Applied Deep Learning for Sequences Class 1 -- Intro

Yoav Goldberg
Bar Ilan University

Course Info

- Website: http://www.cs.biu.ac.il/~89-687
  - Discussions: on moodle forums.
- Office hours: by appointment, after class.
- Requirements:
  - Small "exercises" (must submit, not graded)
  - Assignments (programming, math or both)
  - Final Exam

Course Info

- Must pass exam to pass the course.
- Must pass all assignments to pass the course.
- Must submit all assignments on time.
    - Each student has 10 "late days" that you can choose how to allocate across all assignments. Beyond that, you will get 10 reduced points for every late day.
- Do not copy, do not hand in other person's code.
- Assignments are non trivial, start on time.

Who are you?

- How many are undergraduate? graduate?

Who are you?

- Who **did not** take ML course?

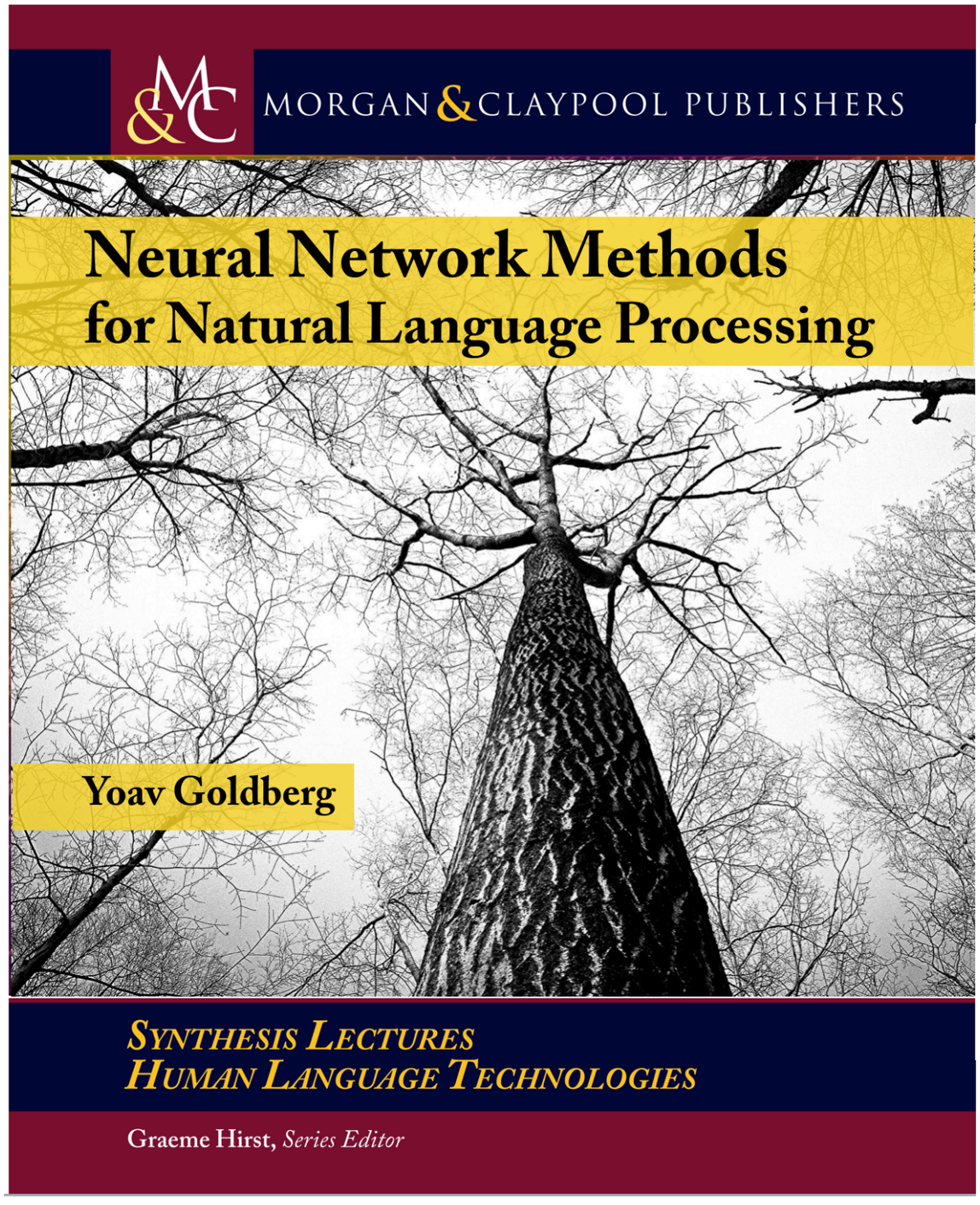
Who are you?

