ECE 5973 - Artificial Neural Networks

Homework 2

Sarah Brown

Question 1

(10 points) Follow the derivation in the tutorial, show that the dual optimization problem of above is indeed shown as (10) there by filling in any missing steps.

With the type of training data we are given, we want to find a function f(x) that has a epsilon small deviation from the obtained targets for all of the training data. This means that we will not accept any errors that are larger than epsilon.

Start this deviation with linear function f as shown below. With <.,.> representing a dot product.

$$f(x) =< w, x > +b$$
$$with w \in X, b \in R$$

So we want f(x) to be "flat." Flat here means that we want a small w. It is possible to get a small w by minimizing the norm

$$||w||^2 =< w, w >$$

So we want something that minimizes:

$$\frac{1}{2}||w|^2$$
$$with: \ y_i - < w, x_i > -b \leq \epsilon$$
$$and\ with: \ < w, x_i > +b - y_i \leq \epsilon$$

The above comes with an assumption that it is possible to approximate the targets y with the training data x with a preffered precision. This might not be possible in all cases.

Can also add slack variables to adjust for some of the weird constraints of the optimization. This gives:

$$\frac{1}{2}||w|^2 + C\Sigma_{i=1}^{l}(\xi_i + \xi_i^*)$$
$$with: \ y_i - < w, x_i > -b \leq \epsilon$$
$$and\ with: \ < w, x_i > +b - y_i \leq \epsilon$$
$$and\ with: \xi_i, \xi_i^* \geq 0$$

The new constant C adjusts flatness of f and the number of deviations larger than epsilon. This

gives us an epsilon-insensitive loss function:

$$|\xi_\epsilon := \begin{cases} 0 & \text{if } |\xi| \leq \epsilon \\ |\xi| - \epsilon & \text{otherwise} \end{cases}$$

The derivation continues by constructing a Lagrange function and showing that the function has saddle points. This Lagrange function is defined as follows.

$$L := \frac{1}{2}||w||^2 + C\Sigma_{i=1}^{l}(\xi_i + \xi_i^*) - \Sigma_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) - \Sigma_{i=1}^{l}\alpha_i(\epsilon + \xi_i - y_i + < w, x_i > + b) - \Sigma$$

With $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ as Lagrange multiplers.

These variables have to be positive. In addition, since we want it to have a saddle point this means that the partial derivatives of L go away. This seems to be due to the behavior of Lagrange functions in general.

The partial derivatives of with respect to the primal variables are:

$$\delta_b L = \delta_b(frac12||w||^2 + C\Sigma_{i=1}^{l}(\xi_i + \xi_i^*) - \Sigma_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) - \Sigma_{i=1}^{l}\alpha_i(\epsilon + \xi_i - y_i + < w, x_i >$$
$$= 0 + C * 0 - 0 - \Sigma_{i=1}^{l}\alpha_i(0 + 0 - 0 + 0 + 1) - \Sigma_{i=1}^{l}(\alpha_i^*(0 + 0$$
$$\rightarrow \delta_b L = \Sigma_{i=1}^{l}(\alpha_i^* - \alpha_i)$$

$$\delta_w L = \delta_w(frac12||w||^2 + C\Sigma_{i=1}^{l}(\xi_i + \xi_i^*) - \Sigma_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) - \Sigma_{i=1}^{l}\alpha_i(\epsilon + \xi_i - y_i + < w, x_i >$$
$$= w - \Sigma_{i=1}^{l}\alpha_i(x_i) - \Sigma_{i=1}^{l}(\alpha_i^*(-x_i)$$
$$\rightarrow \delta_w L = w - \Sigma_{i=1}^{l}x_i(\alpha_i^- \alpha_i^*)$$

$$\delta_{\xi_i^{(*)}} L = \delta_{\xi_i^{(*)}}(frac12||w||^2 + C\Sigma_{i=1}^{l}(\xi_i + \xi_i^*) - \Sigma_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) - \Sigma_{i=1}^{l}\alpha_i(\epsilon + \xi_i - y_i + < w, x$$
$$\delta_{\xi_i^{(*)}} L = \delta_{\xi_i^*} L = C - \alpha_i^{(*)} - \eta_i^{(*)}$$

Where $\xi_i^{(*)}$, $\alpha_i^{(*)}$ and $\eta_i^{(*)}$ refer to the pair of variables with *'s.

