4

6

2.2 $X=(1,1,2,2)^T$ $y=(1,2,2,3)^T$ $w=\begin{pmatrix}1\\1\end{pmatrix}$ * (~~I use the~~: I take the transpose because I feel more comfortable with those dimensions)

~~X = addconstant (X)~~

~~w = w - matmul (matmul(X,~~

~~y-pred = matmul (X, w) = $\begin{pmatrix}1&1&2&2\end{pmatrix}$ $\begin{pmatrix}1\\1\end{pmatrix}$~~

X = addconstant (X)

y-pred = matmul (X, w) = $\begin{pmatrix}1&1\\1&1\\1&2\\1&2\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}2\\2\\3\\3\end{pmatrix}$

diff-y = y-pred - y = $\begin{pmatrix}2\\2\\3\\3\end{pmatrix}-\begin{pmatrix}1\\2\\2\\3\end{pmatrix}=\begin{pmatrix}1\\0\\1\\0\end{pmatrix}$

1,5 $w = w - \left(\frac{2}{N}\cdot \underbrace{\text{matmul}(X^T, \text{diff-y})}\right) = \begin{pmatrix}1\\1\end{pmatrix} - \frac{2}{4}\cdot\begin{pmatrix}2\\3\end{pmatrix} = \begin{pmatrix}0\\-0{,}5\end{pmatrix}$

✓

$\begin{pmatrix}1&1&1&1\\1&1&2&2\end{pmatrix}\cdot\begin{pmatrix}1\\0\\1\\0\end{pmatrix}=\begin{pmatrix}2\\3\end{pmatrix}$

0,5 2.3 We could create feature maps in order to add more features to the data so we can take that dependence into account.

2.4.a

0,5 $\nabla_w J(w,x) = \sum(\bar{w}\cdot\bar{x}-y)^3\cdot\bar{x} + \lambda\,\|\bar{w}\|^3$ X

0,25 2.4.b You would just change the gradient in the function with the new formula.

$w = w - \mu \nabla_w J(w,x)$

X 2.4.c The error would be higher using MAE ~~due~~ because it's the MSE but squared. For this exact reason, the only change with this would be that because the error is greater, the steps of the gradient descent would be higher as well, which we could control anyway with μ so there is no reason one should be better than the other one.

Notez avec exactitude votre numéro de table ou d'inscription.
Il est interdit au candidat de signer sa copie ou d'y mettre un signe quelconque pouvant identifier l'auteur de la copie.

1

université PARIS-SACLAY

000053

ANNÉE : ....

UNITÉ D'ENSEIGNEMENT : ....

ÉPREUVE DE : ....

NOTE	19,25 /20	Coefficient	Note affectée du coefficient

NOM : [redacted]
Prénoms : [redacted]
N° D'INSCRIPTION OU DE TABLE
SALLE D'EXAMEN :

Remplissez très lisiblement l'en-tête à gauche et le talon ci-dessous

Si votre composition comporte plusieurs feuilles, numérotez-les .../...

5

1 1.1.a I would say that the model of the left is more expressive because by incrementing the value of λ you get more improvement in the validation score, which means that the model was overfitting, ~~more~~ sign of a complex model. The variation is explained by the fact that with little to no regularization, the model was overfitting the training set and by incrementing the regularization term, ~~you~~ it reduced the overfitting and therefore incrementing the val. score.

0,5 1.1.b I would say that the first model fit better the data because it's a more complex model which ~~can~~ probably can predict more accurately future data. By adding a regularization, you ~~make~~ try to reduce the overfitting and allowing yourself to use such complexity in the model.

No X 1.2 The model of case d is the one which is complex ~~and~~ with regularization because even if it doesn't ~~fit ever~~ overfit the points (because of the regularization), we can see more curves on the line which is a sign of complexity.

?

Notez avec exactitude votre numéro de table ou d'inscription.
Il est interdit au candidat de signer sa copie ou d'y mettre un signe quelconque pouvant identifier l'auteur de la copie.

1.3.a) By incrementing the complexity of the model, you allowed the model to learn the patterns of the data, ~~turning~~ lowering both errors. Also, in the least complex models, ~~the overfitting was to~~ there was overfitting due to its ~~the~~ lack ~~of~~ expressivity. We can see in the more complex models that this overfitting disapear.

1.3.b) Because it's normally very hard to achieve a 100% accuracy because of outlayers or even the complexity of the problem. The same happens with lowering the error to 0.

1.3.c) For that model, we can expect the same error that in the validation set, assuming the validation set hasn't been overfitted by an extensive parameter-tuning process.

1

1.4.a) The solid line is the ~~decision line~~ hyperplane that serves as the decision boundary to clasify the examples. Examples in different sides are going to be clasified ~~as~~ in ~~the~~ the opposite class.

1.4.b) The two dash lines are the lines in which the points are called support vector because it's the line from which there is no error clasifying a point in its truth ground class.