- Who takes (or previously took) NLP?

Course Info

- Background:
  - Linear Algebra
  - Calculus (differentiation)
  - A little bit of probability
  - Programming (python)
  - Machine Learning

Book



+ some newer stuff.

Course Info

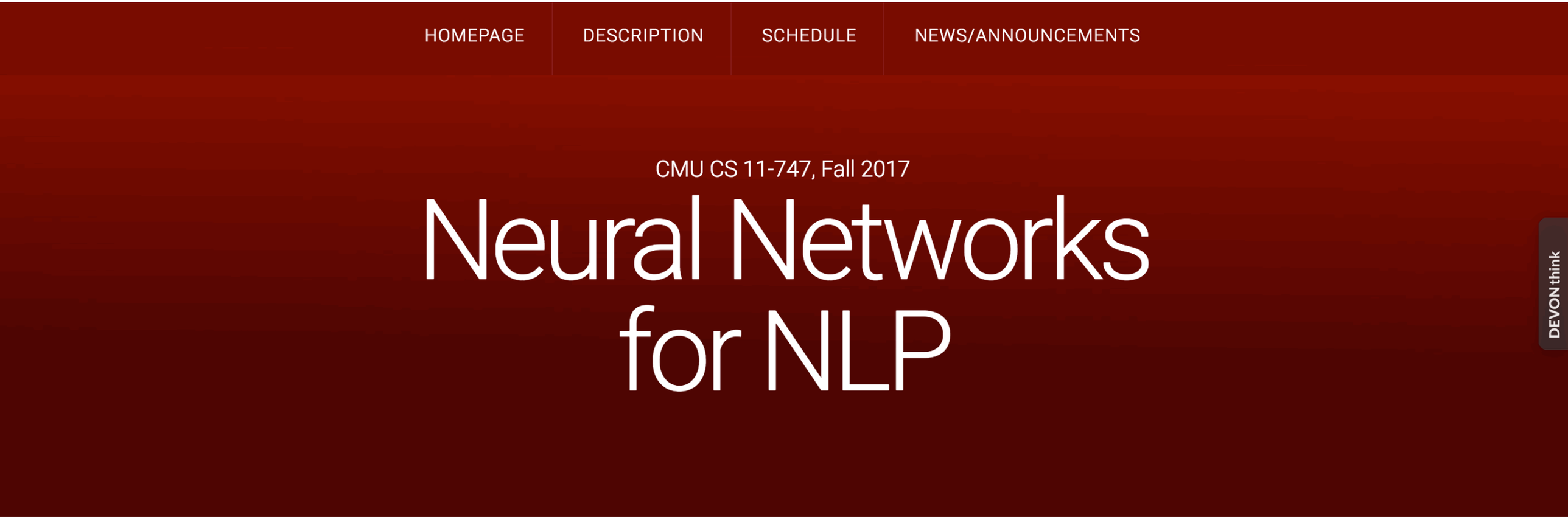
- This is not an easy course.

- It is probably an important one.

Good other course

http://www.phontron.com/class/nn4nlp2018/schedule.html

http://www.phontron.com/class/nn4nlp2021/schedule.html



What is Learning?

What is Learning?

- Making predictions
- Based on past observations
  - finding patterns in data
  - learning from data
  - generalization...

Types of Learning

- Supervised
  - Regression
  - Classification
    - Structured
- Unsupervised
  - Semi-supervised
- Reinforcement

Machine Learning

- Learning a mathematical function from x to y.
- Need to represent objects in the world as mathematical objects.
- Measurements / Features.
- Hypothesis class.
  - Inductive bias.

What is Deep Learning

What is Deep Learning

- Specific class of learning algorithms
- Rich hypothesis class (can represent borel measurable functions)
- Many "layers" of simple components.
- Find parameters with gradient-based methods (so everything needs to be differentiable)
- Minimal feature engineering (**learn** a good representation)

What are Sequences

- Sequential data is everywhere.

