UNIVERSITY OF FREIBURG

Department of Computer Science

Dr. Joschka Bödecker, Dr. Frank Hutter, Dr. Michael Tangermann

Mockup Exam Machine Learning, Summer Term 2017

Question	Points
1. Short Questions	/20
2. Principal Component Analysis	/10
3. Backpropagation	/10
4. Algorithm-Independent Principles	/10
5. Decision Trees	/10
6. Linear Regression	/10
Total	/70

Short answer question	a	b	c	d	e	f	g	h	i	j
Points (out of 2 each)										

This mockup exam reflects the structure of the real exam. The questions can cover all topics discussed in the course, and you should by no means focus on the topics given here.

In the real exam there will be a maximum of **70 points** as well and you have **90 minutes** to answer the questions. You can write your answers either in **English or German**.

This examination is closed book: **you are not allowed to use any notes or calculators**. Please answer questions in the space provided, and if necessary continue your answers on the back of the same sheet. Please also put your name and matriculation number on every page, in case pages get separated.

Your name: _____

Your matriculation number: _____

Your signature: _____

Good luck!

Part 1 – Short questions, short answers

(a) **(2P)** Name two possible reasons to use linear regression in contrast to a non-linear fit.

Solution: *Smaller computational complexity, less overfitting due to fewer parameters, interpretability of results (e.g. R^2 value) and analytical solution (no iterative scheme necessary).*

(b) **(2P)** Consider the sample covariance matrix (covariance matrix estimated on data) in LDA. What do you commonly observe for the largest and smallest eigenvalues of the matrix in a high-dimensional space with few data points?

Solution: *Largest eigenvalues will be overestimated, smallest eigenvalues will be underestimated.*

(c) **(2P)** Consider the following statement:

"The function $\mathbf{k} : \mathbb{R}^N \text{ x } \mathbb{R}^N \rightarrow \mathbb{R}$ with $\mathbf{k}(x, y) = \frac{x^T y}{||x|| \cdot ||y||}$ defines a positive definite kernel function." Is this true or false? Discuss why.

Solution: *True, see Assignment 7.*

(d) **(2P)** What is the purpose of the VC-dimension in the Vapnik-Chervonenkis inequality?

Solution: *Estimates capacity of hypothesis class; bounds generalization error; bound the difference between empirical and actual risk. Don't accept: VC-dim definition.*

(e) **(2P)** Describe the main idea behind Backpropagation Through Time (BPTT) for training recurrent neural networks.

Solution: *Unroll computation, share weights, then apply regular BP*

(f) **(2P)** Consider applying valid convolutions to an RGB image with 24 x 24 pixels. You use 6 filters of size 5x5 with a stride of 3. What are the output dimensions (width, height, depth)?

Solution: *w = h = (24 - 5) / 3 + 1 = 7; d = 6. Can also figure it out without formula.*

(g) **(2P)** Explain two advantages and two disadvantages of Ensembles of Classifiers (e.g., Bagging, Boosting algorithms).

Solution: *Advantages: 1) Generally higher accuracy 2) Protects against mistakes that a single classifier can make Disadvantages: 1) Higher computational cost 2) Less interpretability.*

(h) **(2P)** Is there an algorithm that performs best on all datasets? If yes, which one is it. If no, explain why not.

Solution: *No, there is no algorithm that performs best on all datasets. Because we can not make any assumptions on what we will see in the test data, one can never know which algorithm is best for a given dataset before testing it. (See also the 'No Free Lunch Theorem'.)*

(i) **(2P)** When estimating the performance of a classifier, one can revert to methods such as k-fold cross-validation and holdout set. Name two advantages of both methods over the other.

Solution: *Holdout: 1) Computationally less expensive. 2) Commonly used in ML challenges and commercial settings; the test set can be truely hidden to ensure an unbiased performance estimation.*

Cross-validation: 1) More accurate estimation, 2) tested on all data points.

(j) **(2P)** Explain in words what the Leave-One-Out procedure does. When do we use it?

