# Artificial Neural Networks (Gerstner). Exercises for week 6

**Regularization and Tricks of the Trade**

## Exercise 1. Cross-validation

Assume $K$ variants of a model, with model $k$ having test error $E_k = E_0 + \epsilon_k$, where $\epsilon_k$ has mean $\mathbb{E}[\epsilon_k] = 0$, auto-variance $\mathbb{E}\left[\epsilon_k^2\right] = v$ and co-variance $\mathbb{E}[\epsilon_k \epsilon_n] = c$.

a. What is the expected value of the *test error*, i.e. the expected test error of the model obtained by averaging over all variants?

b. What is the variance of the average test error? Does the number $K$ of variants play a role?

c. Consider implementing $K$-fold cross validation. If every $E_k$ is the error of one of the folds, how do variance $v$ and correlation $\rho = \frac{c}{v}$ change with respect to $K$? How does the variance of the average test error behave as $K$ varies, assuming that we have large number of sample points?

## Exercise 2. Dropout (from exam 2019)

We have a deep network of $2n$ hidden layers ($n > 2$) of neurons with sharp threshold functions $g(a) = 1$ for $a > 0$ and zero otherwise. After training with dropout, somewhere in hidden layer $n$, we have a hidden neuron $i$ which receives input from 4 hidden neurons in layer $n-1$. All weights onto neuron $i$ are equal to one and the threshold of neuron $i$ is 2.7.

Each of the four hidden neurons $j$ in layer $n-1$ receives input from the same 2 neurons in layer $n-2$. The weight vectors and thresholds of the four neurons in layer $n-1$ are:

j=1 (1,0) and threshold 0
j=2 (1,0) and threshold 0.5
j=3 (1,1) and threshold 1
j=4 (1,-1)and threshold 1

a. Qualitatively sketch the two-dimensional space representing the activity of the 2 neurons in layer $n-2$ and indicate the region (by shading it with crosses x x x) in which neuron $i$ responds positively.

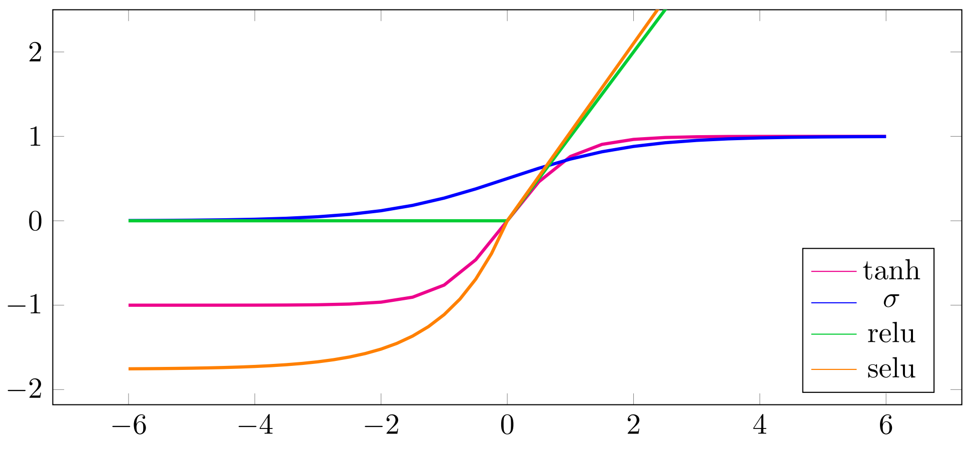
b. Dropout: Remove neurons $j = 1$ and $j = 4$ in layer $n-1$, rescale the weights appropriately, and sketch the input space where neuron $i$ responds positively (by shading it with crosses x x x ).

c. Dropout: Remove neurons $j = 2$ and $j = 3$ in layer $n-1$, rescale the weights appropriately, and sketch the input space where neuron $i$ responds positively (by shading it with crosses x x x ).

d. Your friend Adam claims: 'Dropout might be a useful trick, but nobody understands how it works'. Your friend Berthilde claims 'Dropout is good for generalization and easy to understand'. Comment on your results (think also of the other 4 combinations of dropping out two neurons) and relate your results to the claims of your friends.

## Exercise 3. Different activation functions

The choice of the non-linearity function $g(x)$ can have a significant impact on learning speed and final performance. Which non-linearity is best, is still an active research question; the favorite non-linearity in the last century was probably the hyperbolic tangent $\tanh(x)$; since 2010, the rectified linear unit $\text{relu}(x) = \max(0, x)$ is highly popular and there is a fair chance that the new favorite will be the scaled exponential linear unit $\text{selu}(x) = \lambda x$ if $x > 0$ and $\text{selu}(x) = \alpha(\exp(x) - 1)$ otherwise, with $\lambda \approx 1.0507$ and $\alpha \approx 1.75814$. Currently, it seems that the key concepts to discuss the different non-linearities are, first, *linearity problem*, second the *vanishing gradient problem* and, third, the *bias shift problem.*



a. Linearity problem

  (i) Show that a multi-layer neural network with linear activation function $g(x) = x$ is equivalent to a single layer linear network. Hint: the product of two matrices is again a matrix.

  (ii) Assume that in each layer the inputs follow a Normal distribution with mean zero and small variance, i.e. $\sigma^2 \ll 1$. For which of the activation functions $\sigma(x) = 1/(1 + \exp(-x))$, $\tanh(x)$, $\text{relu}(x)$ and $\text{selu}(x)$ is a deep network basically equivalent to a linear network for this input distribution? Hint: Consider the case $\sigma^2 \to 0$ using a Taylor expansion around 0.

b. Vanishing gradient problem

  (i) Assume now the inputs are such that they also fall into the non-linear regimes. For simplicity we assume that in each layer the activations are $a_1 = -10, a_2 = -5, a_3 = -1, a_4 = 1, a_5 = 5, a_6 = 10$. Without a calculator, determine the fraction of values close to zero of $g(a_i)$ and $g'(a_i)$ for all $i$ and $g = \sigma, \tanh, \text{relu}, \text{selu}$. For example, for tanh none of the values $\tanh(-10), \tanh(-5), \ldots, \tanh(10)$ is close to zero but $4/6 = 2/3$ of the values of $\tanh' = 1 - \tanh^2$ are close to zero.

