Artificial Neural Networks (Gerstner). Solutions 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 ϵ_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?

Solution:

a.

$$\mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K} E_k\right] = \frac{1}{K}\left(\mathbb{E}\left[\sum_{k=1}^{K} E_0\right] + \mathbb{E}\left[\sum_{k=1}^{K} \epsilon_k\right]\right) \tag{1}$$

$$= E_0 \tag{2}$$

b.

$$\mathbb{E}\left[\left(\frac{1}{K}\sum_{k=1}^{K} E_k - \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K} E_k\right]\right)^2\right] \tag{3}$$

$$= \mathbb{E}\left[\left(\frac{1}{K}\sum_{k=1}^{K} E_k - E_0\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{K}\sum_{k=1}^{K} (E_k - E_0)\right)^2\right] \tag{4}$$

$$= \frac{1}{K^2}\mathbb{E}\left[\sum_{k=1}^{K}(E_k - E_0)^2 + \sum_{k\neq n}(E_k - E_0)(E_n - E_0)\right] \tag{5}$$

$$= \frac{1}{K^2}\left(\sum_{k=1}^{K}\mathbb{E}\left[(E_k - E_0)^2\right] + \sum_{k\neq n}\mathbb{E}\left[(E_k - E_0)(E_n - E_0)\right]\right) \tag{6}$$

$$= \frac{1}{K^2}\left(\sum_{k=1}^{K}\mathbb{E}\left[\epsilon_k^2\right] + \sum_{k\neq n}\mathbb{E}\left[\epsilon_k \epsilon_n\right]\right) \tag{7}$$

$$= \frac{1}{K^2}\left(Kv + K(K-1)c\right) \tag{8}$$

$$= \frac{1}{K}v + \frac{K-1}{K}c \tag{9}$$

c. The number of training samples for each model k increases as K increases, thus v decreases. However, the correlation $\rho = \frac{c}{v}$ increases as the folds have more number of data points in common. Substituting $\rho = \frac{c}{v}$ in the formula that we calculated in part (b), we observe the trade off between v and ρ as K varies:

$$\frac{1}{K}v + \frac{K-1}{K}c = v\left(\frac{1}{K} + \frac{K-1}{K}\rho\right). \tag{10}$$

For the case when we have a large number of training samples, the minimum variance is achieved when K takes an intermediate value.

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

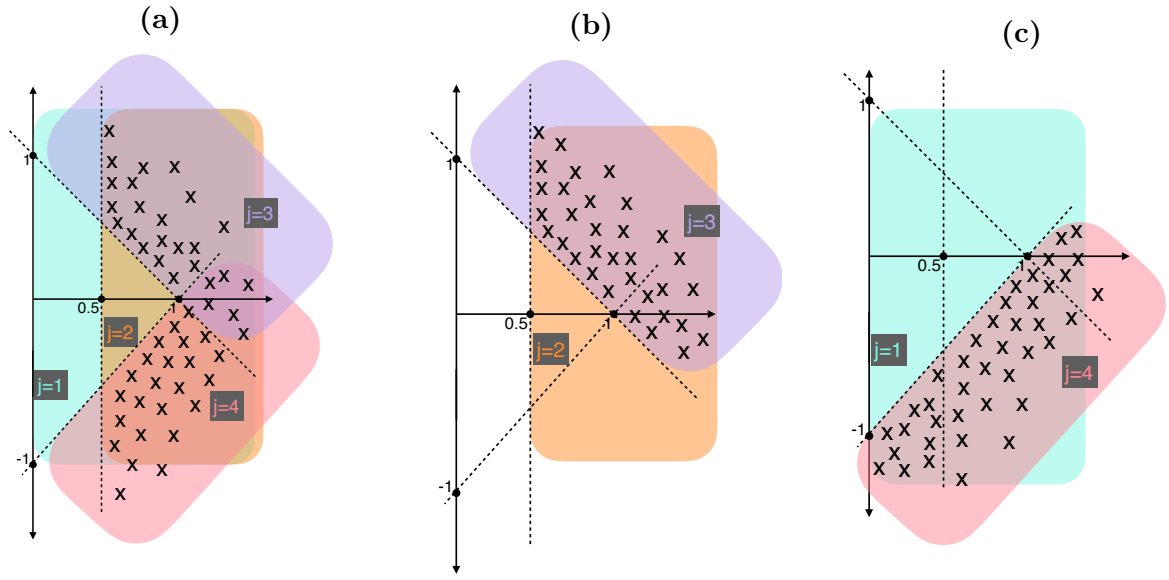
a. Qualitatively sketch the two-dimensional space representing the activity of the 2 neurons in layer $n-2$ and indicate the region (by shading it with crosses x x x) in which neuron i responds positively.

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.

Solution:



a. Since we have a sharp threshold activation function g, every neuron j at layer $n-1$ contributes exactly $+1$ if activated. Neuron i at layer n requires at least 3 active neurons j at layer $n-1$ to get activated (recall that neuron i is connected to all neurons j with weigth $+1$ and it has threshold 2.7). Therefore the crosses in Fig. **(a)** lies at the intersection of at least 3 neurons.