Solution: *It is a special form of k-fold cross-validation, where k is the number of data points in the data set. Computationally expensive, used when there is low amount of data available.*

Part 2 – Dimensionality reduction with PCA for classification (10P)

Your task is to write pseudocode for a classification problem. The script should output the estimated classification accuracy on unseen data. Provided are the matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, containing the dataset (N number of samples and d number of features, standardized with zero mean and unit variance), and vector $\boldsymbol{y} \in \{0,1\}^N$, containing the corresponding labels. Several functions are available for its use in the pseudo-script:

`computePCA`
Input:
$\boldsymbol{C_x} \in \mathbb{R}^{d \times d}$, covariance matrix of the original data.
Output:
$\boldsymbol{\lambda} \in \mathbb{R}^d$, vector containing the eigenvalues of $\boldsymbol{W}$, sorted in descending order
$\boldsymbol{W} \in \mathbb{R}^{d \times d}$, matrix containing the corresponding eigenvectors of $\boldsymbol{C_x}$.

`trainClassifier`
Input:
$\boldsymbol{X} \in \mathbb{R}^{N \times d}$, matrix containing the dataset to train the classifier (training set).
$\boldsymbol{y} \in \{0,1\}^N$, vector containing the corresponding labels for each row of $\boldsymbol{X}$.
Output:
Model: Trained classifier.

`applyClassifier`
Input:
Model, classifier model as returned by `trainClassifier`.
$\boldsymbol{X} \in \mathbb{R}^{N \times d}$, dataset on which the labels will be predicted (test set).
Output:
$\boldsymbol{y} \in \{0,1\}^N$, predicted labels.

(a) **(9P)** On the next page, write pseudo-code solving the scenario described above. Include a step of dimensionality reduction using PCA. Choose the abstraction level of your pseudo-code such that your script contains a maximum number of 15 lines.

Solution: Here, we give actual code, but any type of high-level pseudocode would be fine.

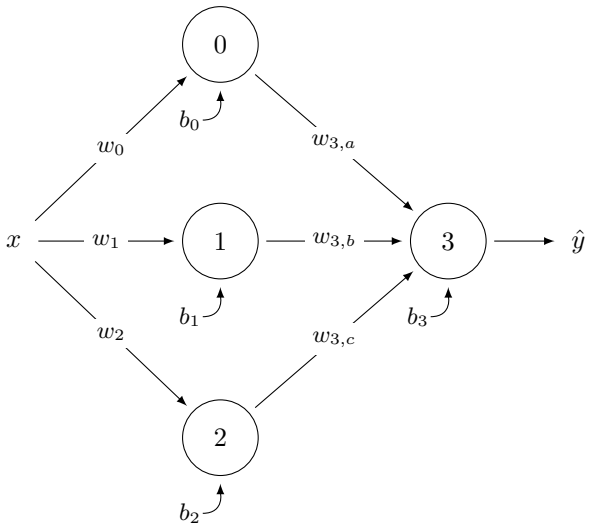
```
accuracy = 1
for idx_sample in 0:X.shape[0]: # Leave one out cross-validation
    # Split train and test data
    X_tr, y_tr = X, y;
    X_tr[idx_sample,:] = []
    Y_tr[idx_sample] = []
    X_te, y_te = X[idx_sample,:], y[idx_sample]

    W, lambda = computePCA(dot(X_tr.T,X_tr))

    # cumulative variance crit.
    ncomp = find(cumsum(lambda)/sum(lambda) >= 0.9)[0]

    model = trainClassifier(dot(X_tr,W[:,0:ncomp]), y_tr)

    y_predict = applyClassifier(model,dot(X_te,W[:,0:ncomp]))

    If y_te != y_predict: accuracy = accuracy - 1/X.shape[0]
```

Scoring:

- **3P** Cross-Validation including accuracy assessment.
- **3P** for the right implementation of PCA projection.
- **3P** for PCA component selection.

(b) **(1P)** Explain your strategy to select the number of PCA components to perform the dimensionality reduction step.
Solution: *The cumulative variance criterion is used to select the number of components: The number of components is set to preserve 0.9 of the original variance. Other strategies include the selection of the eigenvalues that are significantly large, in contrast to the smallest of the set.*

Part 3 – Backpropagation (10P)



Consider the neural network above with weights w_i and biases b_i pictured above. Determine the symbolic expressions for the forward pass **(3P)** and the gradients $\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial b_0}, \frac{\partial L}{\partial b_1}$ with the backward pass **(7P)**!