  (ii) The update of a weight $w_{ij}$ is proportional to $g'(a_i) \cdot g(a_j)$. Determine the fraction of $g'(a_i) \cdot g(a_j)$ that are close to zero considering all combinations of $a_i$ and $a_j$ and all activations $g = \sigma, \tanh, \text{relu}, \text{selu}$.

  (iii) The $\delta$'s in backpropagation are in each layer multiplied with $g'$. Consider backpropagation through 3 layers, i.e. terms like $g'(a_i)g'(a_j)g'(a_k)$. Determine the fraction of such terms that are close to zero for $g = \sigma, \tanh, \text{relu}, \text{selu}$.

c. Bias shift problem

Consider a simple classification task. The data exist in $\mathcal{R}^N$. Data points from $C_0$ (with target $t = 0$) are uniformly distributed in each dimension such that $x_i \in [1, 2]$ for $i = 1 \ldots N$. Data points from $C_1$ (with target $t = 1$) are uniformly distributed in each dimension such that $x_i \in [3, 4]$ for $i = 1 \ldots N$. We want to learn to classify points using a logistic sigmoid unit trained with the cross–entropy loss; from last week, this results in the weight update rule

$$\Delta w_i = \eta \cdot (t - y) \cdot x_i$$

where $y = \sigma\left(\sum_i^N w_i x_i\right)$.

Points are presented one at a time (i.e. stochastic gradient descent).

  (i) Assume we start with all weights $w_i = 0$ and present the point $\mathbf{x}^a$ from $C_0$, update the weights, then present $\mathbf{x}^b$. Give the drive $a = \sum_i^N w_i x_i^b$ of the output unit in response to $\mathbf{x}^b$, in terms of $\eta$, $\mathbf{x}^a$ and $\mathbf{x}^b$. Note: we do not yet need to specify which class $\mathbf{x}^b$ belongs to.

(ii) In general, we can encounter oscillations in stochastic gradient descent if a single training example strongly affects the network output – for instance, if it results in the same network output for any possible input.

We assume that if $a < -5$, $y \approx 0$, and if $a > 5$, $y \approx 1$. Under what conditions will the network output $y$ be the same for all possible inputs $\mathbf{x}^b$ after the first training step? Can we choose a small enough $\eta$ to prevent this, independent of $N$? What if we had chosen $\mathbf{x}^a$ from $C_1$ instead?

(iii) A common input normalization technique to to remove the mean from the dataset, such that $E[x_i] = 0$ across all dimensions $x_i$. Assume that each data point has an equal probability of coming from either $C_0$ and $C_1$. What are the new data ranges for $C_0$ and $C_1$ after removing the mean? Repeating step (ii), do we get the same result?

(iv) Consider a deep network where each hidden layer uses one of the following activation functions: tanh, $\sigma$, relu, or selu. Given what we've seen above, can you suggest one of the activation functions? Note that one layer's output is another layer's input.

d. Summarize your results by ranking the different activation functions for each of the problems discussed in this exercise.

| | linearity problem | vanishing gradient problem | bias shift problem |
|---|---|---|---|
| tanh | | | |
| $\sigma$ | | | |
| relu | | | |
| selu | | | |

**Exercise 4. Normalization of activations across multiple layers**

In class we have seen, that by an appropriate normalization of the input patterns (for each input component: zero mean, unit standard deviation) combined with a Gaussian distribution of input weights ($\langle w_{ij}^{(1)} \rangle = 0$ and $\langle [w_{ij}^{(1)}]^2 \rangle = 1/N$) we can ensure that the activation variable of neurons in the first layer has mean $\langle a_i^{(1)} \rangle = 0$ and variance $\langle [a_i^{(1)}]^2 \rangle = 1$. In the following we assume that the distribution of activations in layer 1 is standard Gaussian, i.e. $a_i^{(1)} \sim N(0,1)$.

The aim of the exercise is to go by induction from layer $n$ to layer $n+1$. We start in layer 1.

Assume that neuron $j$ in layer 1 has a rectified linear activation function, i.e., $x_j^{(1)} = [a_j^{(1)}]_+$.

a. What is the mean $\langle x_j^{(1)} \rangle$?

b. Assume that the weights in layer 2 are initialized with zero mean and variance $\langle [w_{kj}^{(2)}]^2 \rangle = [c^2]/N_1$ where $N_1$ is the number of hidden neurons in the first layer.

What is the mean activation $\langle \tilde{a}_k^{(2)} \rangle$ in layer 2? Here $\tilde{a}_k^{(2)} = \sum_{j=1}^{N_1} w_{kj}^{(2)} x_j^{(1)}$. The total activation of neuron $k$ in layer 2 is $a_k^{(2)} = \tilde{a}_k^{(2)} - \theta_k$.

What value should you choose for the threshold $\theta_k$ in layer 2, so that $\langle a_k^{(2)} \rangle = 0$ in layer 2?

c. Assume that you found a threshold so that $\langle a_k^{(2)} \rangle = 0$. Calculate the variance $\langle [a_k^{(2)}]^2 \rangle$ as a function of the constant $c$.

d. Choose $c$ such that the variance is one.

e. Can you now go from layer 2 to layer 3?