1. (a).

$$f(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\sum_{k=1}^{K} y_{ik}x_i^T\theta_k - log(\sum_{k=1}^{K} exp(x_i^T\theta_k)))$$

$$\frac{d(f(\theta))}{d\theta_k} = \frac{1}{m}\sum_{i=1}^{m}(\frac{d(\sum_{k=1}^{K} y_{ik}x_i^T\theta_k)}{d\theta_k} - \frac{d(log(\sum_{k=1}^{K} exp(x_i^T\theta_k)))}{d\theta_k})$$

$$= \frac{1}{m}\sum_{i=1}^{m}(y_{ik}x_i - \frac{1}{\sum_{j=1}^{K} exp(x_i^T\theta_j)} \cdot \frac{d(\sum_{k=1}^{K} exp(x_i^T\theta_k))}{d\theta_k})$$

$$= \frac{1}{m}\sum_{i=1}^{m}(y_{ik}x_i - \frac{1}{\sum_{j=1}^{K} exp(x_i^T\theta_j)} \cdot exp(x_i^T\theta_k) \cdot \frac{d(x_i^T\theta_k)}{d\theta_k})$$

$$= \frac{1}{m}\sum_{i=1}^{m}(y_{ik}x_i - \frac{1}{\sum_{j=1}^{K} exp(x_i^T\theta_j)} \cdot exp(x_i^T\theta_k) \cdot x_i)$$

$$= \frac{1}{m}\sum_{i=1}^{m}(y_{ik} - \frac{exp(x_i^T\theta_k)}{\sum_{j=1}^{K} exp(x_i^T\theta_j)})x_i$$

(b).

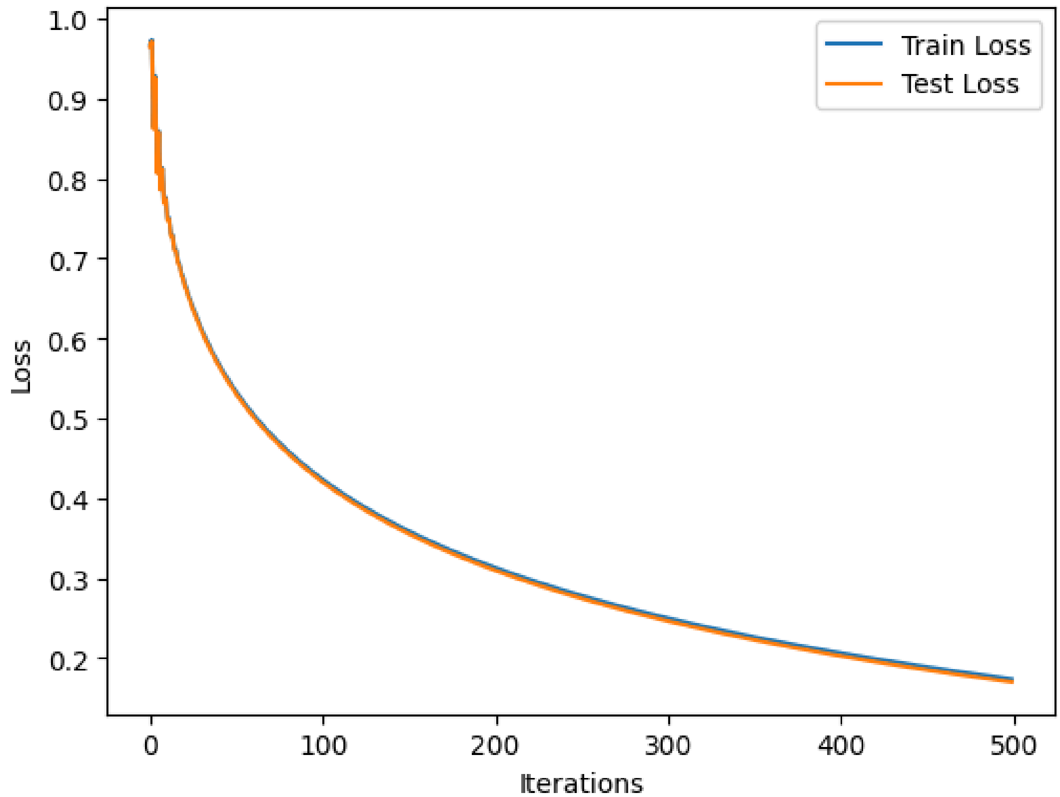
1. Use D = max θi, θi - D <= 0. This makes exp(θi - D) <= 1, which prevents exp(θi) from becoming very large due to very large values in θi, thus preventing overflow.