Use the squared error loss function $L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$. You can reuse previously defined expressions, e.g. you do not need to expand $\frac{\partial L}{\partial \hat{y}}$ again, if you evaluated it previously.

Units 0, 1 and 2 have the rectified linear activation function h_{relu}, while unit 3 is activated linearly with h_{linear}. You are not required to evaluate their derivates and can denote them as h'_{relu} and h'_{linear} respectively. Please use a_i to denote the activation of unit i before applying the activation function, and z_i for its result.

Solution:

	Forward pass:	*Backward pass:*	*Parameter Gradients:*
4.	$\hat{y} = h_{linear}(a_3)$	$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$	
3.	$a_3 = w_{3,a} \cdot z_0 + w_{3,b} \cdot z_1 + w_{3,c} \cdot z_2 + b_3$	$\frac{\partial L}{\partial a_3} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_3} = \frac{\partial L}{\partial \hat{y}} \cdot h'_{linear}(a_3)$	
2.	$z_0 = h_{relu}(a_0)$ $z_1 = h_{relu}(a_1)$ $z_2 = h_{relu}(a_2)$	$\frac{\partial L}{\partial z_0} = \frac{\partial L}{\partial a_3} \frac{\partial a_3}{\partial z_0} = \frac{\partial L}{\partial a_3} w_{3,a}$ $\frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial a_3} \frac{\partial a_3}{\partial z_1} = \frac{\partial L}{\partial a_3} w_{3,b}$	
1.	$a_0 = w_0 \cdot x + b_0$ $a_1 = w_1 \cdot x + b_1$ $a_2 = w_2 \cdot x + b_2$	$\frac{\partial L}{\partial a_0} = \frac{\partial L}{\partial z_0} \frac{\partial z_0}{a_0} = \frac{\partial L}{\partial z_0} h'_{relu}(a_0)$ $\frac{\partial L}{\partial a_1} = \frac{\partial L}{\partial z_1} \frac{\partial z_1}{a_1} = \frac{\partial L}{\partial z_1} h'_{relu}(a_1)$	$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial a_0} \frac{\partial a_0}{\partial w_0} = \frac{\partial L}{\partial a_0} x$ $\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a_1} \frac{\partial a_1}{\partial w_1} = \frac{\partial L}{\partial a_1} x$ $\frac{\partial L}{\partial b_0} = \frac{\partial L}{\partial a_0} \frac{\partial a_0}{\partial b_0} = \frac{\partial L}{\partial a_0}$ $\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial a_1} \frac{\partial a_1}{\partial b_1} = \frac{\partial L}{\partial a_1}$

Part 4 – Algorithm Independent Principles (10P)

Your friend works at a company that develops weather prediction systems. There is one data point per day, and predictions shall be made for the next days. Now your friend wants to try out different types of models and evaluate their performance using cross-validation.

1. **(2P)** Are the individual data points in your friend's data independently distributed? Why or why not?
   Solution: *No. E.g., learning about tomorrow's weather changes our belief about the weather two days from now.*

2. **(2P)** What problems would occur when using standard k-fold cross-validation for this data?
   Solution: *Overfitting. No predictive capability to future data points.*

3. **(6P)** Explain which changes are necessary to the following regular cross validation code (words are enough, you don't need to correct the code) to suit your friend's time series data.

```
num_points = len(X)
num_folds = 10
# Shuffle data to make sure there are no sorting artifacts
indices = np.shuffle(range(num_points))
X = X[indices]
y = y[indices]

fold_indicators = [i % num_folds for i in range (len(X))]
fold_indicators.sort()

for fold in range(num_folds):
    X_train = X[fold_indicator != fold]
    y_train = y[fold_indicator != fold]
    X_valid = X[fold_indicator == fold]
    y_valid = y[fold_indicator == fold]

    model.fit(X_train, y_train)
    scores.append(model.score(X_valid, y_valid)
```

   Solution:

   (a) Don't shuffle the data.

   (b) Possibly: Divide the data in blocks which are of the same length as the prediction horizon.

   (c) Only train on folds which have a lower index than the current one; change equal comparison to lower than comparison.

