Artificial Neural Networks (Gerstner). Solutions for week 4

Regularization and Tricks of the Trade

Exercise 1. Softmax function for multi-class problems.

We have a network with K output units indexed by $1 \leq k \leq K$. We want that output unit k gives the posterior probability for class k

$$\hat{y}_k = P(C_k|\boldsymbol{x}) \tag{1}$$

Show that we arrive at the softmax function if we set the joint probabilities as $p(\boldsymbol{x}, C_k) = \frac{1}{Z} e^{a_k}$, where Z is a normalization constant.

Hints: Proceed analogously to the 'derivation' of the sigmoidal unit in class last week. Remember that the softmax function is

$$\hat{y}_k = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{2}$$

Solution:

$$\begin{aligned} P(C_k|\boldsymbol{x}) &= \frac{P(\boldsymbol{x}|C_k)P(C_k)}{\sum_j P(\boldsymbol{x}, C_j)} && (3) \\ &= \frac{P(\boldsymbol{x}, C_k)}{\sum_j P(\boldsymbol{x}, C_j)} && (4) \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} && (5) \end{aligned}$$

Exercise 2. Bagging and dropout

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

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

b. What is the variance of the bagging test error?

c. For dropout: do you expect the co-variance c to be close to zero, between zero and v, close to v or larger than v? Justify your answer with one sentence.

Solution:

a.

$$\begin{aligned} \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K} E_k\right] &= \frac{1}{K}\left(\mathbb{E}\left[\sum_{k=1}^{K} E_0\right] + \mathbb{E}\left[\sum_{k=1}^{K} \epsilon_k\right]\right) && (6) \\ &= E_0 && (7) \end{aligned}$$

b.

$$
\begin{aligned}
&\mathbb{E}\left[\left(\frac{1}{K}\sum_{k=1}^{K} E_k - \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K} E_k\right]\right)^2\right] && (8)\\
=\;& \frac{1}{K^2}\mathbb{E}\left[\sum E_k^2 + 2\sum_{i\neq j} E_i E_j - 2K\sum E_k E_0 + K^2 E_0^2\right] && (9)\\
=\;& \frac{1}{K^2}\left(\mathbb{E}\left[\sum (E_0+\epsilon_k)^2\right] + 2\mathbb{E}\left[\sum_{i\neq j}(E_0+\epsilon_i)(E_0+\epsilon_j)\right] - 2K^2E_0^2 + K^2E_0^2\right) && (10)\\
=\;& \frac{1}{K^2}\left(KE_0^2 + Kv + 2\frac{K(K-1)}{2}(E_0^2+c) - K^2E_0^2\right) && (11)\\
=\;& \frac{1}{K}v + \frac{K-1}{K}c && (12)
\end{aligned}
$$

c. If the variants are independent $\mathbb{E}[\epsilon_k\epsilon_n] = 0$, if they are all the same $\mathbb{E}[\epsilon_k\epsilon_n] = \mathbb{E}[\epsilon_k^2]$. Dropout is a random sample of a subset of a bigger network. The size of the subset depends on the dropout rate. Therefore, for small dropout rates, the subsets will be very similar and the co-variance close to v. For larger dropout rate, the co-variance will be smaller.

Exercise 3. Bias and variance of gradient estimators

For training data $(\boldsymbol{x}^1, y^1), \ldots, (\boldsymbol{x}^P, y^P)$ and some loss $E(\boldsymbol{w}) = \frac{1}{P}\sum_\mu \ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)$, the gradient is given by $\nabla E(\boldsymbol{w}) = \frac{1}{P}\sum_\mu \nabla\ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)$, with e.g. $\ell(x,y) = \frac{1}{2}(x-y)^2$.

a. In each step of stochastic gradient descent one sample $(\boldsymbol{x}^\mu, y^\mu)$ of the training data is selected. Show that $\nabla\ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)$ is an unbiased estimator of $\nabla E(\boldsymbol{w})$, if each training sample is selected with equal probability. Hint: An estimator $\hat{\theta}$ of a quantity θ is called unbiased, if its expectation $\mathbb{E}\left[\hat{\theta}\right] = \theta$.

b. Instead of single sample stochastic gradient descent it is common practice to use mini-batches. Show that the mini-batch gradient estimator $\frac{1}{M}\sum_{i=1}^{M}\nabla\ell(f_{\boldsymbol{w}}(\boldsymbol{x}^i), y^i)$, with $1 < M < P$, has lower variance than the single sample estimator, if the samples $(\boldsymbol{x}^i, y^i)$ in each mini-batch are sampled uniformly from the training data.