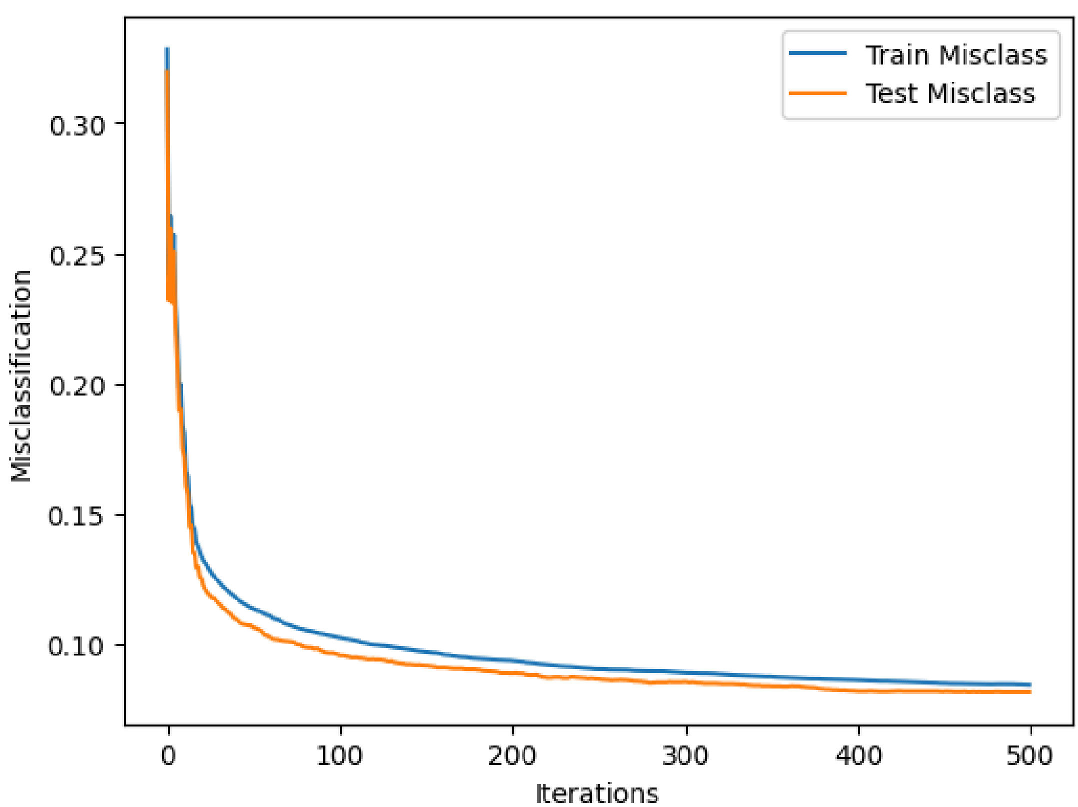
2. Use D = min θi, θi - D >= 0. This makes exp(θi - D) >= 1, which prevents exp(θi) from becoming very close to zero due to very negative values in θi, thus preventing underflow.

(c).

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
from sklearn.preprocessing import OneHotEncoder

data = sio.loadmat('mnist.mat')
Xtrain, Xtest = data['trainX'].astype(float), data['testX'].astype(float)
ytrain, ytest = data['trainY'][0], data['testY'][0]
ytrain_onehot = OneHotEncoder(categories='auto').fit_transform(ytrain.reshape(-1, 1))
m, n = Xtrain.shape

def compute_gradient(X, y, theta):
    s = np.dot(X, theta)
    exp_s = np.exp(s)
    gradient = exp_s.T / np.sum(exp_s, axis=1)
    return np.dot(X.T, (gradient.T - y)) / m

def compute_loss(X, y, theta):
    s = np.dot(X, theta)
    exp_s = np.exp(s)
    gradient = exp_s.T / np.sum(exp_s, axis=1)
    return np.mean(np.log(np.abs(gradient - y)))

def accuracy(X, y, theta):
    preds = np.argmax(np.dot(X,theta), axis=1)
    return np.mean(preds == y)
```