Part 5 – Decision Trees (10P)

(a) **(2P)** Explain the decision tree algorithm in words. Give the definition of entropy, expressed in terms of $p(v_k)$ where v_k is the number of data points that have value k for attribute v. Think of what you would do in the following three distinct cases that can happen when deciding on a split point in a node:

- All instances have the same class label, but various attribute values
- All instances have the same value for all attributes, but various class labels
- The instances have various class labels and various attribute values

Solution: *A decision tree algorithm aims to create a tree data structure, where every* node *represents a split decision based on the value of a feature and every* leaf *contains the value that is being predicted.*

The decision tree is often constructed from top to bottom (root to leafs) and at every node the feature that splits the data will be selected based on which has the highest information gain.

Entropy $H(V) = -\sum_{k=1}^{K} p(v_k) \log_2 p(v_k)$

Information gain $I = N \cdot H(V) - N_L \cdot H(V_1) - N_r \cdot H(V_r)$*, where* N_l *and* N_r *are the number of data points in the left and right child node, respectively.*

If at some point all instances have the same class label, but various attribute values, we will create a leaf node, predicting the class label.

If at some point all instances have the same value for all attributes, but various class labels, we will create a leaf node with a prediction, for example the mode (classification), mean (regression) or median (regression) of the class labels.

The instances have various class labels and various attribute values a new node is inserted, splitting the data based on the feature that maximizes Information Gain.

(b) **(1P)** Consider the following dataset. Explain why the ID column should not be considered while building your model.

ID	PK	speed	color	buy
1	High	Fast	Blue	Yes
2	High	Slow	Blue	No
3	Low	Fast	Red	Yes
4	Low	Fast	Blue	No

Solution: *It is an arbitrary selected number, it has no correlation with the target attribute. Splitting on this will create a high information gain, however new data points will have presumably unseen values for this attribute.*

(c) **(5P)** Execute the decision tree algorithm on the given dataset, where the labels are defined by the *buy* attribute. You should ignore the ID attribute. Calculate the entropy and information gain for the split in the root node. Use the rounded logarithmic values provided in the table. For all other nodes, you don't need to explicitly calculate the entropy and information gain, as you can infer the best possible split according to this criterion.

$log(1)$	0
$log(3/4)$	-0.4
$log(2/3)$	-0.6
$log(1/2)$	-1
$log(1/4)$	-2
$log(1/3)$	-1.6

Solution: *Entropy root node:*

$$\begin{aligned} H(V) &= -\sum_{k=1}^{K} p(v_k) \log_2 p(v_k) \\ &= -(p(yes) \log_2 p(yes) + p(no) \log_2 p(no)) \\ &= -(0.5 \log_2 0.5 + p(0.5) \log_2 0.5) \\ &= 1 \end{aligned}$$

We can split on PK, Speed and Color. If we split on PK, the information gain is:

$$\begin{aligned} I &= -N \cdot H(V) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k)) \\ &= -N \cdot H(V) + N_l \cdot (p_l(yes) \log_2(p_l(yes)) + p_l(no) \log_2(p_l(no))) \\ &\qquad + N_r \cdot (p_r(yes) \log_2(p_r(yes)) + p_r(no) \log_2(p_r(no))) \\ &= -4 \cdot 1 + 2 \cdot (0.5 \log_2(0.5) + 0.5 \log_2(0.5)) + 2 \cdot (0.5 \log_2(0.5) + 0.5 \log_2(0.5)) \\ &= -4 + 2 + 2 \\ &= 0 \end{aligned}$$

Likewise, we can split on Speed:

$$\begin{aligned} I &= -N \cdot H(V) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k)) \\ &= -N \cdot H(V) + N_l \cdot (p_l(yes) \log_2(p_l(yes)) + p_l(no) \log_2(p_l(no))) \\ &\qquad + N_r \cdot (p_r(yes) \log_2(p_r(yes)) + p_r(no) \log_2(p_r(no))) \\ &= -4 \cdot 1 + 3 \cdot (2/3 \log_2(2/3) + 1/3 \log_2(1/3)) + 1 \cdot (0 \log_2(0) + 1 \log_2(1)) \\ &= -4 + 3 \cdot -0.9 + 1 \cdot 0 \\ &= 1.3 \end{aligned}$$

And we can split on Color, which results in the same information gain.

So, we need to split on Color or Speed (both equally good), both splitting the tree in one leaf node containing one data point and one node containing three data points. Both splits are equally good afterwards, leading to a backbone tree with depth 3.