- Examples?

Types of Sequences

- Continuous / Numeric

- Discrete

Characteristics of Sequential Data

Characteristics of Sequential Data

- Sequential
- Items, "vocabulary", "alphabet"
- "time" dependence
- Locality
- Long-distance relations

Stretching the boundaries

- Are trees sequences?

Stretching the boundaries

- Are trees sequences?
- Can we think of trees as sequences?
- Pros? cons?

Stretching the boundaries

- Are **graphs** sequences?
- Can we think of **graphs** as sequences?
- Pros? cons?

Stretching the boundaries

- Are **images** sequences?
- Can we think of **images** as sequences?
- Pros? cons?

Kinds of Problems

Kinds of Problems

- Classification
    - Of sequences
    - Of items within sequences
- Similarity
    - Of sequences
    - Of items within sequences
- Segmentation
- Mapping
- Sequence to Sequence
- Predict next item ("language modeling")
- ...

This course

- Linear models
- Gradient-based learning, losses, regularization, feed-forward nets / MLP
- Representation learning
- Language models and word embeddings
- Multi-task learning
- 1D convolutions
- Recurrent Networks (plain, gated)
- Attention Networks (and self attention, transformers)
- Sequence to Sequence
- Adversarial Training, Contrastive Learning
- Pre-training, fine tuning.

This course

- Implementing neural networks
  - By hand
  - with a toolkit
- Efficiency
  - GPU vs CPU
  - Batching

This course

- How to **model** with neural networks?
- How to **describe** neural networks?
- What are the **benefits** of different architectures?
- What are the **limitations** of different architectures?
- What is **hard** to learn? what is **easy** to learn?
- What **we don't know** yet?

Learning 101

- Building blocks -- Linear Model -- Perceptron

Linear Model

Housing Data

(whiteboard)

Language Classification

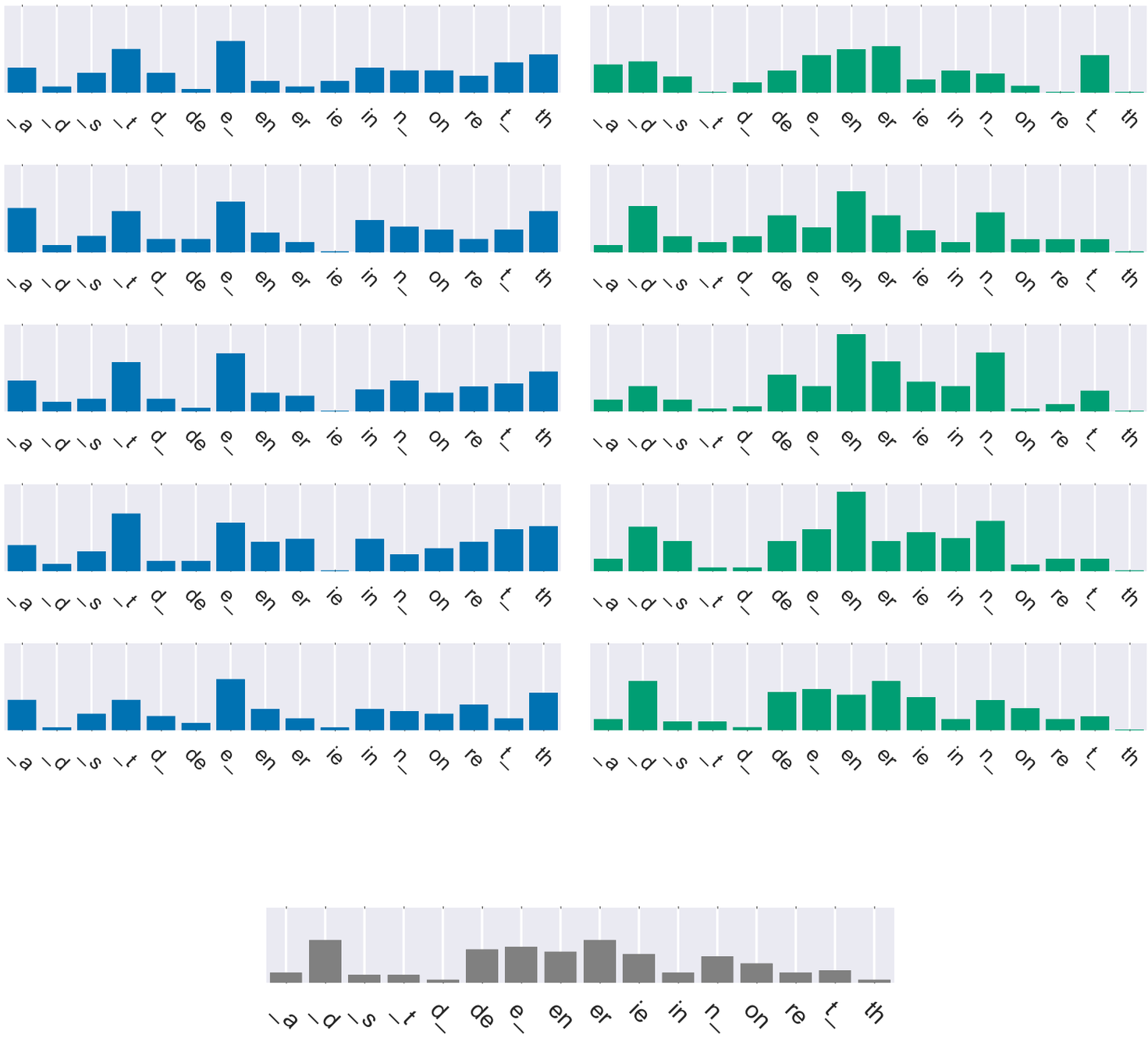
- Astrophysicists use the term "metal" to collectively describe all elements other than hydrogen and helium. In that sense, the metallicity of an object is the proportion of its matter made up of chemical elements other than hydrogen and helium.