```
stepsize = 1e-5
Theta = np.zeros((n, 10))
train_loss, test_loss, train_misclass, test_misclass = [], [], [], []
for _ in range(500):
    Theta -= stepsize * compute_gradient(Xtrain, ytrain_onehot, Theta)
    train_loss.append(compute_loss(Xtrain, ytrain, Theta))
    test_loss.append(compute_loss(Xtest, ytest, Theta))
    train_misclass.append(1 - accuracy(Xtrain, ytrain, Theta))
    test_misclass.append(1 - accuracy(Xtest, ytest, Theta))

train_loss, test_loss = np.array(train_loss), np.array(test_loss)
train_misclass, test_misclass = np.array(train_misclass), np.array(test_misclass)
plt.plot(train_loss, label="Train Loss")
plt.plot(test_loss, label="Test Loss")
plt.xlabel('Iterations')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.plot(train_misclass, label="Train Misclass")
plt.plot(test_misclass, label="Test Misclass")
plt.xlabel('Iterations')
plt.ylabel('Misclassification')
plt.legend()
plt.show()
```


Train Loss
Test Loss
Loss
Iterations
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
100
200
300
400
500


Train Misclass
Test Misclass
Misclassification
Iterations
0.30
0.25
0.20
0.15
0.10
0
100
200
300
400
500

2. (a) total $= 5+100+30+3+2 = 140$

$$H(X) = -\left(\frac{5}{140}\log_2\frac{5}{140} + \frac{100}{140}\log_2\frac{100}{140} + \frac{30}{140}\log_2\frac{30}{140} + \frac{3}{140}\log_2\frac{3}{140} + \frac{2}{140}\log_2\frac{2}{140}\right)$$

$$\approx -(-0.1717 - 0.3467 - 0.4762 - 0.1188 - 0.0876)$$

$$\boxed{\approx 1.2}$$

(b) Top drawer = nails + zip ties $= 100 + 30 = 130$

Bottom drawer = rest $= 5+3+2 = 10$

$$H(X|Y) = -\left[P(top)\cdot\sum_{X=x} P(X=x|Y=top)\log_2 P(X=x|Y=top) + P(bottom)\sum_{X=x} P(X=x|Y=bottom)\log_2 P(X=x|Y=bottom)\right]$$

$$= -\left[0.5\left(\frac{100}{130}\log_2\frac{100}{130} + \frac{30}{130}\log_2\frac{30}{130}\right) + 0.5\left(\frac{5}{10}\log_2\frac{5}{10} + \frac{3}{10}\log_2\frac{3}{10} + \frac{2}{10}\log_2\frac{2}{10}\right)\right]$$

$$\approx -[0.5\cdot(-0.78) + 0.5\cdot(-1.48)]$$

$$\boxed{\approx 1.13}$$

(c) $I(X;Y) = 1.2 - 1.13 = \boxed{0.07}$

(d) Top drawer = phillips + flathead + $\frac{1}{2}\cdot$ nails $= 2+3+\frac{1}{2}\cdot 100 = 55$

Bottom drawer = rest $= 5 + 30 + \frac{1}{2}\cdot 100 = 85$

$$H(X|Y) = -\left[0.5\left(\frac{2}{55}\log_2\frac{2}{55} + \frac{3}{55}\log_2\frac{3}{55} + \frac{50}{55}\log_2\frac{50}{55}\right) + 0.5\left(\frac{5}{85}\log_2\frac{5}{85} + \frac{30}{85}\log_2\frac{30}{85} + \frac{50}{85}\log_2\frac{50}{85}\right)\right]$$

$$\approx -[0.5(-0.174 - 0.229 - 0.125) + 0.5(-0.24 - 0.53 - 0.45)]$$

$$\approx -[0.5\cdot(-1.748)]$$

$$\approx 0.874$$

$$\boxed{I(X;Y) = 1.2 - 0.874 = 0.326}$$

This one is give the largest information gain.

3. (a) i. $y_i = x + z_i = x + N(0,1)$

$N(0,1) = y_i - x$

$P(y_i|x) = \frac{1}{\sqrt{2\pi}} \cdot \exp[-(y_i - x)^2/2]$

$P(y_1, \dots y_m|x) = \frac{1}{\sqrt{2\pi}} \prod_{i=1}^{m} \exp[-(y_i - x)^2/2]$

log-likelihood: $\log P(y_1 \dots y_m|x) = \log(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}}) + \sum_{i=1}^{m} (\frac{-(y_i - x)^2}{2})$

$= \sum_{i=1}^{m} \log((2\pi)^{-\frac{1}{2}}) - \frac{1}{2} \sum_{i=1}^{m} (y_i - x)^2$

$= -\frac{m}{2} \log(2\pi) - \frac{1}{2} \sum_{i=1}^{m} (y_i - x)^2$

MLE for x: $\frac{d \log p(x_1 \dots x_m|x)}{dx} = \sum_{i=1}^{m} (y_i - x)$

$0 = \sum_{i=1}^{m} (y_i - x)$

$0 = \sum_{i=1}^{m} y_i - mx$

$\boxed{\hat{x} = \frac{1}{m} \sum_{i=1}^{m} y_i}$

ii. Bias: $E[\hat{x}] = E[\frac{1}{m} \sum_{i=1}^{m} y_i] = \frac{1}{m} \sum_{i=1}^{m} E[y_i] = x$, $\boxed{Bias = E[\hat{x}] - x = 0}$

Variance: $Var(\hat{x}) = Var(\frac{1}{m} \sum_{i=1}^{m} y_i) = \frac{1}{m^2} \sum_{i=1}^{m} Var(y_i)$

since $y_i = x + N(0,1)$, $x \in R$, $z \sim N(0,1)$. $Var(y_i) = 1$

$Var(\hat{x}) = \frac{1}{m^2} \sum_{i=1}^{m} 1 = \boxed{\frac{1}{m}}$

iii. Bias: $Bias(\hat{x})$ is unbias, so $\boxed{Bias(\hat{x}) = 0 \text{ as } m \to +\infty}$

Variance: $\boxed{Var(\hat{x}) = \frac{1}{m} \to 0 \text{ as } m \to +\infty}$

(b) i. $$\frac{d\left(\frac{1}{m}\sum_{i=1}^{m}(y_i-x)^2+p(x-\bar{x})^2\right)}{dx}=0$$

$$-\frac{2}{m}\sum_{i=1}^{m}(y_i-x)+2p(x-\bar{x})=0$$

$$-\frac{2}{m}\sum_{i=1}^{m}y_i+2x+2px-2p\bar{x}=0$$

$$(2+2p)x=\frac{2}{m}\sum_{i=1}^{m}y_i+2p\bar{x}$$

$$\boxed{x_{MAP}=\frac{1}{1+p}\cdot\left(\frac{1}{m}\sum_{i=1}^{m}y_i+p\bar{x}\right)}$$

ii. Bias: $$E[x_{MAP}]=E\left[\frac{1}{1+p}\cdot\left(\frac{1}{m}\sum_{i=1}^{m}y_i+p\bar{x}\right)\right]$$

$$=\frac{1}{1+p}\cdot\left(E\left[\frac{1}{m}\sum_{i=1}^{m}y_i\right]+p\bar{x}\right)$$

$$=\frac{1}{1+p}\cdot\left(E[\hat{x}]+p\bar{x}\right)$$

$$=\frac{1}{1+p}\cdot(x+p\bar{x})$$

$$\boxed{Bias(x_{MAP})=E[x_{MAP}]-x=\frac{x+p\bar{x}}{1+p}-\frac{x+px}{1+p}=\frac{p(\bar{x}-x)}{1+p}}$$

Variance: $$Var(x_{MAP})=Var\left(\frac{1}{1+p}\cdot\left(\frac{1}{m}\sum_{i=1}^{m}y_i+p\bar{x}\right)\right)$$

$$=\frac{1}{(1+p)^2}\cdot Var\left(\frac{1}{m}\sum_{i=1}^{m}y_i\right)$$