(d) **(1P)** Is there a decision tree that has a smaller depth than this one? Draw this one. Why did the decision tree algorithm as executed in the previous question not generate this tree?

Solution: *Yes, if we would have chosen to split on PK, the tree could have had depth two. Because the heuristic the tree uses, it does not look ahead and may go initially with suboptimal choices.*

(e) **(1P)** Name two differences between Bagging Decision Trees and Random Forest

Solution: *Random Forests attempt to induce more randomness by restricting the number of features that can be selected from in every node.*

Random Forests are also presumably faster than Bagging, due to this.

(Note that both algorithms induce randomness by training the individual trees on bootstrap samples from the training set.)

Part 6 – Linear Regression (10P)

Farmer Huber has cattle near the Feldberg (Black Forest). He is interested in predicting the weather to know in advance when to lead his animals down to the valley for the winter season. From the German weather service, he downloaded the data containing the daily maximal temperature of the last 500 days. In his notation, the time is called x_i and the temperatures are called y_i for $i = 1 \dots 500$. He has read about linear regression as a model to predict the continuous outcomes for future time points. He also knows that a simple one-dimensional linear regression model could look like $f(x) = w \cdot x + b + \epsilon$ where ϵ is an error term.

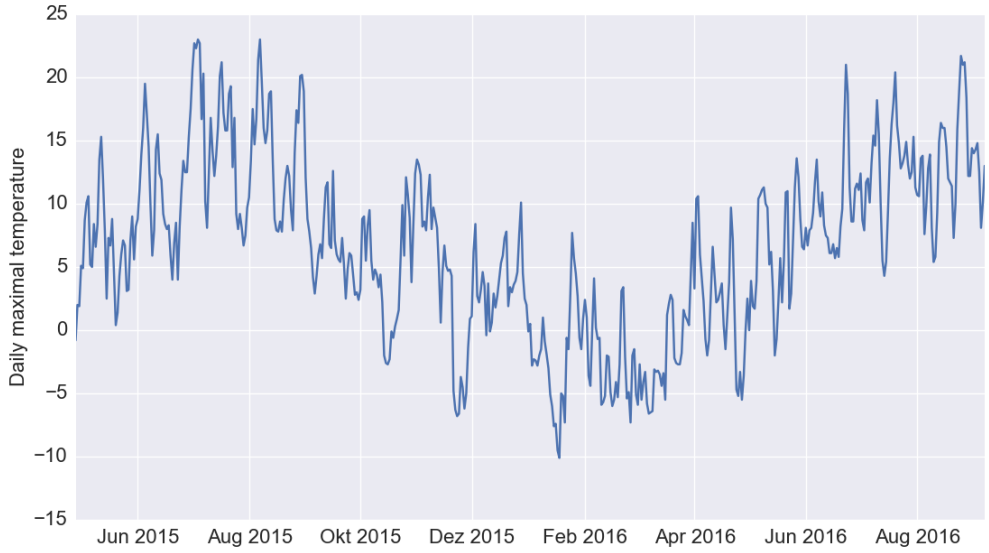


Figure 1: Daily maximum temperature at Feldberg (Black Forest).

(a) **(2P)** Write down the objective function of a linear regression model minimizing the quadratic loss (also known as L_2-loss).

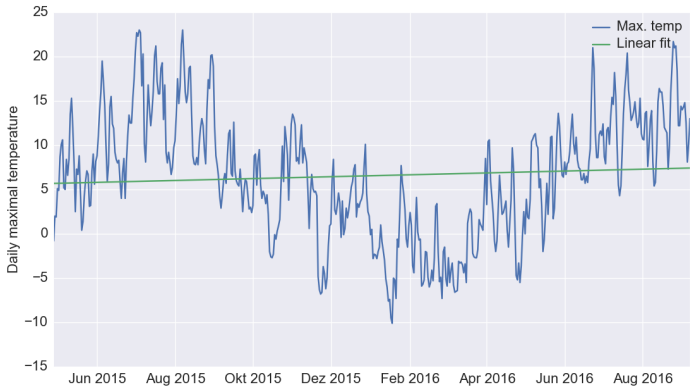
Solution:

$$arg\min_{w,b} \sum_{i=1}^{500} (w \cdot x_i + b - y_i)^2$$

[1P] for correctly writing down the squared loss. [1P] for writing $arg\min_{w,b}$ *Solutions with min instead of argmin were graded full points. Half a point was subtracted for each small mistake, e.g. a missing sum or small notation problems.*

(b) **(2P)** Now, he downloaded a software to fit a linear regression model. The program fits the data and delivers the outputs $b = 5.69$ and $w = 3.48 \cdot 10^{-03}$. Please help farmer Huber by providing an explanation of the temperature development *according to the model* and add the linear fit to Figure 1.

