Homework 11

1. Dual of a QP and differentiability

(a) consider $p^* = \max_x c^T x - \frac{1}{2} x^T Q x : Ax \le b$

$= \min_x (-c^T x + \frac{1}{2} x^T Q x)$

$Q \in S^n_{++}$, $Q > 0$ } symmetric positive definite

$\mathcal{L}(\vec{x}, \vec{\lambda}) = -c^T \vec{x} + \frac{1}{2} \vec{x}^T Q \vec{x} + \vec{\lambda}^T (A\vec{x} - \vec{b})$

Lagrangian

dual function

$g(\vec{\lambda}) = \min_x \mathcal{L}(\vec{x}, \vec{\lambda})$

Then the dual problem

$$\boxed{d^* = \max_{\vec{\lambda} \ge 0} g(\vec{\lambda})}$$

Primal problem in standard form

$p^* = \min_x -c^T x + \frac{1}{2} x^T Q x$

s.t. $Ax - b \le 0$

$A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ $c \in \mathbb{R}^n$

$Q \in S^n_{++}$

$f_0 = -c^T x + \frac{1}{2} x^T Q x$

$f_1 = Ax - b$

1) Hession $\nabla^2 f_0 = Q > 0$ is positive definite $\therefore$ convex

2) constraint[s] f_1 is affine and

$\exists x_0 \in \text{Relint}(D)$ s.t. $Ax_0 - b \le 0$ ⇝ sol'n in half space

therefore feasible, satisfying weak slaters condition

By 1) and 2) Strong Duality holds

(b) $p^* = \min_x -c^T x + \frac{1}{2} x^T Q x$

$$= \max_{\vec{\lambda} \geq 0} \underbrace{\min_x -\vec{c}^T \vec{x} + \frac{1}{2} \vec{x}^T Q \vec{x} + \vec{\lambda}^T (A\vec{x} - \vec{b})} = d^*$$

$\lambda \in \mathbb{R}^m$

$$\nabla_x \mathcal{L}(\vec{x}, \vec{\lambda}) = -\vec{c} + Qx + A^T \vec{\lambda} = 0$$

$A_{n \times m}$

$$\Rightarrow \vec{x} = Q^{-1}(\vec{c} - A^T \vec{\lambda})$$

minimum

$Q^{-1} \in S^n_{++}$

plugging in $\vec{x}$

$$p^* = \max_{\vec{\lambda} \geq 0} \underbrace{-c^T Q^{-1}(\vec{c} - A^T \vec{\lambda}) + \frac{1}{2}(\vec{c}^T - \vec{\lambda}^T A) Q^{-1} Q Q^{-1} (\vec{c}^T - A^T \vec{\lambda}) + \vec{\lambda}^T (A Q^{-1}(\vec{c} - A^T \vec{\lambda}) - \vec{b})}$$

let $h(c) = -\vec{c}^T Q^{-1} \vec{c} + c^T A^T \vec{\lambda} + \frac{1}{2}(\vec{c}^T Q^{-1} - \lambda^T A Q^{-1})(\vec{c}^T - A^T \vec{\lambda}) + \vec{\lambda} A Q^{-1}(\vec{c} - A^T \vec{\lambda}) - \vec{\lambda}^T \vec{b}$

$$\nabla_c^2 h(c) = -2Q^{-1} + Q^{-1}$$

$$= -Q^{-1} \prec 0 \qquad \text{since } Q \succ 0 \Leftrightarrow Q^{-1} \succ 0$$

so p^* is concave function of c

(c) $p^*(c) = \max_x f_x(c)$

where $f_x(c) = c^T x - \frac{1}{2} x^T Q x$

where $f(c) = \max_x f_x(c)$

since $p^*(c)$ is concave, in order to find $f(c)$ we take the gradient

$$\nabla_x f_x(c) = 0$$

$$c - Qx = 0$$

$x = Q^{-1} c \Rightarrow x^T = c^T Q^{-1}$ $\qquad Q \in S^n_+ \Rightarrow Q^{-1} \in S^n_+$

$$\Rightarrow f(c) = c^T Q^{-1} c - \frac{1}{2} c^T Q^{-1} c$$

$$= \frac{1}{2} c^T Q^{-1} c$$

Then we can take any subgradient of this function

subgradient g at x s.t.