$$=\frac{1}{(1+p)^2}\cdot Var(\hat{x})$$

$$=\frac{1}{(1+p)^2}\cdot\frac{1}{m}$$

$$\boxed{=\frac{1}{m(1+p)^2}}$$

iii. Bias: $Bias(x_{MAP})\neq\bar{x}$, it's bias

$$\boxed{Bias(x_{MAP})=\frac{p(\bar{x}-x)}{1+p}\text{ as } m\to+\infty}$$

Variance: $$\boxed{Var(x_{MAP})=\frac{1}{m(1+p)^2}\to 0\text{ as } m\to+\infty}$$

(c). $MSE = E[(x-\hat{x})^2] = B^2 + V$

$$= \left(\frac{p(\bar{x}-x)}{1+p}\right)^2 + \frac{1}{m(1+p)^2}$$

$B = \frac{1}{1+p} \quad p = \frac{1}{B} - 1$

$$MSE = \left(\frac{1-B}{B} \cdot \frac{1}{1+\frac{1}{B}-1} \cdot (\bar{x}-x)\right)^2 + \frac{1}{m(1+\frac{1}{B}-1)^2}$$

$$= ((1-B)\cdot(\bar{x}-x))^2 + \frac{B^2}{m}$$

$$= (1-B)^2\Delta^2 + \frac{B^2}{m}$$

minimize the MSE:

$$\frac{d(MSE)}{dB} = -2(1-B)\Delta^2 + \frac{2B}{m}$$

$$0 = -2(1-B)\Delta^2 + \frac{2B}{m}$$

$$0 = -2\Delta^2 + 2B\Delta^2 + \frac{2B}{m}$$

$$2\Delta^2 = 2B(\Delta^2 + \frac{1}{m})$$

$$B = \frac{2\Delta^2}{2(\Delta^2+\frac{1}{m})} = \frac{\Delta^2}{\Delta^2+\frac{1}{m}}$$

$$p = \frac{1}{B} - 1 = \frac{\Delta^2+\frac{1}{m}}{\Delta^2} - \frac{\Delta^2}{\Delta^2} = \frac{\frac{1}{m}}{\Delta^2} \boxed{= \frac{1}{m\Delta^2}}$$

4. (a).

$$\min_{\theta} \|X\theta - y\|_2^2 + \rho\|\theta\|_2^2$$

$$\frac{d(\|X\theta - y\|_2^2 + \rho\|\theta\|_2^2)}{d\theta} = 2X^T(X\theta - y) + 2\rho\theta$$

$$0 = 2X^T(X\theta - y) + 2\rho\theta$$

$$0 = 2X^TX\theta - 2X^Ty + 2\rho\theta$$

$$\theta(2X^TX + 2\rho I) = 2X^Ty$$

$$\theta = (2X^TX + 2\rho I)^{-1} 2X^Ty$$

$$\boxed{\theta = (X^TX + \rho I)^{-1} X^Ty}$$

(b).

$$y_i = x_i^T\bar{\theta} + z_i, \quad z_i \sim N(0,1)$$

$$E[\theta] = E[(X^TX + \rho I)^{-1}X^Ty]$$

$$= E[(X^TX + \rho I)^{-1}X^T(X^T\bar{\theta} + z)]$$

$$= E[(X^TX + \rho I)^{-1}X^TX\bar{\theta} + (X^TX + \rho I)^{-1}X^Tz]$$

$$= (X^TX + \rho I)^{-1}X^TX\bar{\theta} + (X^TX + \rho I)^{-1}X^TE[z]$$

Since $z_i \sim N(0,1)$, $E[z] = 0$

$$\boxed{E[\theta] = (X^TX + \rho I)^{-1}X^TX\bar{\theta}}$$

(c)

$$\text{Bias}(\theta) = E[\theta] - \bar{\theta}$$

$$\boxed{= ((X^TX + \rho I)^{-1}X^TX - I)\bar{\theta}}$$

(d)

$$\Sigma = \text{Cov}(\theta) = E[(\theta - E[\theta])(\theta - E[\theta])^T]$$

$$= E[((X^TX + \rho I)^{-1}X^Tz)((X^TX + \rho I)^{-1}X^Tz)^T]$$

$$= E[(X^TX + \rho I)^{-1}X^Tzz^TX(X^TX + \rho I)^{-1}]$$

$$= (X^TX + \rho I)^{-1}X^TE[zz^T]X(X^TX + \rho I)^{-1}$$

Since $z_i \sim N(0,1)$, $E[zz^T] = 1$

$$\boxed{\Sigma = (X^TX + \rho I)^{-1}X^TX(X^TX + \rho I)^{-1}}$$

(e). Bias:

