Exercises for the course
Machine Learning 2
Summer semester 2021

Abteilung Maschinelles Lernen
Institut für Softwaretechnik und theoretische Informatik
Fakultät IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Exercise Sheet 3

Exercise 1: Maximum Entropy Distributions (5 + 10 + 5 P)

Consider a discrete random variable X defined over positive numbers $\mathbb{N}$, each of them occuring with respective probability $\Pr(X = k)$. The entropy $H(X)$ is given by

$$H(X) = -\sum_{k=0}^{\infty} \Pr(X = k) \log \Pr(X = k)$$

We would like to find the probability distribution that maximizes the entropy $H(X)$ under the expectation constraint

$$\mathbb{E}[X] = \sum_{k=0}^{\infty} k \Pr(X = k) = 1.$$

and forcing probabilities to be strictly positive, i.e. $\Pr(X = k) > 0$ for all $k \in \mathbb{N}$. The latter can be enforced using the reparameterization $\Pr(X = k) = \exp(s_k)$. To enforce that probability values sum to one, we can impose the additional constraint

$$\sum_{k=0}^{\infty} \Pr(X = k) = 1.$$

The problem of finding the maximum-entropy distribution can therefore be modeled as a constrained optimization problem, and such problems can be solved using the method of Lagrange multipliers.

(a) *Write* the Lagrange function $\mathcal{L}((s_k)_{k\in\mathbb{N}}, \lambda_1, \lambda_2)$ corresponding to the constrained optimization problem above, where $\lambda_1, \lambda_2 \in \mathbb{R}$ are used to incorporate the sum-to-one and the expectation constraints respectively.

(b) *Show* that the probability distribution that maximizes the objective $H(X)$ has the form:

$$\Pr(X = k) = \alpha \cdot \beta^k$$

(c) *Show* that $\alpha = 0.5$ and $\beta = 0.5$, i.e. the probability distribution is geometric with failure and success probabilities 0.5, and k denoting the number of failures before success. (Hint: you can make use of the geometric series and Gabriel's staircase series).

Exercise 2: Independent Components in Two Dimensions (5 + 10 + 10 P)

High entropy can be seen as the result of superposing many independent low-entropy sources. Independent component analysis (ICA) aims to recover the independent sources from the data by finding projections of the data that have low entropy, i.e. that diverge the most from a Gaussian probability distribution.

In the following, we consider a simple two-dimensional problem where we are given a joint probability distribution $p(x, y) = p(x)\, p(y|x)$ with

$$p(x) \sim \mathcal{N}(0, 1),$$
$$p(y|x) = \frac{1}{2}\,\delta(y - x) + \frac{1}{2}\,\delta(y + x),$$

where $\delta(\cdot)$ denotes the Dirac delta function. A useful property of linear component analysis for two-dimensional probability distributions is that the set of all possible directions to look for in $\mathbb{R}^2$ is directly given by

$$\left\{ \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \quad 0 \le \theta < 2\pi \right\}.$$

The projection of the random vector (x, y) on a particular component can therefore be expressed as a function of θ:

$$z(\theta) = x\,\cos(\theta) + y\,\sin(\theta).$$

As a result, ICA in the two-dimensional space is reduced to finding the values of the parameter $\theta \in [0, 2\pi[$ that maximize a certain objective $J(z(\theta))$.

(a) *Sketch* the joint probability distribution $p(x, y)$, along with the projections $z(\theta)$ of this distribution for angles $\theta = 0$, $\theta = \pi/8$ and $\theta = \pi/4$.

(b) *Find* the principal components of $p(x, y)$. That is, find the values of the parameter $\theta \in [0, 2\pi[$ that maximize the variance of the projected data $z(\theta)$.