Solution:

a. Let μ be a random training sample selected uniformly at random from $\{1, ..., P\}$. Then

$$
\begin{aligned}
\mathbb{E}\left[\nabla\ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)\right] &= \frac{1}{P}\sum_\mu \nabla\ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu) && (13)\\
&= \nabla E(\boldsymbol{w}) && (14)
\end{aligned}
$$

b. Let $X_\mu = \nabla\ell(f_{\boldsymbol{w}}(\boldsymbol{x}^\mu), y^\mu)$ be the gradient from a single training sample. The variance of the mini-batch gradient of size M, $\frac{1}{M}\sum_{i=1}^{M} X_i$ for $\{i, ..., M\}$ selected uniformly at random from $\{1, ..., P\}$ is

$$\mathbb{E}\left[\left(\frac{1}{M}\sum_{i=1}^{M} X_i - \mathbb{E}[X]\right)^2 \right] = \tag{15}$$

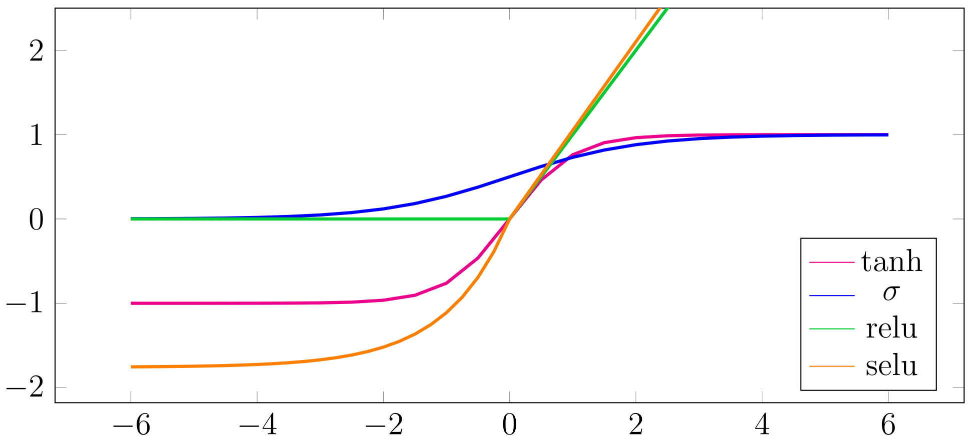
$$= \frac{1}{M^2}\mathbb{E}\left[\sum_{i=1}^{M} X_i^2 + 2\sum_{i\neq j}^{M} X_i X_j + M^2\mathbb{E}[X]^2 - 2M^2\mathbb{E}[X]^2 \right] \tag{16}$$

$$= \frac{1}{M^2}\left(B\mathbb{E}\left[X^2\right] + M(M-1)\mathbb{E}[X]^2 + M^2\mathbb{E}[X]^2 - 2M^2\mathbb{E}[X]^2 \right) \tag{17}$$

$$= \frac{1}{M}\left(\mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 \right) = \frac{1}{M}\mathrm{Var}[X] \tag{18}$$

Exercise 4. 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 $\mathrm{relu}(x) = \max(0, x)$ is highly popular and there is a fair chance that the new favorite will be the scaled exponential linear unit $\mathrm{selu}(x) = \lambda x$ if $x > 0$ and $\mathrm{selu}(x) = \alpha(\exp(x) - 1)$ 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, *using the nonlinearity*, second the *vanishing gradient problem* and, third, the *bias shift problem*.



a. Using the nonlinearity

   (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 $\sigma(x) = 1/(1+\exp(-x))$, $\tanh(x)$, $\mathrm{relu}(x)$ and $\mathrm{selu}(x)$ is a deep network basically equivalent to a linear network for this input distribution?

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 inputs are $x_1 = -10, x_2 = -5, x_3 = -1, x_4 = 1, x_5 = 5, x_6 = 10$. Without a calculator, determine the fraction of values close to zero of $g(x_i)$ and $g'(x_i)$ for all i and $g = \sigma, \tanh, \mathrm{relu}, \mathrm{selu}$. For example, for tanh none of the values $\tanh(-10), \tanh(-5), \dots, \tanh(10)$ 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 $g'(x_i) \cdot g(x_j)$. Determine the fraction of $g'(x_i) \cdot g(x_j)$ that are close to zero considering all combinations of x_i and x_j and all activations $g = \sigma, \tanh, \mathrm{relu}, \mathrm{selu}$.

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

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 $i = 1 \dots N$. Data points from C_1 (with target $t = 1$) are uniformly distributed in each dimension such that $x_i \in [3, 4]$ for $i = 1 \dots N$. 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 η, $\mathbf{x}^a$ 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 η 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 either the hyperbolic tangent or logistic sigmoid nonlinearity. Given what we've seen above, can you suggest one non–linearity or the other between hidden layers? 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.

	linear for $x \sim \mathcal{N}(0, \sigma \ll 1)$	vanishing gradient problem	bias shift problem
tanh			
σ			
relu			
selu			

Solution:

a. (i) For a network with 1 hidden layer and $g(x) = x$ we have

$$\hat{y}_i = g\left(\sum_j w_{ij}^{(2)} g\left(\sum_k w_{jk}^{(1)} x_k\right)\right) \tag{19}$$

$$= \sum_j w_{ij}^{(2)} \sum_k w_{jk}^{(1)} x_k \tag{20}$$

$$= \sum_{j,k} w_{ij}^{(2)} w_{jk}^{(1)} x_k \tag{21}$$

$$= \sum_k w_{ik} x_k \,, \tag{22}$$

where $w_{ik} = \sum_j w_{ij}^{(2)} w_{jk}^{(1)}$ is the product of the two matrices.
The same argument applies to networks with more then 1 hidden layer.