$$\forall z \quad f(z) \geq f(x) + g^T (z - x)$$

Taylor expand around c

$$f(z) \simeq \frac{1}{2} c^T Q^{-1} c + (Q^{-1} c)^T (z - c)$$

then subgradient $\boxed{g = c^T Q^{-1}}$

(d). Assume 2 subgradients g_1, g_2 at every point x

$$f(z) \geq f(x) + g_1^T(z-x) = \frac{1}{2}x^TQ^{-1}x + g_1^T(z-x)$$
$$f(z) \geq f(x) + g_2^T(z-x) = \frac{1}{2}x^TQ^{-1}x + g_2^T(z-x)$$

since this is true for all z

2. KKT conditions

$$\min_{x \in \mathbb{R}^n} \sum^n \left(\tfrac{1}{2} d_i x_i^2 + r_i x_i\right)$$

s.t. $a^T x = 1$, $x_i \in [-1, 1]$, $i = 1, \dots, n$

where $|a_i| \geq 1$ and $d_i > 0$ for $i = 1, \dots, n$

(a) $D = \operatorname{diag}(d_1, \dots, d_n)$, $r = (r_1, \dots, r_n)$

$$\left.\begin{aligned} \tfrac{1}{2} x^T D x &= \sum^n \tfrac{1}{2} d_i x_i^2 \\ r^T x &= \sum^n r_i x_i \end{aligned}\right\} \text{ by definition}$$

so $\min_x \sum \frac{1}{2} d_i x_i^2 + r_i x$ ...

can be written as

$$\boxed{\begin{aligned} &\min_x \tfrac{1}{2} x^T D x + r^T x \\ &\quad \text{s.t. } a^T x \leq 1 \\ &\qquad\quad x_i^2 \leq 1 \end{aligned}}$$

$x_i \in [-1, 1]$ is encompassed in $x_i^2 \leq 1$

(b) • The objective and constraints are differentiable

let $f_0 = \frac{1}{2}x^T D x + r^T x$

$h_1 = a^T x - 1$ equality constraint function

$f_1 = x_i^2 - 1$

$$\Rightarrow \left.\begin{aligned} \nabla f_0 &= Dx + r \\ \nabla h_1 &= a \\ \frac{\partial f_1}{\partial x_i} &= 2x_i \end{aligned}\right\}$$

$\therefore$ objective and constraints are differentiable

• Strong duality holds

$\nabla^2 f_0 = D \succ 0$ since $d_i > 0 \ \forall i$

$\therefore$ objective function is convex

equality constraint

$a^T x - 1 = 0$ is satisfied if we let

let $x^T = (0, \cdots, \frac{1}{a_k}, \cdots, 0)$ for some $|a_k| > 1$

strict

then inequality constraint

$x_i^2 - 1 < 0$ for $i = 1, \ldots, n$ except for $i = k$

strict

and for $x_k^2 - 1 = \frac{1}{a_k^2} - 1 < 0$ since $|a_k| > 1$

Therefore the problem is strictly feasible, ie satisfies slaters condition

and since the the problem is convex

$\boxed{\text{Strong duality holds}}$

f_0, f_1 are convex and h_1 is affine

• The optimization problem is convex

$f_0 = \frac{1}{2} x^T D x + r^T x$ is convex

$\nabla^2 f_0 = D \succ 0$ since hessian is positive definite

$f_1 = x_i^2 - 1$ is convex since

$f_1'' = 2 > 0$ second derivative is positive

and

$h_p = a^T x - 1$ is affine

$\therefore$ this problem is convex

(c) show $\mathcal{L}(x,\lambda,\mu) = \frac{1}{2}x^T(D+\Lambda)x + (r+\mu a)^T x - (\mu + \frac{1}{2}\sum \lambda_i)$

where $\Lambda = \mathrm{diag}(\lambda)$

$\lambda = (\lambda_1, \dots, \lambda_n)$: inequality $\quad \mu$: equality

$$\mathcal{L}(x,\lambda,\mu) = \frac{1}{2}x^T D x + r^T x + \underbrace{x^T \Lambda x - \lambda \mathbb{1}}_{\text{inequality}} + \underbrace{\mu a^T x - \mu}_{\text{equality}}$$