These partials all equal 0 which gives:

$$\delta_b L = \Sigma_{i=1}^{l}(\alpha_i^* - \alpha_i)$$
$$\delta_w L = w - \Sigma_{i=1}^{l}x_i(\alpha_i^- \alpha_i^*)$$
$$\delta_{\xi_i^{(*)}} L = \delta_{\xi_i^*} L = C - \alpha_i^{(*)} - \eta_i^{(*)}$$

These can then be substituted back into our Lagrange function:

$$\text{maximize} \begin{cases} -\frac{1}{2}\Sigma_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) < x_i . x_j > \\ -\epsilon\Sigma_{i=1}^{l}(\alpha_i + \alpha_i^*) + \Sigma_{i=1}^{l}y_i(\alpha_i - \alpha_i^*) \end{cases}$$
$$\text{subject to} \Sigma_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \text{and} \alpha_i, \alpha_i^* \in [0, C]$$

Question 2

(10 points) Repeat Q3 in HW1 using support vector regression. You can use any package this time (e.g., scikit-learn's SVR implementation).

In [6]:
```
import yfinance as yf
import pandas as pd
import numpy as np

from sklearn.svm import SVR
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

def get_price(tick,start='2020-10-01',end=None):
    return yf.Ticker(tick).history(start=start,end=end)['Close']

def get_prices(tickers,start='2020-10-01',end=None):
    df=pd.DataFrame()
    for s in tickers:
        df[s]=get_price(s,start,end)
    return df

feature_stocks=['tsla','fb','twtr','amzn','nflx','gbtc','gdx','intc','dal','c
predict_stock='msft'

# training set
start_date_train='2020-10-01'
end_date_train='2020-12-31'

X_train=get_prices(feature_stocks,start=start_date_train,end=end_date_train)
y_train=get_prices([predict_stock],start=start_date_train,end=end_date_train)

# testing set
start_date_test='2021-01-01' # end date omit, default is doday
X_test=get_prices(feature_stocks,start=start_date_test)
y_test=get_prices([predict_stock],start=start_date_test)

X_train=np.array(X_train)
y_train=np.array(y_train)
X_test=np.array(X_test)
y_test=np.array(y_test)

dummyFeatureX_train = np.array(pd.get_dummies(X_train[0]))
dummyFeatureX_test = np.array(pd.get_dummies(X_test[0]))
ytrain = np.ravel(y_train,)
regr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
regr.fit(X_train, ytrain)
```

Out[6]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svr', SVR(epsilon=0.2))])
```

In [7]:
```
y_predictionsPrice = regr.predict(X_test) # uses model to make predictions
y_predictionsPrice
```

Out[7]:
```
array([218.17559645, 217.8798197 , 216.21813407, 216.10230375,
       216.0850678 , 215.6176551 , 215.22703363, 215.15906639,
       214.94520899, 214.96452617, 215.01883469, 215.21239336,
       215.05585853, 215.69975801, 215.79971263, 215.80932964,
```

```
215.91675675, 216.10278161, 215.92729545, 216.08805395,
215.64840417, 215.85225246, 215.4678803 , 215.61189305,
215.23525863, 215.17263976, 215.05341804, 214.96288961,
214.91363886, 214.89795085, 214.89971195, 214.90094963,
214.89123284, 214.90827755, 214.91035836, 214.895755  ,
214.89319209, 214.89097688, 214.88853461, 214.89588524,
214.89512588, 214.91083549, 214.9057116 , 214.89268958,
214.89866211, 214.90086077, 214.89643728, 214.8967951 ])
```

In [ ]: