I2DL :: Notation

Note: The lecture uses many examples from maths, statistics and machine learning. Therefore notation may overlap, but the context should make clear how to uncerstand the notation. If notation is unclear nevertheless, please contact the instructors.

Data

$\mathcal{X}$: **feature space** / input space
with dimensions of $\mathcal{X}$ depending on the type of NN

$\mathcal{Y}$: **target space**
e.g.: $\mathcal{Y} = \mathbb{R}$ for regression, $\mathcal{Y} = \{0, 1\}$ or $\mathcal{Y} = \{-1, +1\}$ for binary classification, $\mathcal{Y} = \{1, \dots, g\}$ for multi-class classification with g classes

$\mathbf{x} \in \mathcal{X}$: **feature vector** / covariate vector

$y \in \mathcal{Y}$: **target variable** / output variable
Concrete samples are called labels

$\left(\mathbf{x}^{(i)}, y^{(i)}\right) \in \mathcal{X} \times \mathcal{Y}$: i -th **observation** / sample / instance / example

$\mathcal{D} = \left(\left(\mathbf{x}^{(1)}, y^{(1)}\right), \dots, \left(\mathbf{x}^{(n)}, y^{(n)}\right)\right) \in \mathbb{D}_n$: **data set** of size n. An n-tuple, a family indexed by $\{1, \dots, n\}$. We use $\mathcal{D}_n$ to emphasize its size.

$\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}} \subseteq \mathcal{D}$: **data sets for training and testing**
Often: $\mathcal{D} = \mathcal{D}_{\text{train}} \dot{\cup} \mathcal{D}_{\text{test}}$

$\mathcal{D}_{subtrain}, \mathcal{D}_{val} \subseteq \mathcal{D}_{\text{train}}$: **data sets in the context of early stopping**
Note: Sophisticated forms also apply cross-validation

Loss, Risk and ERM

$L : \mathcal{Y} \times \mathbb{R}^g \to \mathbb{R}_0^+$: **loss function**: Quantifies "quality" $L(y, f(\mathbf{x}))$ of prediction $f(\mathbf{x})$.

(Theoretical) risk: $\mathcal{R} : \mathcal{H} \to \mathbb{R}$, $\mathcal{R}(f) = \mathbb{E}_{((\mathbf{x},y)\sim\mathbb{P}_{xy})}[L(y, f(\mathbf{x}))]$

Empirical risk: $\mathcal{R}_{\text{emp}} : \mathcal{H} \to \mathbb{R}$, $\mathcal{R}_{\text{emp}} : \Theta \to \mathbb{R}$; $\mathcal{R}_{\text{emp}}(\boldsymbol{\theta}) = \sum_{i=1}^{n} L\left(y^{(i)}, f\left(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}\right)\right)$

Empirical risk minimization (ERM): $\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$

Regularized risk: $\mathcal{R}_{\text{reg}} : \mathcal{H} \to \mathbb{R}, \mathcal{R}_{\text{reg}}(f) = \mathcal{R}_{\text{emp}}(f) + \lambda \cdot J(f)$ with regularizer $J(f)$, complexity control parameter $\lambda > 0$ (analogous for $\boldsymbol{\theta}$).

Gradient based Optimization

$\hat{\boldsymbol{\theta}}^{[t]}$: t-th step of an optimizer
$\alpha \in \mathbb{R}_+$: **step-size / learning rate**
$\mathbf{d} \in \mathbb{R}^d$: descent direction in $\hat{\boldsymbol{\theta}}$
$J_1, ..., J_K$: mini-batches of fixed size m with $k \in \{1, ..., K\}$

(Feedforward) Neural Network

$\mathcal{X} \subseteq \mathbb{R}^p$: p-dimensional **feature space** / input space

(F)NN: $f(\mathbf{x}) = \tau \circ \phi \circ \sigma^{(l)} \circ \phi^{(l)} \circ \sigma^{(l-1)} \circ \phi^{(l-1)} \circ \ldots \circ \sigma^{(1)} \circ \phi^{(1)}$, with
$\phi^{(i)}$: affine transformation in hidden layer i
$\sigma^{(i)}$: activation function in hidden layer i
ϕ: affine function in output layer
τ: activation function in output layer

Hidden layer i, $i \in \{1 \dots l\}$ with $m^{(i)}$ = nmb. of neurons in i
$\mathbf{W}^{(i)}$: weight matrix $\left(m^{(i-1)} \times m^{(i)}\right)$
If $\mathbf{x}^{(p\times 1)}$:
$\mathbf{b}^{(i)}$: bias
$\mathbf{z}^{(i)} = \mathbf{z}_{out}^{(i)} = \sigma^{(i)}(\phi^{(i)}) = \sigma^{(i)}(\mathbf{z}_{in}^{(i)}) = \sigma^{(i)}(\mathbf{W}^{(i)T}\mathbf{z}^{(i-1)} + \mathbf{b}^{(i)})$: activation
If $\mathbf{X}^{(n\times p)}$:
$\mathbf{B}^{(i)}$: bias $(n \times m)$
$\mathbf{Z}^{(i)} = \mathbf{Z}_{out}^{(i)} = \sigma^{(i)}(\phi^{(i)}) = \sigma^{(i)}(\mathbf{Z}_{in}^{(i)}) = \sigma^{(i)}(\mathbf{Z}^{(i-1)}\mathbf{W}^{(i)} + \mathbf{B}^{(i)})$: activation