1.4.c) The top 3 seem to be doing a reasonably good job meanwhile the bottom 3 are overfitting the data due to the complexity of the lines. The 4th one could be arguable but the last two are clearly overfitting. ~~Eye~~ There could be controversy on the 3rd one as well due to the line in the left-top corner

1,5

4.d) ~~I would choose the~~ that shows that the model it's creating a circle around the data. However, it's not enough information to conclude there is overfitting.

0,25

1.4.d) I would pick the second one due to it's simplicity because it makes understandable the process of predicting, while having a good score at the same time.

0,5

1.5) The test set should only be used after all the work because otherwise it could be the case that Benji is overfitting the data to the test set by making many predictions on it.

0,25

1.6) ~~We could~~ Seems at first sight that the problem could be the underrepresentation of women and people from different ethnics in the dataset so we could test this hypothesis and rebalance the dataset according to it.

2.1) Linear regression is used for predicting continuous data, that's why it's called regression.

Cost function:

$$J(w) = \frac{1}{N}\sum_{n=1}^{N}(x_n w - y_n)^2$$ → Assuming the following dimensions: $(X_{N,d})(W_{D,1})$ $(Y_{N,1})$

The optimization algorithm is gradient descent which subtract the gradient of the cost function respect to w to w.

2

```
def fit(n_iters, learning_rate, X, y
    w= initialize w();  X= addconstant(x) # used to assume the bias
     for i=0... n_iters:
          y_pred = matmul (X, w)
          w = w - (learning_rate · 2/N · ~~matmul(y_pred - y, X)~~
                      matmul (X^T, y_pred - y)
    return w
```

```
def predict(X, w)
    X= add constant(X)
    return matmul(X, w)
```

6/5

Excellent !

(8)

(4) (1a) The goal is to predict the variable 'edible/poisonous'.

(1b) It's a supervised task ~~while~~ and the sub-category is classification.

(1c) We have a matrix which rows are each mushroom and in the colums are the features (possibly including the predicting one or in other vector y)

(1d) ~~A naive bayes~~ A gaussian naive bayes seems good for this task because it seems simple at first sight, also it would help us understand the model and not use a black box so we can learn more about mushrooms :)

1,25

(1e) We should be worried about the imbalance of the classes because of the fact that there are ~~most~~ more edible mushrooms. Also, we care more about predicting correct the poisonous ones since otherwise we could die :(We should use a performance metric which takes into account more the false negatives (wrongly predict 'edible') than the false positives.

(2a) The goal is predicting the winning team giving two pokemon teams.

(2b) It's a supervised task of clasification.

(2c) We could have a 3dimensional structure with dimensions $N^{ex} \times N^{pok} \times D$ in which N^{ex} is the number of fights, N^{pok} is the number of pokemon per fight and D is the attributes per pokemon. We could unroll the last two so we have a 2d matrix which is easier to train a model with, having a $N^{ex} \times (N^{pok} \cdot D)$ dimensional matrix.

✓

(2d) This is a very difficult problem since the features are correlated and the same pokemon can appear in different indices in each fight. I would use a deep neural network which will take a lot of time to train but since it should take all possibilities into account, is a reasonable price.

1,25

(2e) We should particularly worry about the complexity of the problem to try to reduce it as much as we could so we are able to predict accurately the results. Also, keep in mind the overfitting ~~so we~~ applying regularization or even dropout to the model.

✓

Notez avec exactitude votre numéro de table ou d'inscription.
Il est interdit au candidat de signer sa copie ou d'y mettre un signe quelconque pouvant identifier l'auteur de la copie.

(9)

université PARIS-SACLAY

ANNÉE : ....

UNITÉ D'ENSEIGNEMENT : ....

ÉPREUVE DE : ....

NOTE /20	Coefficient	Note affectée du coefficient

NOM : [redacted]

Prénoms :

N° D'INSCRIPTION OU DE TABLE

SALLE D'EXAMEN :

Remplissez très lisiblement l'en-tête à gauche et le talon ci-dessous

Si votre composition comporte plusieurs feuilles, numérotez-les .../...

0,75

(2.5a) This means that ~~as~~ in each epoch, we would ~~train~~ update the weight as many times as mini-batches exist.

```
def fit(...)
    mini_batches = doBatches(X)
    for i=0...n_iter:
        for mini-batch in mini batches:
            $w = w - \eta \nabla_\theta J(\text{mini-batch})$
    return w
```

?

✓

0,5

(2.5.b) First, because we update the weights more frequently, it's faster. And second, we mitigate the noise issue by creating mini-batches which are not representative of all the data since it is the case as well in the full-batch (because of the noise). This will give least-represented points more importance in their mini-batches.

(2.5.c) As I said in the previous question it is faster because you update the weights #mini-batches times per epoch instead of only 1. Even if it sometimes it doesn't take a good direction, it converges faster usually.

✗

Notez avec exactitude votre numéro de table ou d'inscription.
Il est interdit au candidat de signer sa copie ou d'y mettre un signe quelconque pouvant identifier l'auteur de la copie.