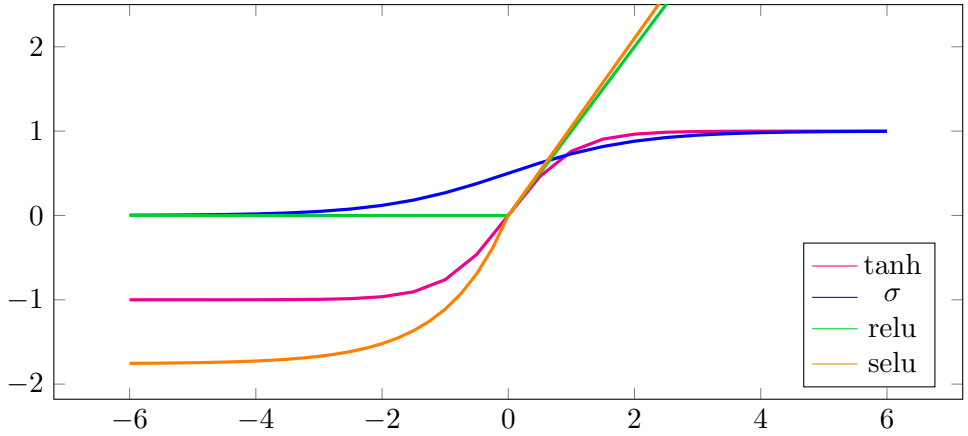
b. Readjusting the weights due to dropout, the weights between neuron i and neurons $j = 2, 3$ will be $+2$ each. Therefore both neurons should be active to exceed the threshold 2.7.

c. Same as (b), just consider neurons $j = 1, 4$ instead.

d. Using dropout, we effectively choose a random subnetwork to update at every training step. An ensemble of the $\binom{4}{2} = 6$ subnetworks –as in (b) and (c)– instead of a single network –as in (a)– with strict decision boundaries should mitigate overfitting and generalize better. The dropout method basically approximates this hypothetical ensembling.

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 δ'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 $i = 1 \dots N$. 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 one of the following activation functions: tanh, σ, 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			
σ			
relu			
selu			

Solution:

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

$$\hat{y}_i = g\left(\sum_j w_{ij}^{(2)} g\left(\sum_k w_{jk}^{(1)} x_k\right)\right) \tag{11}$$

$$= \sum_j w_{ij}^{(2)} \sum_k w_{jk}^{(1)} x_k \tag{12}$$

$$= \sum_{j,k} w_{ij}^{(2)} w_{jk}^{(1)} x_k \tag{13}$$

$$= \sum_k w_{ik} x_k \,, \tag{14}$$

where $w_{ik} = \sum_j w^{(2)}_{ij} w^{(1)}_{jk}$ 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 functions 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 $\sigma(0) = \frac{1}{2}$, 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 14 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,

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

and

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

where we have absorbed the bias by taking $x^a_{N+1} x^b_{N+1} = (-1)(-1) = 1$.

(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. relu does not take negative values, thus it may cause instability whereas selu allows negative values suggesting 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

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 θ_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?

Solution:

a.

$$\langle x_j^{(1)} \rangle = \int_0^\infty \frac{x}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx \tag{15}$$

$$= \frac{-\sigma}{\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \Big|_0^\infty \tag{16}$$

$$= 0 - (-\frac{-\sigma}{\sqrt{2\pi}}) = \frac{\sigma}{\sqrt{2\pi}} \tag{17}$$

b.

$$\langle \tilde{a}_k^{(2)} \rangle = \langle \sum_{j=1}^{N_1} w_{kj}^{(2)} x_j^{(1)} \rangle \tag{18}$$

$$= \sum_{j=1}^{N_1} \langle w_{kj}^{(2)} \rangle \langle x_j^{(1)} \rangle \tag{19}$$

Since $w_{kj}^{(2)}$ and $x_j^{(1)}$ are independent random variables, we can split the expectation into multipliers. The weights in the second layer are initialized with zero mean, therefore $\langle \tilde{a}_k^{(2)} \rangle = 0$. Then we can choose $\theta_k = 0$. Here, note that if $\langle \tilde{a}_k^{(2)} \rangle$ wasn't zero due to different initialization or input distributions, we could still have zero mean $\langle a_k^{(2)} \rangle$ by choosing $\theta_k = \langle \tilde{a}_k^{(2)} \rangle$.

c.

$$\langle [a_k^{(2)}]^2 \rangle = \Big\langle \Big(\sum_{j=1}^{N_1} w_{kj}^{(2)} x_j^{(1)} \Big)^2 \Big\rangle \tag{20}$$

$$= \sum_{j=1}^{N_1} \Big\langle \Big(w_{kj}^{(2)} x_j^{(1)} \Big)^2 \Big\rangle + \sum_{i \neq j} \Big\langle w_{kj}^{(2)} w_{ki}^{(2)} x_j^{(1)} x_i^{(1)} \Big\rangle \tag{21}$$

The second summand here is 0 since each weight is i.i.d. and has zero mean. Now we need to calculate $\langle [x_j^{(1)}]^2 \rangle$. Here, it's important to note that the integrand below is an even function.

$$\langle [x_j^{(1)}]^2 \rangle = \int_0^\infty \frac{x^2}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx \tag{22}$$

$$= \frac{1}{2} \int_{-\infty}^\infty \frac{x^2}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx \tag{23}$$

$$= \frac{\sigma^2}{2} \tag{24}$$

Therefore we have:

$$\langle [a_k^{(2)}]^2 \rangle = \sum_{j=1}^{N_1} \left\langle \left(w_{kj}^{(2)} \right)^2 \right\rangle \left\langle \left(x_j^{(1)} \right)^2 \right\rangle \tag{25}$$

$$= N_1 \frac{c^2}{N_1} \frac{\sigma^2}{2} \tag{26}$$

$$= \frac{c^2 \sigma^2}{2} \tag{27}$$

d. $c = \sqrt{2}/\sigma$

e. Assuming $a_k^{(2)}$ is Gaussian, we can repeat the same steps (a)-(d).