- Ein einzelnes Atom dieser Elemente hat keine metallischen Eigenschaften; es ist kein Metall. Erst wenn mehrere solcher Atome miteinander wechselwirken und zwischen ihnen eine metallische Bindung besteht, zeigen solche Atomgruppen (cluster) metallische Eigenschaften.

Letter Bigrams

Letter Bigrams

Letter Bigrams



(whiteboard)

Summarize

- Simple hypothesis class -- linear model, perceptron

- See exercise 1 online.

Learning as Optimization

Problem Setup

Data:

$\mathbf{x_1}, \ldots, \mathbf{x_n}$ examples / instances / items

$\mathbf{y_1}, \ldots, \mathbf{y_n}$ labels

Desired:

$f(\mathbf{x})$

Problem Setup

Data:

$\mathbf{x_1}, \ldots, \mathbf{x_n}$ examples / instances / items

$\mathbf{y_1}, \ldots, \mathbf{y_n}$ labels

Desired:

$f(\mathbf{x})$

y's are vectors. why/how?

Problem Setup

Data:

$\mathbf{x_1}, \ldots, \mathbf{x_n}$ examples / instances / items

$\mathbf{y_1}, \ldots, \mathbf{y_n}$ labels

Desired:

$f_\theta(\mathbf{x})$

Problem Setup

Data:

$\mathbf{x_1}, \ldots, \mathbf{x_n}$ examples / instances / items

$\mathbf{y_1}, \ldots, \mathbf{y_n}$ labels

Desired:

$$f_\theta(\mathbf{x}) = \mathbf{wx} + b \qquad \theta = \mathbf{w}, b$$

learning: finding good values for θ

Problem Setup

Data:

$\mathbf{x_1}, \ldots, \mathbf{x_n}$ examples / instances / items

$\mathbf{y_1}, \ldots, \mathbf{y_n}$ labels

Desired:

$$f_\theta(\mathbf{x}) = \mathbf{wx} + b \qquad \theta = \mathbf{w}, b$$

learning: finding **good** values for θ?

Objective Function

$$\mathbf{Y} = \mathbf{y_1}, \ldots, \mathbf{y_n}$$

$$\hat{\mathbf{Y}}_\theta = f_\theta(\mathbf{x_1}), \ldots, f_\theta(\mathbf{x_n})$$

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_\theta)$$

Objective Function

$$\mathbf{Y} = \mathbf{y_1}, \ldots, \mathbf{y_n}$$

$$\hat{\mathbf{Y}}_\theta = f_\theta(\mathbf{x_1}), \ldots, f_\theta(\mathbf{x_n})$$

Ideally:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_\theta) \propto \sum_{i=1}^{n} \ell(\mathbf{y_i}, f_\theta(\mathbf{x_i}))$$

the objective decomposes over local losses

Objective Function

$$\mathbf{Y} = \mathbf{y_1}, \ldots, \mathbf{y_n}$$

$$\hat{\mathbf{Y}}_\theta = f_\theta(\mathbf{x_1}), \ldots, f_\theta(\mathbf{x_n})$$

Ideally:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_\theta) \propto \sum_{i=1}^{n} \ell(\mathbf{y_i}, f_\theta(\mathbf{x_i}))$$

the objective decomposes over local losses

Learning: $\arg\min_\theta \mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_\theta)$

Loss Functions $\ell(\mathbf{y}, \hat{\mathbf{y}})$

- 0-1 Loss

$$\begin{cases} 0 & \mathbf{y} = \hat{\mathbf{y}} \\ 1 & \text{otherwise} \end{cases}$$

Loss Functions $\ell(\mathbf{y}, \hat{\mathbf{y}})$

- Hinge (margin) loss

$$t = \arg\max_i \mathbf{y}_{[i]}$$

$$p = \arg\max_{i \neq t} \hat{\mathbf{y}}_{[i]}$$

$$\ell_{\text{hinge}} = \max(0, 1 - (\hat{\mathbf{y}}_{[t]} - \hat{\mathbf{y}}_{[p]}))$$

correct score should be
higher than incorrect score
by at least 1

Loss Functions $\ell(\mathbf{y}, \hat{\mathbf{y}})$

- log-loss (also called cross-entropy)

  if the output $\hat{\mathbf{y}}$ is a probability vector:

$$\hat{\mathbf{y}}_{[k]} = P(y = k|\mathbf{x}) \qquad \sum_k \hat{\mathbf{y}}_{[k]} = 1$$

$$\hat{\mathbf{y}}_{[k]} \geq 0$$

$$\ell_{\text{cross-ent}} = -\sum_k \mathbf{y}_{[k]} \log \hat{\mathbf{y}}_{[k]}$$

Loss Functions $\ell(\mathbf{y}, \hat{\mathbf{y}})$

- log-loss (also called cross-entropy)

  if the output $\hat{\mathbf{y}}$ is a probability vector:

$$\hat{\mathbf{y}}_{[k]} = P(y = k|\mathbf{x}) \qquad \sum_k \hat{\mathbf{y}}_{[k]} = 1$$

$$\hat{\mathbf{y}}_{[k]} \geq 0$$

$$\ell_{\text{cross-ent}} = -\sum_k \mathbf{y}_{[k]} \log \hat{\mathbf{y}}_{[k]}$$

for "hard" (0 or 1) labels: $\ell_{\text{cross-ent}} = -\log \hat{\mathbf{y}}_{[t]}$

Loss Functions $\ell(\mathbf{y}, \hat{\mathbf{y}})$

- log-loss (also called cross-entropy)

if the output $\hat{\mathbf{y}}$ is a **probability** vector:

$$\hat{\mathbf{y}}_{[k]} = P(y = k|\mathbf{x}) \qquad \sum_k \hat{\mathbf{y}}_{[k]} = 1$$

$$\hat{\mathbf{y}}_{[k]} \geq 0$$

$$\ell_{\text{cross-ent}} = -\sum_k \mathbf{y}_{[k]} \log \hat{\mathbf{y}}_{[k]}$$

for "hard" (0 or 1) labels: $\ell_{\text{cross-ent}} = -\log \hat{\mathbf{y}}_{[t]}$

probability output

- the softmax function:

$$\text{softmax}(\mathbf{v})_{[i]} = \frac{e^{\mathbf{v}_{[i]}}}{\sum_{i'} e^{\mathbf{v}_{[i']}}}$$

the exponentiation makes positive.
the normalization make it sum to 1.

probability output

- the softmax function:

$$\text{softmax}(\mathbf{v})_{[i]} = \frac{e^{\mathbf{v}_{[i]}}}{\sum_{i'} e^{\mathbf{v}_{[i']}}}$$

Log-linear model
(aka "logistic regression")