If we write inequality constraint as $\frac{1}{2}x_i^2 \le \frac{1}{2}$

$$\Rightarrow \mathcal{L}(x,\lambda,\mu) = \frac{1}{2}x^T D x + r^T x + \frac{1}{2}x^T \Lambda x - \frac{1}{2}\lambda \mathbb{1} + \mu a^T x - \mu$$

$$= \boxed{\frac{1}{2}x^T(D+\Lambda)x + (r+\mu a)^T x - (\mu + \frac{1}{2}\lambda\mathbb{1})}$$

where $\mathbb{1}^T = (1, \dots, 1)$ $\boxtimes$

KKT conditions

Primal feasibility

$a^T x \le 1$

$x_i^2 \le 1$

Dual feasibility

$\lambda_i \ge 0$

complementary slackness

$$\frac{1}{2}x^T\Lambda x - \frac{1}{2}\lambda \mathbb{1} = 0$$

Stationarary

$$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$$

$$= (D+\Lambda)x + r + \mu a = 0$$

$$x = -(D+\Lambda)^{-1}(r+\mu a)$$

(d) from (c)

$$\nabla_x \mathcal{L}(x, \lambda, \mu) = 0$$

$$\Rightarrow \quad x^* = -\underbrace{(D + \Lambda^*)^{-1}}(r + \mu^* a)$$

in terms of optimal $\lambda: \lambda^*$ and $\mu: \mu^*$

componentwise

$$x^* = -\frac{(r_i + \mu_i^* a_i)}{d_i + \lambda_i^*}$$

since $(D + \Lambda)$ is diagonal

∎

(e) plugging in previous x^* to $\mathcal{L}(x, \lambda, \mu)$

matrix form

$$\mathcal{L}(x^*, \lambda, \mu) := g(\lambda, \mu)$$

$$= \frac{1}{2} x^{*T}(D + \Lambda) x^* + (r + \mu a)^T x^* - \left(\mu + \frac{1}{2} \lambda \mathbb{1}\right)$$

$$= +\frac{1}{2}(r + \mu a)^T (D + \Lambda)^{-1} (r + \mu a)$$

$$- (r + \mu a)^T (D + \Lambda)^{-1} (r + \mu a) \qquad - \left(\mu + \frac{1}{2} \lambda \mathbb{1}\right)$$

$$= -\frac{1}{2}(r + \mu a)^T (D + \Lambda)^{-1} (r + \mu a) - \left(\mu + \frac{1}{2} \lambda \mathbb{1}\right)$$

componentwise (rearranging)

$$= \boxed{-\mu - \frac{1}{2} \sum^{n} \left[\frac{(r_i + \mu a_i)^2}{d_i + \lambda_i} + \lambda_i \right]}$$

∎

(f) Find $\max_{\lambda \geq 0} g(\lambda, \mu)$ for fixed μ

note $g(\lambda,\mu)$ is concave so we need only take the gradient and set it equal to zero

$$\nabla_\lambda g(\lambda,\mu) = 0$$

$$= -\mu - \frac{1}{2} \sum^n \left[\frac{(r_i + \mu a_i)^2}{d_i + \lambda_i} + \lambda_i \right] = 0$$

$$= \frac{1}{2} \sum \left[\frac{(r_i + \mu a_i)^2}{(d_i + \lambda_i)^2} + 1 \right] = 0$$

$$\frac{(r_i + \mu a_i)^2}{(d_i + \lambda_i)^2} = -1$$

$$|r_i + \mu a_i| - d_i = \lambda_i$$

where

$|r_i + \mu a_i| \geq d_i$ - in order to satisfy $\lambda_i \geq 0$

(g) $x^*(\mu) = g(\lambda^*, \mu)$ where $\lambda_i^* = |r_i + \mu a_i| - d_i$

$$= \boxed{-\mu - \frac{1}{2}\sum^{n} \frac{(r_i+\mu a_i)^2}{|r_i+\mu a_i|} + |r_i+\mu a_i| - d_i}$$

(h)

Scanned with CamScanner

3. A matrix problem with strong duality

$p^* = \min_{\Delta} c^T(A+\Delta)^{-1}b \ : \ \|\Delta\| \le 1$ $\quad$ $\|\cdot\|$: largest singular value norm