(c) *Find* the independent components of $p(x, y)$, more specifically, find the values of the parameter $\theta \in [0, 2\pi[$ that maximize the non-Gaussianity of $z(\theta)$. We use as a measure of non-Gaussianity the excess kurtosis defined as

$$\text{kurt}\,[z(\theta)] = \frac{\text{E}\big[(z(\theta) - \text{E}[z(\theta)])^4\big]}{(\text{Var}[z(\theta)])^2} - 3.$$

Exercise 3: Deriving a Special Case of FastICA (5 + 5 + 5 + 10 P)

We consider our input data x to be in $\mathbb{R}^d$ and coming from some distribution $p(x)$. We assume we have applied as an initial step the centering and whitening procedures so that:

$$\text{E}[x] = 0 \qquad \text{and} \qquad \text{E}[xx^\top] = I.$$

To extract an independent component, we would like to find a unit vector w (i.e. $\|w\|^2 = 1$) such that the excess kurtosis of the projected data:

$$\text{kurt}\,[w^\top x] = \frac{\text{E}\big[(w^\top x - \text{E}[w^\top x])^4\big]}{(\text{Var}[w^\top x])^2} - 3.$$

is maximized.

(a) *Show* that for any w subject to $\|w\|^2 = 1$, the projection $z = w^\top x$ has mean 0 and variance 1.

(b) Show using the method of Lagrange multipliers that the projection w that maximizes kurt $[w^\top x]$ is a solution of the equation

$$\lambda w = \text{E}[x \cdot (w^\top x)^3]$$

where λ can be resolved by enforcing the constraint $\|w\|^2 = 1$.

(c) The solution of the previous equation cannot be found analytically, and must instead be solved iteratively using e.g. the Newton's method. The Newton's method assumes that the equation is given in the form $F(w) = 0$ and then defines the iteration as $w^+ = w - J(w)^{-1}F(w)$ where w^+ denotes the next iterate and where J is the Jacobian associated to the function F. *Show* that the Newton iteration in our case takes the form:

$$w^+ = w - (\text{E}[3xx^\top (w^\top x)^2 - \lambda I])^{-1} \cdot (\text{E}[x \cdot (w^\top x)^3] - \lambda w)$$

(d) *Show* that when making the decorrelation approximation $\text{E}[xx^\top (w^\top x)^2] = \text{E}[xx^\top] \cdot \text{E}[(w^\top x)^2]$, the Newton iteration further reduces to

$$w^+ = \gamma \cdot (\text{E}[x \cdot (w^\top x)^3] - 3w)$$

where γ is some constant factor. (The constant factor does not need to be resolved since we subsequently apply the normalization step $w^+ \leftarrow w^+/\|w^+\|$.)

Exercise 4: Programming (30 P)

Download the programming files on ISIS and follow the instructions.

Independent Component Analysis

In this exercise, you will implement an ICA algorithm similar to the FastICA method described in the paper *"A. Hyvärinen and E. Oja. 2000. Independent component analysis: algorithms and applications"* linked from ISIS, and apply it to model the independent components of a distribution of image patches.

In [1]:

```
import numpy
import matplotlib
%matplotlib inline
from matplotlib import pyplot as plt
import sklearn
import sklearn.datasets
import sklearn.feature_extraction.image
import utils
```

As a first step, we take a sample image, extract a collection of (8×8) patches from it and plot them.

In [2]:

```
I = sklearn.datasets.load_sample_image('china.jpg')
X = sklearn.feature_extraction.image.extract_patches_2d(I, (8,8), max_patches=10000, random_state=0)
utils.showimage(I)
utils.showpatches(X)
```

As a starting point, the patches we have extracted are flattened to appear as abstract input vectors of $8 \times 8 \times 3 = 192$ dimensions. The input data is then centered and standardized.

In [3]:

```
X = X.reshape(len(X),-1)
X = X - X.mean(axis=0)
X = X / X.std()
```

Whitening (10 P)

A precondition for applying the ICA procedure is that the input data has variance 1 under any projection. This can be achieved by whitening, which is a transformation $\mathcal{W}:\mathbb{R}^d \to \mathbb{R}^d$ with $z = \mathcal{W}(x)$ such that $\mathrm{E}[zz^\top] = I$.