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{x}\mathbf{W} + \mathbf{b})$$

$$\hat{\mathbf{y}}_{[i]} = \frac{e^{(\mathbf{x}\mathbf{W}+\mathbf{b})_{[i]}}}{\sum_i e^{(\mathbf{x}\mathbf{W}+\mathbf{b})_{[i]}}}$$

training as optimization

$$\mathbf{Y} = \mathbf{y_1}, \ldots, \mathbf{y_n}$$

$$\hat{\mathbf{Y}}_\theta = f_\theta(\mathbf{x_1}), \ldots, f_\theta(\mathbf{x_n})$$

Ideally:

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_\theta) \propto \sum_{i=1}^{n} \ell(\mathbf{y_i}, f_\theta(\mathbf{x_i}))$$

the objective decomposes over local losses

Learning: $\arg\min_\theta \mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}_\theta)$

training as optimization

- via gradient descent
- via stochastic gradient descent
- via batched stochastic gradient descent

When training, consider

- Regularization
- L2, L1
- Dropout

Multilayer Networks

Linear Classifier

Binary:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

$$\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

Multi class:

$$f(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

$$\arg\max_i \text{softmax}(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})_{[i]}$$

Non-Linear Classifier

$$f_\theta(\mathbf{x}) \ = \mathbf{w} g(\mathbf{W}' \cdot \mathbf{x} + \mathbf{b}') + b$$

Non-Linear Classifier

Multi-layer Perceptron (MLP):

$$f_\theta(\mathbf{x}) = \ \mathrm{NN}_{\mathrm{MLP2}}(\mathbf{x}) = \mathbf{y}$$

$$\mathbf{h^1} = g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1})$$

$$\mathbf{h^2} = g^2(\mathbf{h^1}\mathbf{W^2} + \mathbf{b^2})$$

$$\mathbf{y} = \mathbf{h^2}\mathbf{W^3}$$

Non-Linear Classifier

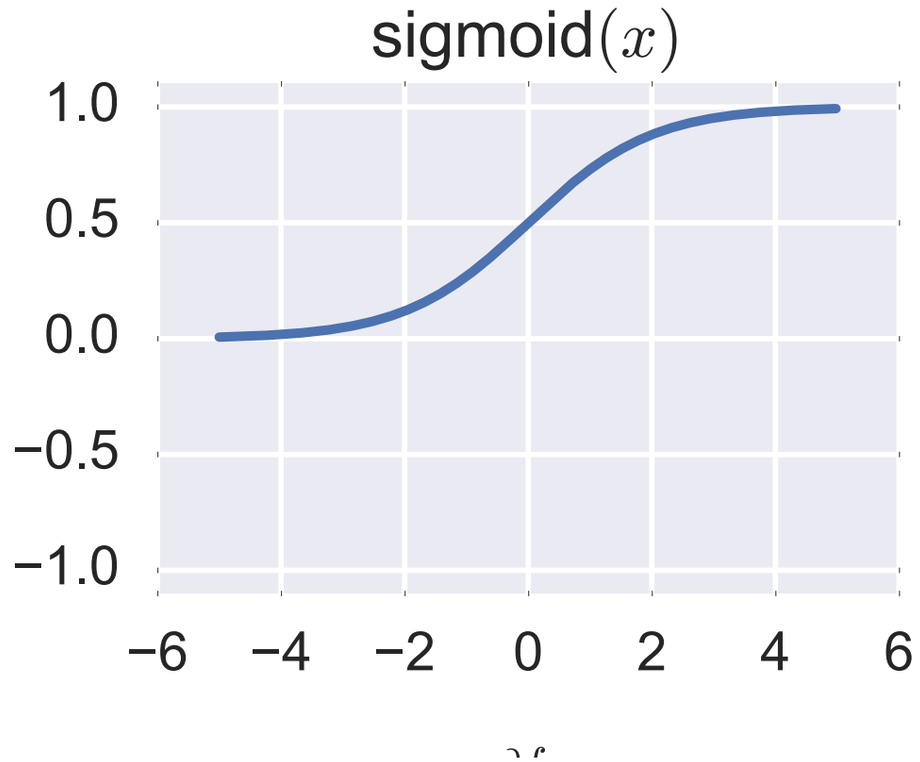
Multi-layer Perceptron (MLP):

$$
\begin{aligned}
f_\theta(\mathbf{x}) = \ & \mathrm{NN}_{\mathrm{MLP2}}(\mathbf{x}) = \mathbf{y} \\
\mathbf{h^1} = & g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1}) \\
\mathbf{h^2} = & g^2(\mathbf{h^1}\mathbf{W^2} + \mathbf{b^2}) \\
\mathbf{y} = & \mathbf{h^2}\mathbf{W^3}
\end{aligned}
$$