PARTIE A RABATTRE

3.1.a ~~$\sigma(\bar{w}_{12}\,\bar{x}) - (t_{n1} - t_{n2}) =$~~

When $k=1$: $(t_{n1} - t_{n2}) = 1 - 0 = 1$

When $k=2$: $(t_{n1} - t_{n2}) = 0 - 1 = -1$

We ~~see~~ that we have 1 and -1 as labels so we can create the variable $t_{n,12}$ to be this new encoding. $t_{n,12} = 1$ if $k_n = 1$ else $t_{n,12} = -1$.

ok ~

0,25

3.1.b $\bar{w}_{21}$ is the weight of ~~the bias~~ the first feature for predicting the likelihood of being class 1.
$\bar{w}_{21}$ is the weight of the bias term to predict the second class.
They don't seem to be related.

0,25

If $t_{n,21}$ is 1 then $t_{n,12} = -1$ so: $t_{n,ij} = -t_{n,ji}$

3.1.c ~~$\sigma(\bar{w}\cdot x_n)$ if $k_n =$~~ ~~$t_{n,kk'}$~~

$$t_{n,kk'} = \begin{cases} 1 & \text{if } k_{\text{true}}^{(k)} = 1 \\ -1 & \text{if } k_{\text{true}}^{(k')} = 1 \\ 0 & \text{otherwise} \end{cases}$$

✓ 0,5 / 0,75

3.2 There are the features of X which is D and the one-hot vectors of Y which are $\frac{K\cdot(K-1)}{2}$ so:

$$D + \frac{K(K-1)}{2}$$

0

3.3 $J(\theta, X) = \sum_{n=1}^{N} (w^T \cdot x_n) - t_n)^2 \cdot t_n$

0,0

3.4.a

0,5

3.5 Because $t_{n,kk'}$ would be 0, there won't be any updates in ~~the to~~ w.

✓

3.6 ~~We have~~ The prediction is $k_{pred}^{test} = w^T X$. We have $\frac{K(K-1)}{2}$ predictions and we take the max as the predicted class.

~

```
def predict (...)
    argmax ( matmul ( w^T, X )
    return preds
```

0,25

3.7 I would choose the accuracy as performance metric.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \cdot 100$$

~

In case of imbalanced classes we may choose the balanced accuracy instead.

3.8 The hyper-parameters are: train ~~and~~ validation and test size of our data, the learning rate, the way we initialize the parameters, ~~the~~ the stopping criterion, the error/loss function chosen, ~~the preprocessing st~~ ~~and~~ the activation function and the way to choose the prediction (max, softmax...).

0,5

2,25

université PARIS-SACLAY

9

ANNÉE : ....

UNITÉ D'ENSEIGNEMENT : ....

ÉPREUVE DE : ....

NOTE /20	Coefficient	Note affectée du coefficient

Remplissez très lisiblement l'en-tête à gauche et le talon ci-dessous

NOM : [redacted]

Prénoms :

N° D'INSCRIPTION OU DE TABLE

SALLE D'EXAMEN :

Si votre composition comporte plusieurs feuilles, numérotez-les .../...

(4) (3a) The goal is to predict future temperature given historical data.

(3b) It's a ~~regression~~ supervised regression task of time-series.

(3c) We have a vector of all the records of temperature in chronological order.

(3d) We could use a recurrent neural network since it could take into account previous temperature as well as the weights of the network to predict the future temperature.

[+0,25] ↓ 1,75

TB

Yes

(3e) We should worry about the fact that we lack a lot of variables to predict future temperature such as CO_2 emisions so ~~it's~~ it would be very complicated to create a reliable model. Also, the variance between seasons could undercover the ~~temperature~~ increment of the temperature over the years due to climate change. I would propose a feature map taking the average of the whole year ~~and then creating a simpler model that given the avg. temperature of~~ , ~~and~~ the averaged mean of the last 4 months, giving a greater impact on the closest months and ~~a boolean with the curren~~ the substraction of the last 2-months temperature so we know if its increasing (positive) or decreasing (negative).

Notez avec exactitude votre numéro de table ou d'inscription.
Il est interdit au candidat de signer sa copie ou d'y mettre un signe quelconque pouvant identifier l'auteur de la copie.

Notez avec exactitude votre numéro de table ou d'inscription.
Il est interdit au candidat de signer sa copie ou d'y mettre un signe quelconque pouvant identifier l'auteur de la copie.

PARTIE
À
RABATTRE

10

4a The goal is to predict the position played of a player given attributes of the player.

4b It's a supervised ~~multilabel~~ multilabel classification task.

4c We have a matrix in which the rows are each player and the columns are their attributes. ~~Note~~

4e The non-standarized labels could be solved by hiring experts on the field to unify ver similar positions due to the limited set of classes and the (seemingly) ease of the task. Another approach would be applying a clustering algorithm to the players and then unify those positions that are very close to each other (this could turn into a weird result though). ~~An~~ We could also mix both approaches and suggest a lot of unification to the experts. Another problem to tackle would be the feature selection since probably not all the features are useful and the ratings could also be biased due to the human-made task.

✓

✓

1,75

TB

4d A seemingly good algorithm for this task would be a decision tree or random forest classifier since they create clear set of rules to predict the ~~[illegible]~~ position, which could be very useful in this problem because certain positions needs a certain level of skills (a center player doesn't need to be as tall as a defense player).