(ii) The relu and selu activation function have a kink at 0; they are therefore not linear for normally distributed input with mean 0. Using $\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x)$ and $\tanh(0) = 0$, the Taylor series of tanh around 0 is $\tanh(x) = 0+1\cdot x+0\cdot x^2+\mathcal{O}(x^3) = x + \mathcal{O}(x^3)$, i.e. the tanh is basically linear for normally distributed inputs with $\sigma^2 \ll 1$. With $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x)$ and $\sigma(x) = \frac{1}{2}$, the Taylor series of σ around 0 is $\sigma(x) = \frac{1}{2}+\frac{1}{4}x+0\cdot x^2+\mathcal{O}(x^3) = \frac{1}{2}+\frac{1}{4}x+\mathcal{O}(x^3)$, i.e. σ is an affine function around 0, but even with the offset $\frac{1}{2}$ the relevant part that depends on x has the same form as in Equation 22 and thus also the σ activation function leads to basically a linear network for the given input distribution.

b. (i)

g	fraction g close to 0	fraction g' close to 0
tanh	0	2/3
σ	1/2	2/3
relu	1/2	1/2
selu	0	1/3

(ii) We determine the fraction of terms close to zero by computing first those that are not close to zero, which is given by the product of the fraction of terms not close to zero of the previous table.

g	fraction $g \cdot g'$ close to 0
tanh	$1 - 1/3 = 2/3$
σ	$1 - 1/2 \cdot 1/3 = 5/6$
relu	$1 - 1/2 \cdot 1/2 = 3/4$
selu	$1 - 2/3 = 1/3$

(iii) We proceed as in the previous exercise.

g	fraction $g' \cdot g' \cdot g'$ close to 0
tanh	$1 - 1/3^3 = 26/27 \approx 0.96$
σ	$1 - 1/3^3 = 26/27 \approx 0.96$
relu	$1 - 1/2^3 = 7/8 = 0.875$
selu	$1 - (2/3)^3 = 19/27 \approx 0.70$

c. (i) With all weights equal to 0, the initial output y will be 0.5 for all possible inputs.

Therefore,

$$\begin{aligned}\Delta w_i &= \eta \cdot (t - y) \cdot x_i^a \\ &= \eta \cdot (0 - 0.5) \cdot x_i^a \\ &= -0.5\eta \cdot x_i^a\end{aligned}$$

and

$$\begin{aligned}a &= \sum_i^{N+1} w_i x_i^b \\ &= \sum_i^{N+1} (-0.5\eta \cdot x_i^a) \cdot x_i^b \\ &= -0.5\eta \sum_i^{N+1} x_i^a x_i^b\end{aligned}$$

where we have absorbed the bias by taking $x_{N+1}^a x_{N+1}^b = (-1)(-1) = 1$.

(ii) From the equation above, we note that $x_i^a x_i^b$ is always positive, so a will be negative for any value of x_i^b. We are interested in the case where $a > -5$, so the output unit does not saturate. The absolute value $|a|$ is minimized for $\mathbf{x}^a = [1, 1, 1, \dots 1]$ and $\mathbf{x}^b = [1, 1, 1, \dots 1]$ (in which case $\mathbf{x}^b$ belongs to class 1). For these data points,

$$\begin{aligned}a &= -0.5 \cdot \eta \cdot \sum_i^{N+1} (1)(1) \\ &= -0.5(N + 1)\eta\end{aligned}$$

In this case, in order to remain unsaturated, we need a dimensionality N such that

$$\begin{aligned}-0.5(N + 1)\eta &> -5 \\ (N + 1)\eta &< 10 \\ N &< \frac{10}{\eta} - 1\end{aligned}$$

which is not possible if N can go arbitrarily high. Since this bound only becomes tighter for any other choice of $\mathbf{x}^b$, we conclude that it is not possible to choose a small enough learning rate to prevent the network from having a fixed output after the first training sample (assuming N is not fixed). If $\mathbf{x}^a$ was chosen from C_1, it would simply flip the sign on a and cause it to assign all points to the opposite class.

(iii) With equal probability for the two classes, the mean in each dimension is $\mu_i = \frac{1}{2}(\frac{1}{2}(1 + 2)) + \frac{1}{2}(\frac{1}{2}(3 + 4)) = 2.5$, such that $x_i \in [-1.5, -0.5]$ for C_0 and $x_i \in [0.5, 1.5]$ for C_1. In this case, we note that the sign of a after the first training sample will depend on the sign of $\mathbf{x}^b$, which is no longer strictly positive. Therefore, the network output will never be fixed across all possible data samples after the first input, regardless of N.

(iv) The logistic sigmoid output is strictly positive. As seen above, strictly positive inputs can contribute to instability in the output unit. Since tanh is symmetric around 0, it tends to produce faster, more stable learning.

d. We order roughly from best to worst.

	linear for $x \sim \mathcal{N}(0, \sigma \ll 1)$	vanishing gradient problem	bias shift problem
selu	1	1	1
relu	1	2	2
tanh	2	3	1
σ	2	4	2