$

$\sum_{c=1}^{C} \lambda_c = 1$

$\text{softmax}_c(\boldsymbol{z}) = \frac{e^{z_c}}{\sum_{c'=1}^{C} e^{z_{c'}}}$ softmax function

$P(y = c|x) = \text{softmax}_c\left(f(x;\Theta)\right)$

Concepts:
Exponential function ensures positivity, sum in the denominator ensures “sum to 1”

Multi-class cross-entropy loss

$$L(\Theta) = -\sum_{i=1}^{N} \log\left[\text{softmax}_{y_i}\left(f(x_i;\Theta)\right)\right]$$

$$= -\sum_{i=1}^{N} \text{f}_{y_i}(x_i;\Theta) - \log\left[\sum_{c\prime=1}^{C} e^{\text{f}_{c\prime}(x_i;\Theta)}\right]$$

output c' of the neural network

inference $\hat{y} = \arg\max_{c} P(y = c|f(x;\Theta^*))$

Concepts:
Categorical distribution of C possible classes, point estimate, most probable category

Cross-entropy loss

Kullback-Leibler (KL) divergence: distance between $q(z)$ and $p(z)$

$$D[q||p] = \int_{-\infty}^{+\infty} q(z)\log[q(z)]\,dz - \int_{-\infty}^{+\infty} q(z)\log[p(z)]\,dz$$

empirical data distribution observed at $\{y_i\}_{i=1}^{N}$

$$q(y) = \frac{1}{N}\sum_{i=1}^{N} \delta[y - y_i]$$

Dirac delta function

Concept:
Weighted sum of point masses

Cross-entropy loss

$$\Theta^* = \arg\min_{\Theta} \left[\int_{-\infty}^{+\infty} q(y) \log[q(y)]\, dy - \int_{-\infty}^{+\infty} q(y) \log[P(y|\Theta)]\, dy \right]$$

$$= \arg\min_{\Theta} \left[- \int_{-\infty}^{+\infty} q(y) \log[P(y|\Theta)]\, dy \right]$$

cross-entropy: amount of uncertainty in a distribution after taking into account what we already know from the other

$$= \arg\min_{\Theta} \left[- \int_{-\infty}^{+\infty} \left(\frac{1}{N} \sum_{i=1}^{N} \delta[y - y_i] \right) \log[P(y|\Theta)]\, dy \right]$$

$$= \arg\min_{\Theta} \left[- \frac{1}{N} \sum_{i=1}^{N} \log[P(y_i|\Theta)] \right] = \arg\min_{\Theta} \left[- \sum_{i=1}^{N} \log\left[P\left(y_i|f(x_i;\Theta)\right)\right] \right]$$