Common Non-linearities

Sigmoid

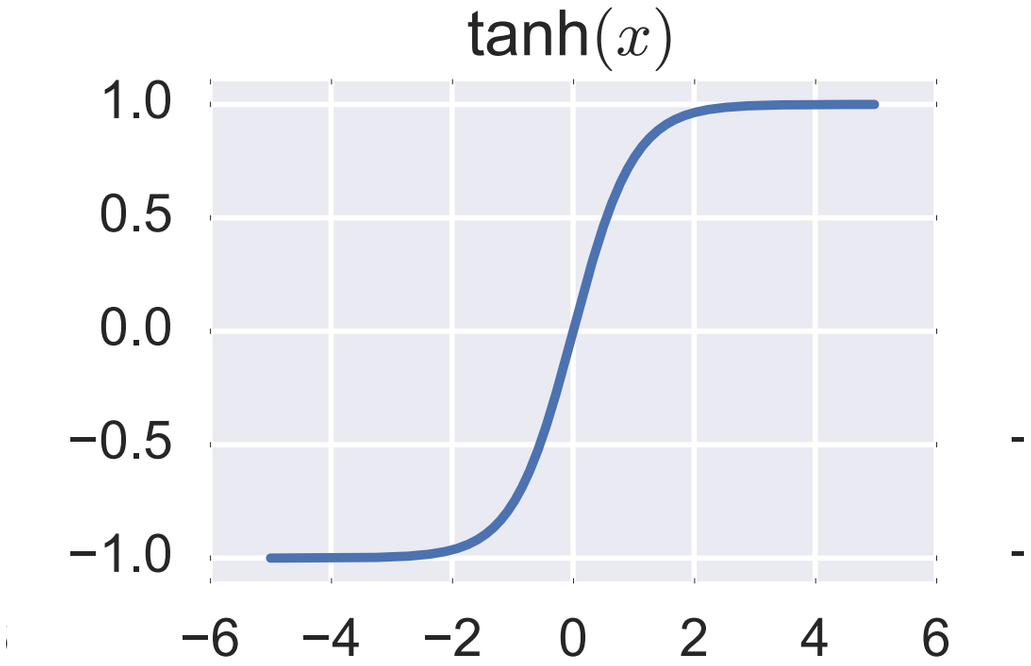
$$\sigma(x) = 1/(1 + e^{-x})$$



Common Non-linearities

tanh

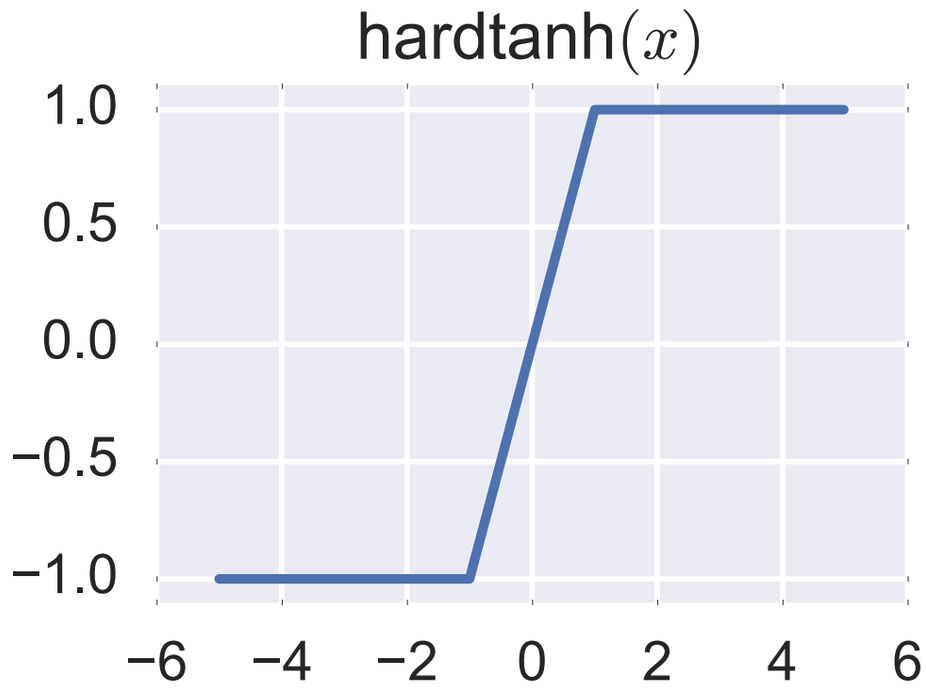
$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$