Neuron j, $j \in \{1 \dots m^{(i)}\}$:
$\mathbf{W}_j^{(i)}$: column j $(m^{(i-1)} \times 1)$ of weight matrix
If $\mathbf{x}^{(p\times 1)}$:
$b_j^{(i)}$: bias
$\mathbf{z}_j^{(i)} = \mathbf{z}_{j,out}^{(i)} = \sigma^{(i)}(\phi_j^{(i)}) = \sigma^{(i)}(\mathbf{z}_{j,in}^{(i)}) = \sigma^{(i)}(\mathbf{W}_j^{(i)T}\mathbf{z}^{(i-1)} + b_j^{(i)})$:

Convolutional Neural Networks

Disclaimer: The given notation and logic only applies to CNNs using both quardratic images as input and quadratic kernels/filters!

$\mathcal{X} \subseteq \mathbb{R}^{H\times B\times C}$: **feature space** representing images with
H: height of image (in pixels),
B: breadth/width of image (in pixels),
C: number of channels/depth of image

2D Covolution for quadratic input and quadratic filter per channel c:
$\mathbf{I} \in \mathbb{R}^{i\times i}$: quadratic Input $(\to h == b)$
$\mathbf{W} \in \mathbb{R}^{k\times k}$: Filter/kernel
$\mathbf{p} \in \mathbb{N}$: Padding
$\mathbf{stride} \in \mathbb{N}$: Stride
$\mathbf{S} \in \mathbb{R}^{o\times o}$: Output matrix with $o = \frac{i-k+2\cdot p}{stride} + 1$

Recurrent Neural Networks

Recurrent Neural Networks
t: current step in RNN
$\mathbf{W}$: weight matrix
$\mathbf{V}$: additional set of weights from the hidden neurons at time-step t to the hidden neurons at time-step $t+1$
$\mathbf{x}^{[t]}$: Input at current step t
$\mathbf{z}^{[t]}$: state at t/vector of short-term memory

$$\mathbf{z}^{[t]} = \sigma(\mathbf{V}^\top \mathbf{z}^{[t-1]} + \mathbf{W}^\top \mathbf{x}^{[t]} + \mathbf{b})$$

Long short-term memory (LSTM)
$\mathbf{z}^{[t]}$: state at t

$$\mathbf{z}^{[t]} = \mathbf{o}^{[t]} \odot \tanh(\mathbf{s}^{[t]})$$

$\mathbf{o}^{[t]}$: output gate
$\mathbf{o}^{[t]} = \sigma(\mathbf{b}_o + \mathbf{V}_o^\top \mathbf{z}^{[t-1]} + \mathbf{W}_o^\top \mathbf{x}^{[t]})$
$\mathbf{s}^{[t]}$: cell state
$\mathbf{s}^{[t]} = \mathbf{e}^{[t]} \odot \mathbf{s}^{[t-1]} + \mathbf{i}^{[t]} \odot \tilde{\mathbf{s}}^{[t]}$
$\tilde{\mathbf{s}}^{[t]}$: new information
$\tilde{\mathbf{s}}^{[t]} = \tanh(\mathbf{b} + \mathbf{V}^\top \mathbf{z}^{[t-1]} + \mathbf{W}^\top \mathbf{x}^{[t]}) \in [-1, 1]$
$\mathbf{i}^{[t]}$: input gate
$\mathbf{i}^{[t]} = \sigma(\mathbf{b}_i + \mathbf{V}_i^\top \mathbf{z}^{[t-1]} + \mathbf{W}_i^\top \mathbf{x}^{[t]}) \in [0, 1]$
$\mathbf{e}^{[t]}$: forget gate
$\mathbf{e}^{[t]} = \sigma(\mathbf{b}_e + \mathbf{V}_e^\top \mathbf{z}^{[t-1]} + \mathbf{W}_e^\top \mathbf{x}^{[t]})$

Gated Recurrent Units (GRU)
$\mathbf{z}^{[t]}$: state at t

$$\mathbf{z}^{[t]} = \mathbf{u}^{[t]} \odot \mathbf{z}^{[t-1]} + (1 - \mathbf{u}^{[t]}) \odot \tilde{\mathbf{z}}^{[t]}$$

$\mathbf{u}^{[t]}$: update gate
$\mathbf{u}^{[t]} = \sigma(\mathbf{W}_u^\top \mathbf{x}^{[t]} + \mathbf{V}_u^\top \mathbf{z}^{[t-1]} + \mathbf{b}_u)$
$\tilde{\mathbf{z}}^{[t]}$: new candidate state
$\tilde{\mathbf{z}}^{[t]} = \tanh(\mathbf{W}_z^\top \mathbf{x}^{[t]} + \mathbf{V}_z^\top \left(\mathbf{r}^{[t]} \odot \mathbf{z}^{[t-1]}\right) + \mathbf{b}_z)$
$\mathbf{r}^{[t]}$: reset gate
$\mathbf{r}^{[t]} = \sigma(\mathbf{W}_r^\top \mathbf{x}^{[t]} + \mathbf{V}_r^\top \mathbf{z}^{[t-1]} + \mathbf{b}_r)$