A simple procedure for whitening a collection of data points $x_1,\dots,x_N$ (assumed to be centered) first computes the PCA components $u_1,\dots,u_d$ of the data and then applies the following three consecutive steps:

1. project the data on the PCA components i.e. $p_{n,i} = x_n^\top u_i$.
2. divide the projected data by the standard deviation in PCA space, i.e. $\tilde{p}_{n,i} = p_{n,i} / \text{std}(p_{:,i})$
3. backproject to the input space $z_n = \sum_i \tilde{p}_{n,i} u_i$.

Task:

- **Implement this whitening procedure, in particular, write a function that receives the input data matrix and returns the matrix containing all whitened data points.**

For efficiency, your whitening procedure should be implemented in matrix form.

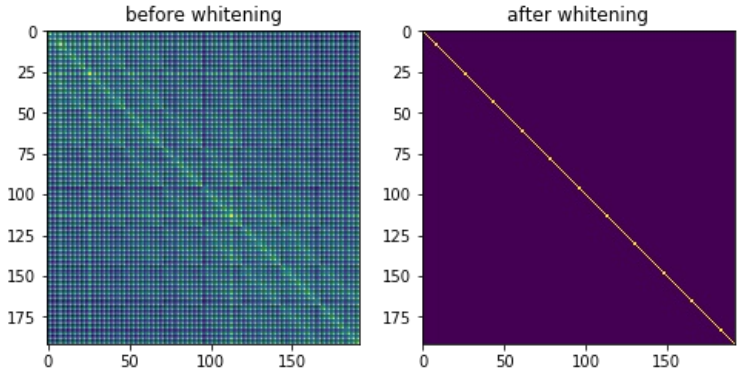
In [4]:

```
##### REPLACE BY YOUR CODE
import solutions
Z = solutions.whitening(X)
#####
```

The code below verifies graphically that whitening has removed correlations between the different input dimensions:

In [5]:

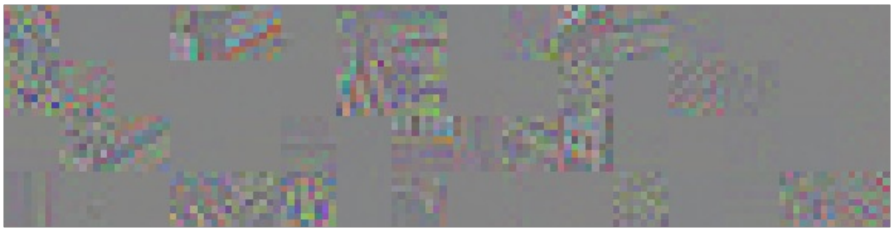
```
f = plt.figure(figsize=(8,4))
p = f.add_subplot(1,2,1); p.set_title('before whitening')
p.imshow(numpy.dot(X.T,X)/len(X))
p = f.add_subplot(1,2,2); p.set_title('after whitening')
p.imshow(numpy.dot(Z.T,Z)/len(Z))
plt.show()
```



Finally, to get visual picture of what will enter into our ICA algorithm, the whitened data can be visualized in the same way as the original input data.

In [6]:

```
utils.showpatches(X)
utils.showpatches(Z)
```



We observe that all high constrasts and spatial correlations have been removed after whitening. Remaining patterns include high-frequency textures and oriented edges of different colors.

Implementing ICA (20 P)

We now would like to learn $h=64$ independent components of the distribution of whitened image patches. For this, we adopt a procedure similar to the FastICA procedure described in the paper above. In particular, we start with random weights $w_1,\dots,w_h \in \mathbb{R}^d$ and iterate multiple times the sequence of operations:

1. $\forall_{i=1}^d~w_i = \mathbb{E}[x \cdot g(w_i^\top x)] - w_i \cdot \mathbb{E}[g'(w_i^\top x)]$
2. $w_1,\dots,w_h=\text{decorrelate}\{w_1,\dots,w_h\}$

where $\mathbb{E}[\cdot]$ denotes the expectation with the data distribution.

The first step increases non-Gaussianity of the projected data. Here, we will make use of the nonquadratic function $G(x) = \frac1a \log \cosh (a x)$ with $a=1.5$. This function admits as a derivative the function $g(x) = \tanh(a x)$, and as a double derivative the function $g'(x) = a \cdot (1-\tanh^2(a x))$.

The second step enforces that the learned projections are decorrelated, i.e.\ $w_i^\top w_j = 1_{i=j}$. It will be implemented by calling in an appropriate manner the whitening procedure which we have already implemented to decorrelate the different input dimensions.

This procedure minimizes the non-Gaussianity of the projected data as measured by the objective:
$$ J(w) = \sum_{i=1}^h (\mathbb{E}[G(w_i^\top x)] - \mathbb{E}[G(\varepsilon)])^2 \qquad \text{where} \quad \varepsilon \sim \mathcal{N}(0,1). $$

Task:

- **Implement the ICA procedure described above, run it for 200 iterations, and print the value of the objective function every 25 iterations.**

In order to keep the learning procedure computationally affordable, the code must be parallelized, in particular, make use of numpy matrix multiplications instead of loops whenever it is possible.

In [7]:

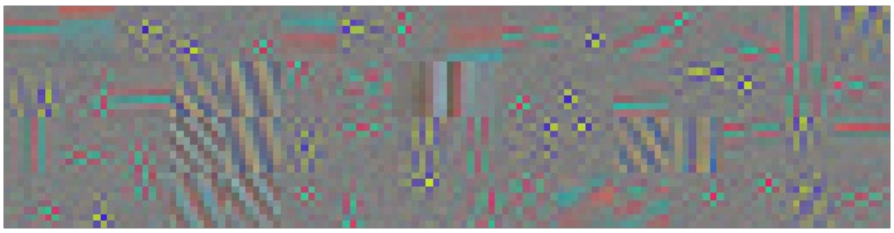
```
##### REPLACE BY YOUR CODE
import solutions
W = solutions.ICA(Z)
#####
```

```
it =  25    J(W) = 1.47
it =  50    J(W) = 1.82
it =  75    J(W) = 1.96
it = 100    J(W) = 2.03
it = 125    J(W) = 2.07
it = 150    J(W) = 2.09
it = 175    J(W) = 2.10
it = 200    J(W) = 2.12
```

Because the learned ICA components are in a space of same dimensions as the input data, they can also be visualized as image patches.

In [8]:

```
utils.showpatches(W)
```



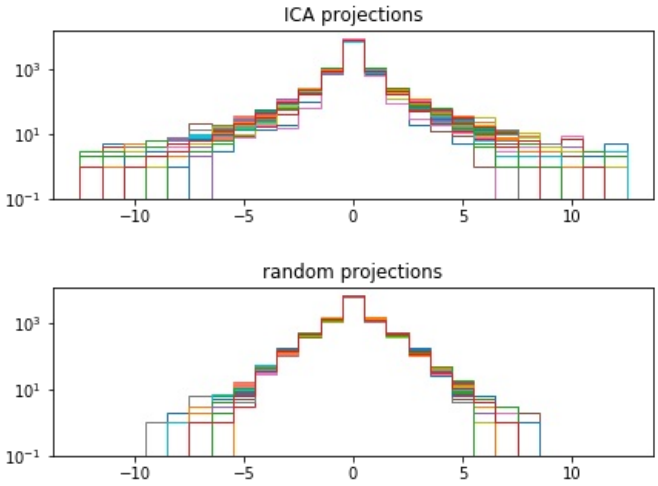
We observe that an interesting decomposition appears, composed of frequency filters, edges filters and localized texture filters. The decomposition further aligns on specific directions of the RGB space, specifically yellow/blue and red/cyan.

To verify that strongly non-Gaussian components have been learned, we build a histogram of projections on the various ICA components and compare it to histograms for random projections.

In [9]:

```
import numpy
plt.figure(figsize=(7,2))
for i in range(64):
    plt.hist(numpy.dot(Z,W[i]),bins=numpy.linspace(-12.5,12.5,26),histtype='step',log=True)
plt.title('ICA projections')
plt.show()

plt.figure(figsize=(7,2))
for i in range(64):
    R = numpy.random.mtrand.RandomState(i).normal(0,1,Z.shape[1])
    plt.hist(numpy.dot(Z,R/(R**2).sum()**.5),bins=numpy.linspace(-12.5,12.5,26),histtype='step',log=True)
plt.title('random projections')
plt.show()
```



We observe that the ICA projections have much heavier tails. This is a typical characteristic of independent components of a data distribution.

Exercise Sheet 3

1 Maximum Entropy Distributions

a) $$\mathcal{L}\left((s_k)_{k\in\mathbb{N}}, \lambda_1, \lambda_2\right) = -\sum_{k=0}^{\infty} e^{s_k} \cdot s_k + \lambda_1 \left(\sum_{k=0}^{\infty} k e^{s_k} - 1\right) + \lambda_2 \left(\sum_{k=0}^{\infty} e^{s_k} - 1\right)$$

$$= \sum_{k=0}^{\infty} e^{s_k}(-s_k) + \sum_{k=0}^{\infty} e^{s_k}(\lambda_1 k) - \lambda_1 + \sum_{k=0}^{\infty} e^{s_k}(\lambda_2) - \lambda_2$$

$$= \sum_{k=0}^{\infty} e^{s_k}(\lambda_1 k + \lambda_2 - s_k) - \lambda_1 - \lambda_2$$

b) $\frac{\partial}{\partial s_k} \mathcal{L}(\cdot) \overset{!}{=} 0 \quad \forall k \in \mathbb{N}$

$$\frac{\partial}{\partial s_k} \mathcal{L}\left((s_k)_{k\in\mathbb{N}}, \lambda_1, \lambda_2\right) = e^{s_k}(\lambda_1 k + \lambda_2 - s_k) - e^{s_k} = 0$$

$$\Leftrightarrow e^{s_k}(\lambda_1 k + \lambda_2 - s_k - 1) = 0 \Leftrightarrow s_k = \lambda_1 k + \lambda_2 - 1$$

$$P_X(X=k) = e^{\lambda_1 k + \lambda_2 - 1} = e^{\lambda_2 - 1} e^{\lambda_1 k}$$

$$= \underbrace{e^{\lambda_2 - 1}}_{\alpha} \underbrace{e^{\lambda_1}}_{\beta}{}^{k} = \alpha \beta^k$$

c) I $\frac{\partial \mathcal{L}(\cdot)}{\partial s_k} \overset{!}{=} 0 \Leftrightarrow P(X=k) = \alpha \beta^k$

II $\frac{\partial \mathcal{L}(\cdot)}{\partial \lambda_2} \overset{!}{=} 0 \Leftrightarrow \sum_{k=0}^{\infty} P(X=k) = 1 \overset{(I)}{\Leftrightarrow} \sum_{k=0}^{\infty} \alpha \beta^k = 1$

$$\Leftrightarrow \frac{\alpha}{1-\beta} = 1 \Leftrightarrow \alpha = 1 - \beta$$

III $\frac{\partial \mathcal{L}(\cdot)}{\partial \lambda_1} \overset{!}{=} 0 \Leftrightarrow \sum_{k=0}^{\infty} k P(X=k) = 1 \overset{(I)}{\Leftrightarrow} \sum_{k=0}^{\infty} k \alpha \beta^k = 1$

$$\Leftrightarrow \alpha \sum_{k=0}^{\infty} k \beta^k = 1 \Leftrightarrow \alpha \frac{\beta}{(1-\beta)^2} = 1 \overset{(II)}{\Leftrightarrow} \frac{(1-\beta)\beta}{(1-\beta)^2} = 1$$

$$\Leftrightarrow \beta = (1-\beta) \Leftrightarrow \beta = 0{,}5$$

2 Independent Components in Two Dimensions

a)

$p(x)$ → sample over x according to Gaussian dist.