$\sigma_{min}(A) > 1$ strict $\quad$ smallest singular value of A

(a) objective function

$f_0(\Delta) = c^T(A+\Delta)^{-1}b$ is well-defined for $\|\Delta\| \le 1$

if $(A+\Delta)^{-1}$ exists

since $\sigma_{min}(A) > 1$ A has no singular values equal to zero

This implies A is full rank, since A is square, A is invertible

proof

$$\Sigma = \begin{pmatrix} \sigma_{max} & & 0 \\ & \ddots & \\ 0 & & \sigma_{min} > 1 \end{pmatrix}_{n \times n}$$

by assumption $\sigma_{min}(A) \ge 1$

since $\Sigma y = 0 \Rightarrow y = 0$

and U and V^T are orthogonal matrices

$\Rightarrow A$ is invertible

$\Rightarrow \det(A) \ne 0$

since $\det(A+\Delta) \ge \det(A) + \det(\Delta)$

and $\det(A) \ne 0$

Scanned with CamScanner

$\det(A+\Delta) \geq \det(A) + \det(\Delta) > 0$

is long as $\det(A) \neq -\det(\Delta)$

than $\det(A+\Delta) \neq 0$

$$\det(A) = \det(U\Sigma V^T) = \det(U)\det(\Sigma)\det(V^T)$$
$$= \det(U)\det(\Sigma)\det(V)$$
$$= \pm\det(\Sigma)$$

since $\det(I) = \det(U^TU) = \det(U)^2 = 1$

$\det(U) = \pm 1$

U, V are orthogonal

$\det(\Delta) = \det(\Sigma_\Delta)$

since $\sigma_{max}(\Delta) \leq 1$

and $\sigma_{min}(A) > 1$

$\det(A) = \det(\Sigma_A) \neq \det(\Sigma_\Delta) = \det(\Delta)$

$\therefore \det(A+\Delta) \neq 0$

and is therefor invertible

(b) Not
convex

(c) $p^* = \min_{\Delta, t} t$

s.t $b^T (A+\Delta)^{-1} b \leq t$

$\|\Delta\| \leq 1$

$t - b^T (A+\Delta)^{-1} b \geq 0$

let $M = \begin{pmatrix} A+\Delta & b^T \\ b & t \end{pmatrix} \succeq 0$

$\Rightarrow \quad A+\Delta \succ 0$

4. (a) the problem

$$\min_{w} \|Xw - y\|_2^2 + \lambda \|w\|_2^2$$

Id change labels from $\{0, 1\}^n$ to

$\{-1, 1\}$

then the problem will find a w that will bring the data points close to either -1 or 1 then any $w_i > 0$ classifies the corresponding data point to 1 and vice versa, essentially classifying the data.

Support Vector Machine Vs. Ridge Regression

In this problem, we compare linear SVM and Ridge Regression in the task of classification. As we shall see, formulating the problem as different optimization problems (here SVM and Ridge Regression) makes a difference in performance. There are three places with todos, follow the todos to complete this problem.

In [21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import Ridge
from sklearn.metrics import accuracy_score
import pdb
```

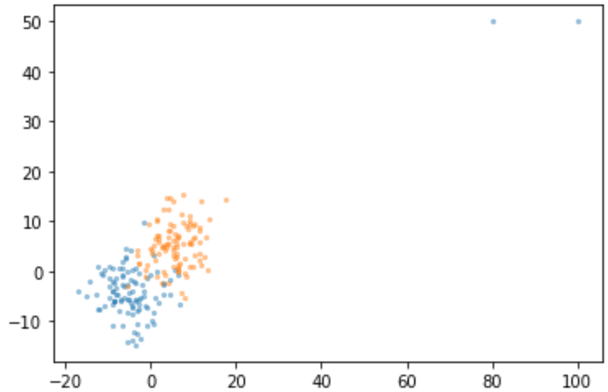
In [7]:
```
# Helper function for visualization. No todo here.
# Usage: plot_boundry(X, y, fitted_model)
#     X: your features, where each row is a data sample
#     y: your labels, can be 0/1 or -1/1
#     fitted_model: a scipy TRAINED model, such as sklearn.svm.SVC
#
def plot_boundry(X, y, fitted_model):

    plt.figure(figsize=(9.8,5), dpi=100)

    for i, plot_type in enumerate(['Decision Boundary']):
        plt.subplot(1,2,i+1)