Common Non-linearities

hard-tanh

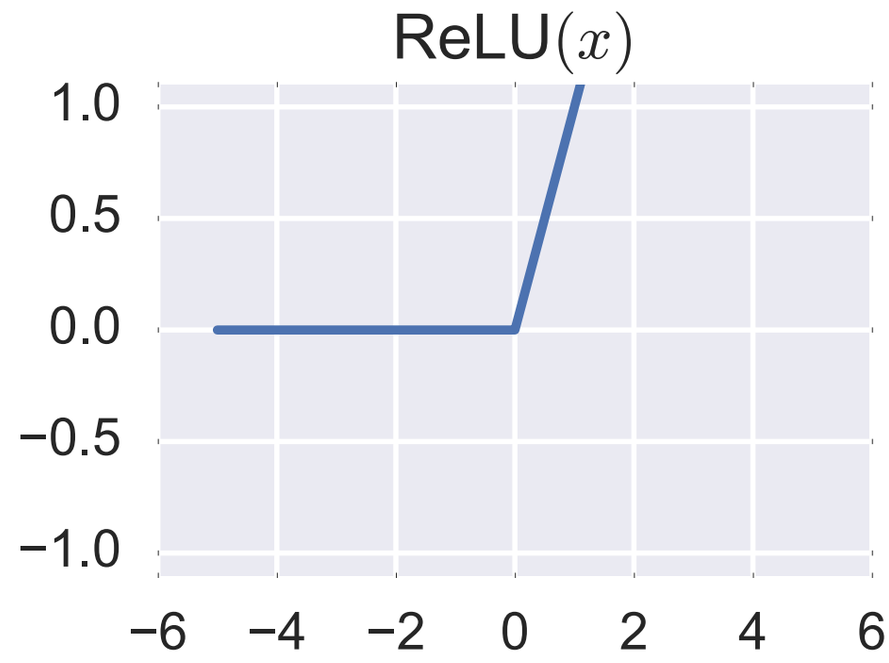
$$\text{hardtanh}(x) = \begin{cases} -1 & x < -1 \\ 1 & x > 1 \\ x & \text{otherwise} \end{cases}$$



Common Non-linearities

ReLU (rectifier, rectified linear unit)

$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} 0 & x < 0 \\ x & \text{otherwise} \end{cases}$$



Which non-linearity to use?

- No good rules.
- Use sigmoid when you want a 0-1 behavior. Otherwise prefer not to use it.
- tanh and ReLU work well.
- There are also fancier ones (i.e. ELU, GELU, ...)

Representation Power

One hidden-layer is enough!

- For every borel-measurable function, there is a multi-layer perceptron with one hidden layer that can approximate it to any desired epsilon.

So why do we need deeper networks?

Benefit of Depth

- Deeper networks can express functions **using narrower layers.**
- **Depth-separation:**
  a line of research that shows functions that can be approximated well with depth *n* networks with linear width at each layer, but require exponential width in n for bounded depth networks.

Nice proof
of the benefit of depth

A Simple Geometric Proof for the Benefit of Depth in ReLU Networks

Asaf Amrami[1,2] **& Yoav Goldberg**[1,2]
[1] Bar Ilan University
[2] Allen Institute for Artificial Intelligence

How many layers to use? And how wide should they be?

- No hard and fast rules.
- In vision, we see that "deeper is better".
  - Not always the case in text / sequences (though with transformers, we may be starting to see this).
- Can think of each layer as **transforming** the previous layer (remember the xor example).
- Narrower layers "compress" the information in the previous layer. Wider layers introduce redundancies.

Summarize

- Linear separation.
- Formal learning problem setup.
- Loss functions.
- Learning as optimization.