negative log-likelihood criterion

Negative log-likelihood and cross-entropy

negative log-likelihood criterion ⇔ cross-entropy criterion

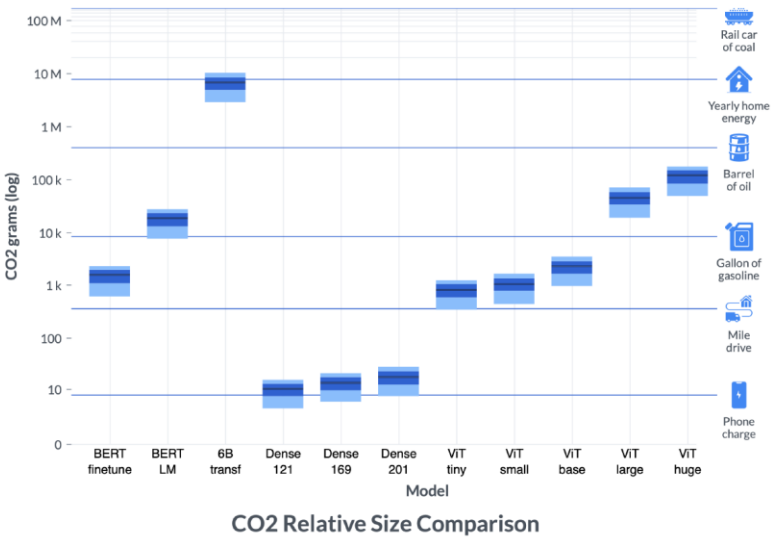
maximizing the data likelihood

minimizing the distance between the model and empirical data distributions

Concept:
Equivalence between negative log-likelihood and cross-entropy criteria

Real-world impact

The cost of model fitting



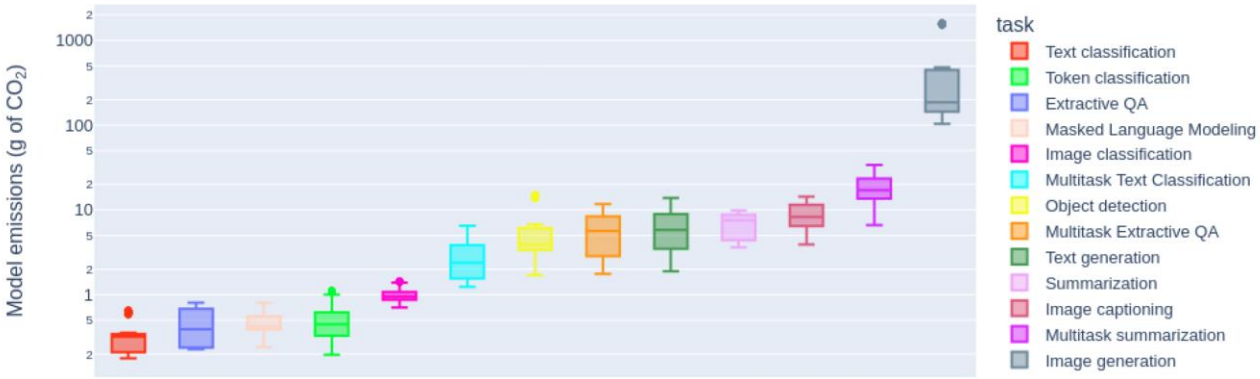
Measuring the carbon intensity of AI in cloud instances

measure operational carbon emissions data per energy unit
- location-based
- time-specific marginal emissions

natural language processing
computer vision applications

arXiv:2206.05229

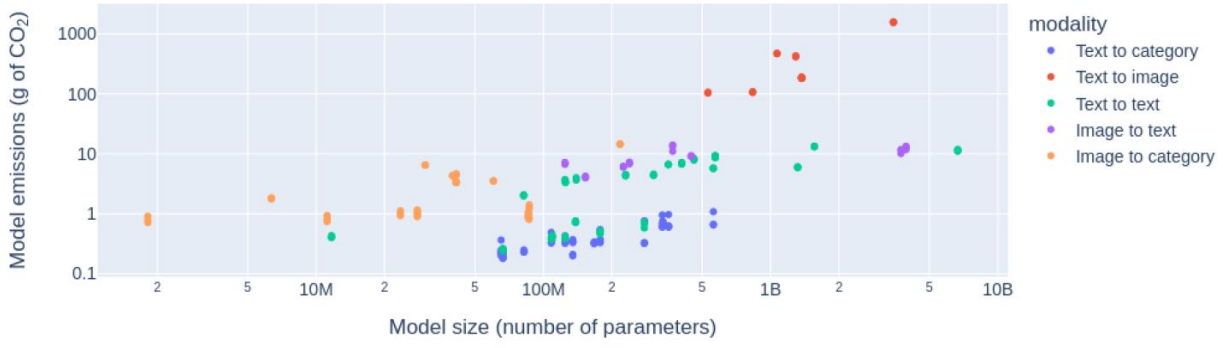
The cost of deployment



for 1000 queries

arXiv:2311.16863

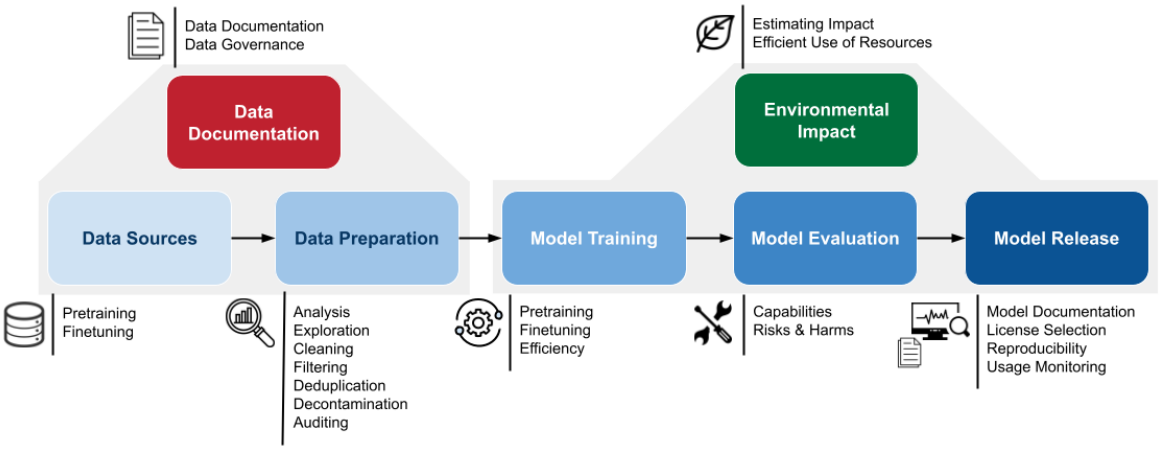
Carbon emitted for 1000 inferences



Note - logarithmic scales in both axes

arXiv:2311.16863

Model development



https://fmcheatsheet.org

Exercises

Today's practice exercises

You will experiment with a **full training pipeline**

- implementing the model architecture
- training and evaluation
- logging the metrics
- using different optimizers

Today's marked exercises

The impact of data on model performance

- data augmentation on a balanced dataset
- weighted sampler on an imbalanced dataset
- weighted cross-entropy loss to mitigate the imbalance

What did we learn today?

- The design space
- Training the model
- Model performance
- Classification
- Real-world impact
- Exercises

EE-559
Deep Learning

andrea.cavallaro@epfl.ch