        mesh_step_size = 0.5  # step size in the mesh
        x_min, x_max = X[:, 0].min() - .1, X[:, 0].max() + .1
        y_min, y_max = X[:, 1].min() - .1, X[:, 1].max() + .1
        x_max = 110
        y_max = 60
        xx, yy = np.meshgrid(np.arange(x_min, x_max, mesh_step_size), np.ar
ange(y_min, y_max, mesh_step_size))
        if i == 0:
            Z = fitted_model.predict(np.c_[xx.ravel(), yy.ravel()])
            Z = np.sign(Z)
        else:
            try:
                Z = fitted_model.predict_proba(np.c_[xx.ravel(), yy.ravel
()])[:,1]
            except:
                plt.text(0.4, 0.5, 'Probabilities Unavailable', horizontala
lignment='center',
                     verticalalignment='center', transform = plt.gca().tran
sAxes, fontsize=12)
                plt.axis('off')
                break
        Z = Z.reshape(xx.shape)
        plt.scatter(X[y==0,0], X[y==0,1], alpha=0.4, label='Edible', s=5)
        plt.scatter(X[y==1,0], X[y==1,1], alpha=0.4, label='Posionous', s=
5)
        plt.imshow(Z, interpolation='nearest', cmap='RdYlBu_r', alpha=0.15,
                   extent=(x_min, x_max, y_min, y_max), origin='lower')
        plt.title(plot_type)
        plt.gca().set_aspect('equal');

    plt.tight_layout()
    plt.subplots_adjust(top=0.9, bottom=0.08, wspace=0.02)
```

In [11]:
```
# load the data
train_data = np.load("ridge_vs_svm_data_train.npy")
X_train = train_data[:, 1:]
y_train = train_data[:, 0]

test_data= np.load("ridge_vs_svm_data_test.npy")
X_test = test_data[:, 1:]
y_test = test_data[:, 0]
```

Here we visualize the training data to get a sense of the distribution. Note the outliers.

In [12]:
```
plt.scatter(X_train[y_train==0,0], X_train[y_train==0,1], alpha=0.4, s=5)
plt.scatter(X_train[y_train==1,0], X_train[y_train==1,1], alpha=0.4, s=5)
```

Out[12]: <matplotlib.collections.PathCollection at 0x7f7f1dc70fd0>

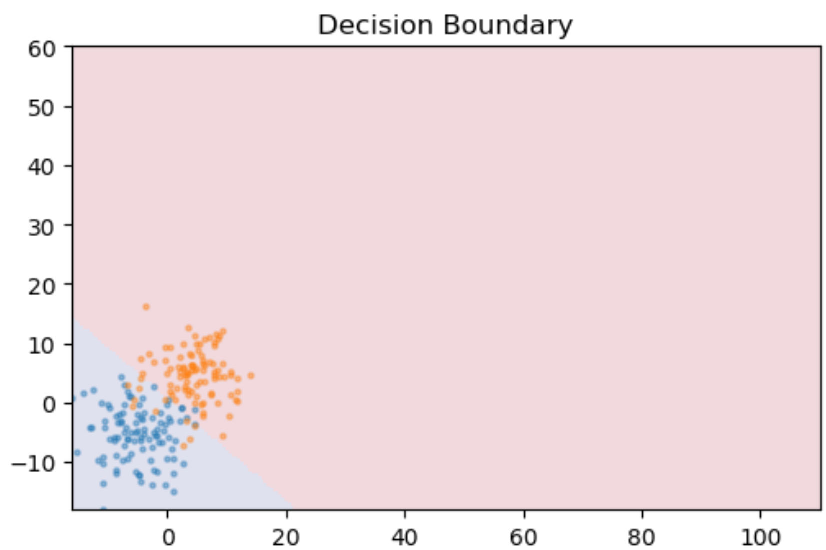


SVM

Fill in the code below to run a **linear** svm to classify the data.

In [36]:
```
fitted_model = None # your trained model (as trained by scipy)
y_pred = None # the prediction of your trained model on the testing data

########## Your beautiful code starts here ##########
# todo: Write code to train an SVM, and generate prediction y_pred.
# Optional: Try using a linear kernel and a polynomial kernel.  How does th
e value of C matter?