Solution: *The model fit states that initially around May 2015 the temperature was around* $5.69\,^{\circ}\mathrm{C}$ *[0.5P] and rises by* $3.48 \cdot 10^{-03}\,^{\circ}\mathrm{C}$ *[0.5P] per day. Different interpretations also obtain the full number of points (e.g. saying that a small positive temperature trend exists, or that the temperature stays approximately constant). Solutions not referring to temperature obtain no points. The plotted line should start around* $5.69\,^{\circ}\mathrm{C}$ *and end around* $7.5\,^{\circ}\mathrm{C}$ *[1P].*



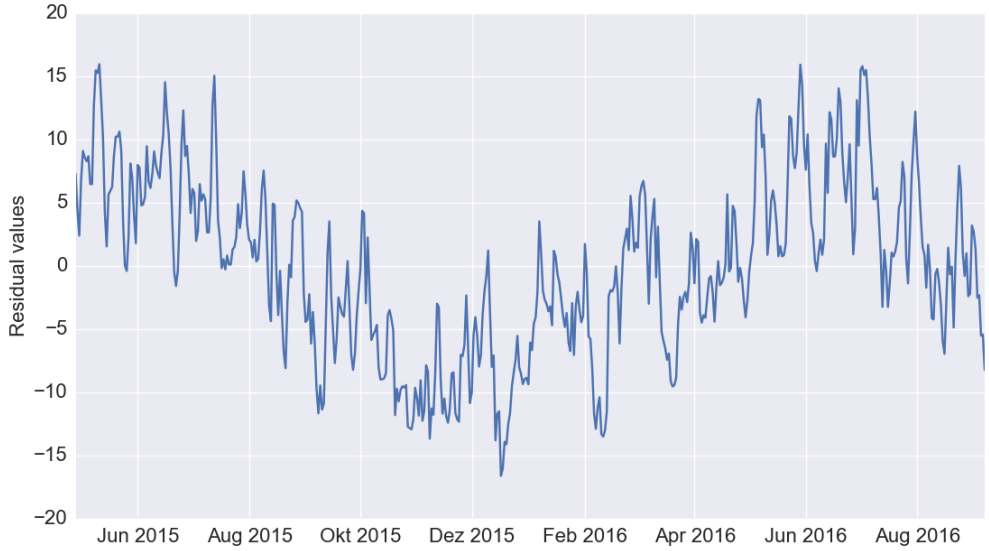


Figure 2: Residual values obtained by the linear fit.

(c) **(2P)** However, you as an expert are not convinced by the results. To further investigate the model, you plot the residuals $f(x_i) - y_i$ for $i = 1 \dots 500$ given in Figure 2. By looking at these residuals, which assumption of the linear regression model is violated? What is the reason for this violation?

Solution: *The residuals are not independent of each other **[1P]**. A strong auto-correlation exists due to seasonal effects and/or a strong correlation between daily temperatures **[1P]**.*

*Other solutions stating that the mean of the error terms is not zero for certain seasons or that the variance of the error terms is not constant for certain intervals also get **1P**.*

(d) **(2P)** In general, linear regression can be extended by adding a regularization term $\alpha^2(b^2 + w^2)$ to the objective function, where α is a parameter determining the strength of the regularization. What is the effect of this additional term and why might it be beneficial?

Solution: *The regularization enforces solutions with smaller norms **[1P]** and this might be beneficial to avoid over-fitting **[1P]**. Mentioning that this helps to reduce the influence of outliers only get **[0.5P]***

(e) **(2P)** Given your background in machine learning, how would you advice Mr. Huber to obtain a more accurate prediction?

Solution: *A solution could be to fit a non-linear model (e.g. sinusoidal model) **[2P]**. Various other solutions are possible.*