$p(y|x)$ → "p of y given x"

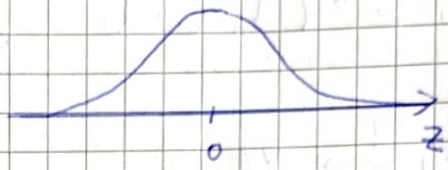
$p(y|x) = \frac{1}{2}\delta(y-x) + \frac{1}{2}\delta(y+x)$

$y = x$ $\quad$ $y = -x$

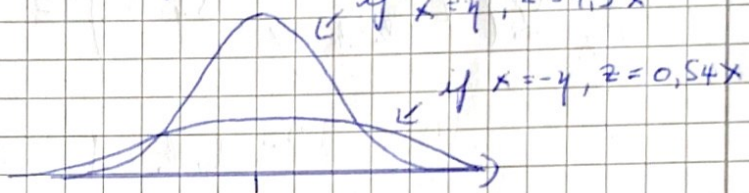
↓

in 50% of samples

If $\sqrt{\beta} = 0$: $z(\sqrt{\beta}) = X$



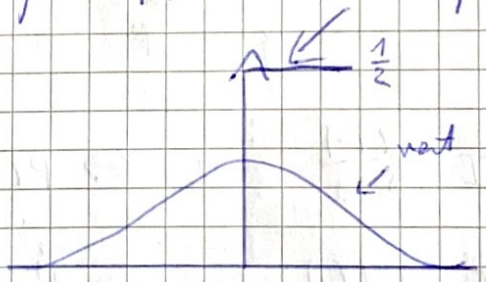
If $\sqrt{\beta} = \frac{1}{8}\pi$: $z = 0{,}92x + 0{,}38y$



If $\sqrt{\beta} = \frac{\pi}{4}\pi$: $z = \frac{\sqrt{2}}{2}(x+y)$ → if $x=y$, $z = \frac{2\sqrt{2}}{2}x = \sqrt{2}x$

→ Gaussian is "taller" → more variance

→ if $x=-y$, $z=0$ → ~~divide~~ peak



b) $E[z] = E[x\cos\vartheta + y\sin\vartheta] = \cos\vartheta \underbrace{E[x]}_{=0} + \sin\vartheta\, E[y]$

$= \sin\vartheta \underset{x}{E}\left[\underset{y|x}{E}[y]\right] = \sin\vartheta \underset{x}{E}[0] = 0$

$\mathrm{Var}[z] = E[(z - \underbrace{E[z]}_{=0})^2] = E[z^2]$

$= E[x^2\cos^2\vartheta + xy\cos\vartheta\sin\vartheta + y^2\sin^2\vartheta]$

$= \cos^2\vartheta \underbrace{E[x^2]}_{=1} + \cos\vartheta\sin\vartheta \underbrace{E[xy]}_{=0} + \sin^2\vartheta \underbrace{E[y^2]}_{=1}$

$= \cos^2\vartheta + \sin^2\vartheta = 1$

c) $E[z(\vartheta)] = 0$, $\mathrm{Var}[z(\vartheta)] = 1$

$\mathrm{kurt}[z(\vartheta)] = \frac{E[(z(\vartheta) - 0)^4]}{1^2} - 3 = E[z(\vartheta)^4] - 3$

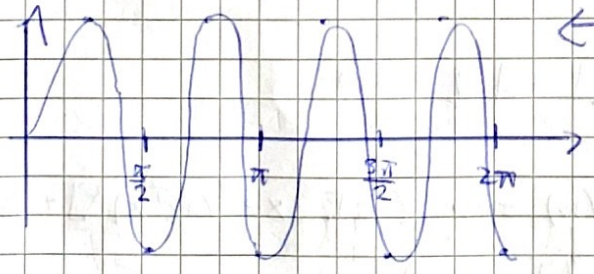
$= E[(x\cos\vartheta + y\sin\vartheta)^4] - 3$

$= E[(x^2\cos^2\vartheta + 2xy\cos\vartheta\sin\vartheta + y^2\sin^2\vartheta)^2] - 3$