# svc = SVC(kernel='poly',gamma='auto')
svc = SVC(kernel='linear')
fitted_model = svc.fit(X_train, y_train)

########## Your beautiful code ends here ##########

y_pred = svc.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)
print("Test Accuracy: {}".format(accuracy))

plot_boundry(X_test, y_test, fitted_model)
```

```
Test Accuracy: 0.92
```

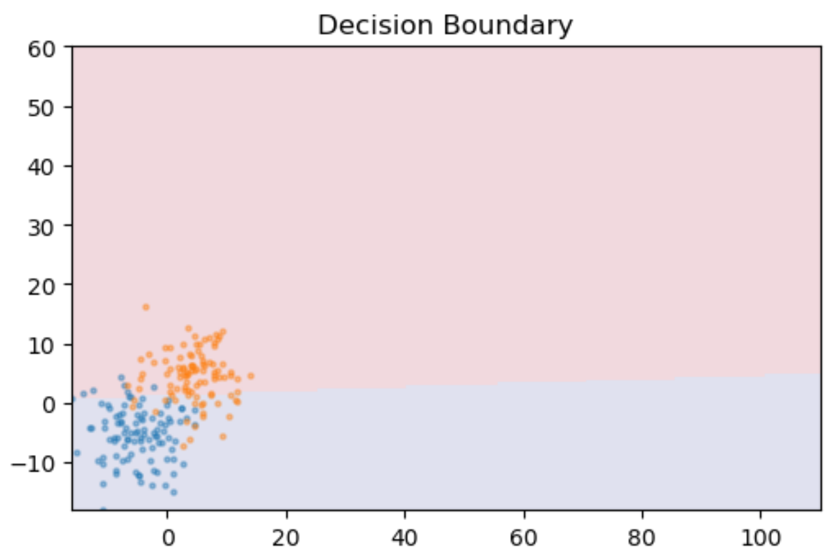


Ridge Regression

Fill in the code below to run ridge regression to classify the data.

In [37]:
```
fitted_model = None # your trained model (as trained by scipy)
y_pred_sign = None # the prediction of your trained model on the testing da
ta

# convert the labels from 0 and 1 to -1 and 1
y_train_sign = np.array(y_train)
y_test_sign = np.array(y_test)
y_train_sign[y_train_sign == 0] = -1
y_test_sign[y_test_sign == 0] = -1

# for the regularization parameter lambda, you can try something around 0.1
:)
# Optional: try choosing different parameters
llambda = 0.1

########## Your beautiful code starts here ##########
# todo: train a fitted_model and run prediction to generate y_pred_sign
clf = Ridge(alpha=llambda)
fitted_model = clf.fit(X_train, y_train_sign)
y_pred_sign = clf.predict(X_test)
y_pred_sign[y_pred_sign < 0] = -1
y_pred_sign[y_pred_sign > 0] = 1
# pdb.set_trace()
########## Your beautiful code ends here ##########

accuracy = accuracy_score(y_pred_sign, y_test_sign)
print("Test Accuracy: {}".format(accuracy))

plot_boundry(X_test, y_test, fitted_model)
```

```
Test Accuracy: 0.85
```



Why Do We See SVM Outperforming Ridge Regression?

In the above, we saw that SVM outperforms ridge regression because SVM is more robust to outliers. The data was actually synthetically generated from two Gaussians --- but remember the two outliers? Can you see how they are impacting the classifer?

The support vector machine finds a line that seperates the data whereas ridge regression is a penalized least squares. So the decision boundary is drawn based on how far away points are rather than how well it seperates the data.

How the Data was produced

In [38]:
```
# Optional: Try changing the positions of the outliers to see how they impa
ct the performanace

n = 100
cov = np.eye(2) * 20

pos = np.hstack([
    np.ones(n).reshape([-1, 1]),
    np.random.multivariate_normal([5, 5], cov, size=n),
])
neg = np.hstack([
    np.zeros(n).reshape([-1, 1]),
    np.random.multivariate_normal([-5, -5], cov, size=n),
])

syn = np.vstack([pos, neg])

outliers = np.array([
    [0, 80, 50,],
    [0, 100, 50,],
])
syn = np.vstack([pos, neg, outliers])
np.random.shuffle(syn)
np.save("ridge_vs_svm_data_train.npy", syn)


pos_test = np.hstack([
    np.ones(n).reshape([-1, 1]),
    np.random.multivariate_normal([5, 5], cov, size=n),
])
neg_test = np.hstack([
    np.zeros(n).reshape([-1, 1]),
    np.random.multivariate_normal([-5, -5], cov, size=n),
])

syn = np.vstack([pos_test, neg_test])

np.random.shuffle(syn)
np.save("ridge_vs_svm_data_test.npy", syn)
```

Credit

Spring 2019: Mong H. Ng, Prof. Ranade
Plotting function from https://github.com/devssh/svm/blob/master/SVM%20Python/Classifier%20Visualization.ipynb (https://github.com/devssh/svm/blob/master/SVM%20Python/Classifier%20Visualization.ipynb)