$= \cos^4\vartheta\, E[x^4] + 4\cos^2\vartheta\sin\vartheta\, E[x^3y] + 6\cos^2\vartheta\sin^2\vartheta\, E[x^2y^2]$
$+ 4\cos\vartheta\sin^3\vartheta\, E[xy^3] + \sin^4\vartheta\, E[y^4] - 3$

$= E[x^4]\left(\underbrace{(\cos^2\vartheta + \sin^2\vartheta)}_{=1} + 4\cos^2\vartheta\sin^2\vartheta\right) - 3$

$= E[x^4]\left(1 + 1 - \frac{1 - \cos 4\vartheta}{2}\right) - 3$ → $\cos 4\vartheta$ needs to be maximized!



← max.

→ Independent components:

$\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}$

3 Deriving a Special Case of Fast ICA

a) $\underset{x}{E}[w^T x] = w^T \underset{x}{E}[x] = 0$

$\underset{x}{E}[w^T x]$: mean of $w^T x$; $\underset{x}{E}[x] = 0$

$$\text{Var}[w^T x] = \underset{x}{E}[(w^T x)^2] = \underset{x}{E}[(w^T x w^T x)] \; E[w^T x x^T w]$$

$$= w^T \underbrace{\underset{x}{E}[x x^T]}_{=1} w = w^T w = \|w\|^2 = 1$$

b) $$\text{kurt}[w^T x] = \frac{E[(w^T x - \overbrace{E[w^T x]}^{=0})^4]}{\underbrace{(\text{Var}[w^T x])^2}_{=1}} - 3$$

$$= E[(w^T x)^4] - 3$$

$$\mathcal{L}(w, \lambda) \Rightarrow E[(w^T x)^4] - 3 + \lambda_2 (1 - \|w\|) = 0$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial w} \Rightarrow E[\tfrac{\partial}{\partial w}(w^T x)^4] + \lambda_2 (-\tfrac{\partial}{\partial w} w^T w) = 0$$

$$\Leftrightarrow E[4(w^T x)^3 x^T] - 4\lambda_1 w^T = 0$$

$$\Leftrightarrow E[(w^T x)^3 x] - \lambda_1 w$$

$$\Leftrightarrow \lambda_1 w = E[x (w^T x)^3]$$

c) $$F(w) = E[x(w^T x)^3] - \lambda_1 w = 0$$

$$J(w) = \frac{\partial F(w)}{\partial w} = E[\tfrac{\partial}{\partial w} x (w^T x)^3] - \lambda_1 = 0$$

$$= E[3 x x^T (w^T x)^2] - \lambda_1 \underline{1} = 0$$

$$w^+ = w - J(w)^{-1} F(w) = w - (E[3 x x^T (w^T x)^2] - \lambda_1 \underline{1})^{-1} (E[x (w^T x)^3] - \lambda_1 w)$$

d) $w^+ = w - (E[3xx^T(w^Tx)^2] - \lambda \mathbb{1})^{-1} (E[x(w^Tx)^3] - \lambda w)$

$f(w) = E[3xx^T(w^Tx)^2] = 3E[xx^T] E[(w^Tx)^2] - \lambda \mathbb{1}$

$= 3\mathbb{1} - \lambda \mathbb{1} = \mathbb{1}(3-\lambda)$

$w^+ = w - \mathbb{1}^{-1}(3-\lambda)(E[x(w^Tx)^3] - \lambda w)$

$\Leftrightarrow w^+ = w - \dfrac{E[x(w^Tx)^3] - \lambda w}{3-\lambda}$

$\Leftrightarrow (3-\lambda) w^+ = 3w - \lambda w - E[x(w^Tx)^3] + \lambda w$

$\Leftrightarrow (3-\lambda) w^+ = -E[x(w^Tx)^3] + 3w$

$\Leftrightarrow w^+ = \underbrace{-\frac{1}{3-\lambda}}_{\gamma} (E[x(w^Tx)^3] - 3w)$