$$\text{Bias}(\theta) = ((X^TX + \rho I)^{-1}X^TX - I)\bar{\theta}$$

As $\rho \to 0$: $\text{Bias}(\theta) \approx (I - I)\bar{\theta} \approx 0$ } Bias(θ) reduce

As $\rho \to \infty$: $\text{Bias}(\theta) \approx (0 - I)\bar{\theta} \approx -\bar{\theta}$

As $m \to \infty$: $\text{Bias}(\theta) \approx (I - I)\bar{\theta} \approx 0$ } Bia(θ) reduce

$$\boxed{\|\text{Bias}(\theta)\|_2 = O\left(\frac{\rho}{\rho + m}\right)}$$

Σ:

$$\Sigma = (X^TX + \rho I)^{-1}X^TX(X^TX + \rho I)^{-1}$$

As $\rho \to 0$: $\Sigma \approx (X^TX)^{-1}$ | As $\rho \to \infty$: $\Sigma \approx 0$ } Σ reduce

As $n \to \infty$: $\Sigma \approx (X^TX)^{-1} \approx 0$ } Σ reduce

$$\boxed{\|\Sigma\|_2 = O\left(\frac{1}{\rho + m}\right)}$$

5. (a) i.

$$L(\theta, s; a, u) = \frac{1}{2}\|\theta\|_2^2 + p\sum_{i=1}^{m} s_i - \sum_{i=1}^{m} u_i (y_i \phi(x_i)^T\theta - 1 + s_i) - \sum_{i=1}^{m} v_i s_i$$

$$\boxed{= \frac{1}{2}\|\theta\|_2^2 + ps^T 1 - \sum_{i=1}^{m} u_i (y_i \phi(x_i)^T\theta - 1 + s_i) - \sum_{i=1}^{m} v_i s_i}$$

ii.

$$\frac{dL(\theta,s,a,u)}{d\theta} = \theta - \sum_{i=1}^{m} u_i y_i \phi(x_i)$$

$$0 = \theta - \sum_{i=1}^{m} u_i y_i \phi(x_i)$$

$$\theta = \sum_{i=1}^{m} u_i y_i \phi(x_i)$$

$$L(\theta; u, v) = \frac{1}{2}\|\sum_{i=1}^{m} u_i y_i \phi(x_i)\|_2^2 - \sum_{i=1}^{m} u_i y_i \phi(x_i)^T \sum_{j=1}^{m} u_j y_j \phi(x_j) + \sum_{i=1}^{m} u_i$$

$$= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j y_i y_j \phi(x_i)^T\phi(x_j) - \sum_{i=1}^{m} u_i u_j y_i y_j \phi(x_i)^T\phi(x_j) + \sum_{i=1}^{m} u_i$$

$$= -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j y_i y_j \phi(x_i)^T\phi(x_j) + u^T 1$$

$$\boxed{= -\frac{1}{2}\|\sum_{i=1}^{m} u_i y_i \phi(x_i)\|_2^2 + u^T 1}$$

$$\frac{dL(\theta,s,a,u)}{ds} = p - \sum_{i=1}^{m} u_i - \sum_{i=1}^{m} v_i$$

$$0 = p - u - v$$

$$p = u + v$$

Since $u \geq 0$ and $v \geq 0$ and $p = u + v$, get $\boxed{0 \leq u \leq p}$

$$\boxed{\begin{array}{ll} \text{Dual:} & \underset{u,v}{\text{maximize}} \ -\frac{1}{2}\|\sum_{i=1}^{m} u_i y_i \phi(x_i)\|_2^2 + u^T 1 \\ & \text{subject to} \quad 0 \leq u \leq p \end{array}}$$

iii.

$$\boxed{\begin{array}{ll} \text{Dual:} & \underset{u,v}{\text{maximize}} \ -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} u_i u_j y_i y_j K_{i,j} \\ & \text{subject to} \quad 0 \leq u \leq p \end{array}}$$

iv.

$$\boxed{\nabla_u \text{Dual} = \left(-\sum_{j=1}^{m} u_j y_i y_j K_{i,j} + 1\right)_{i=1}^{m}}$$

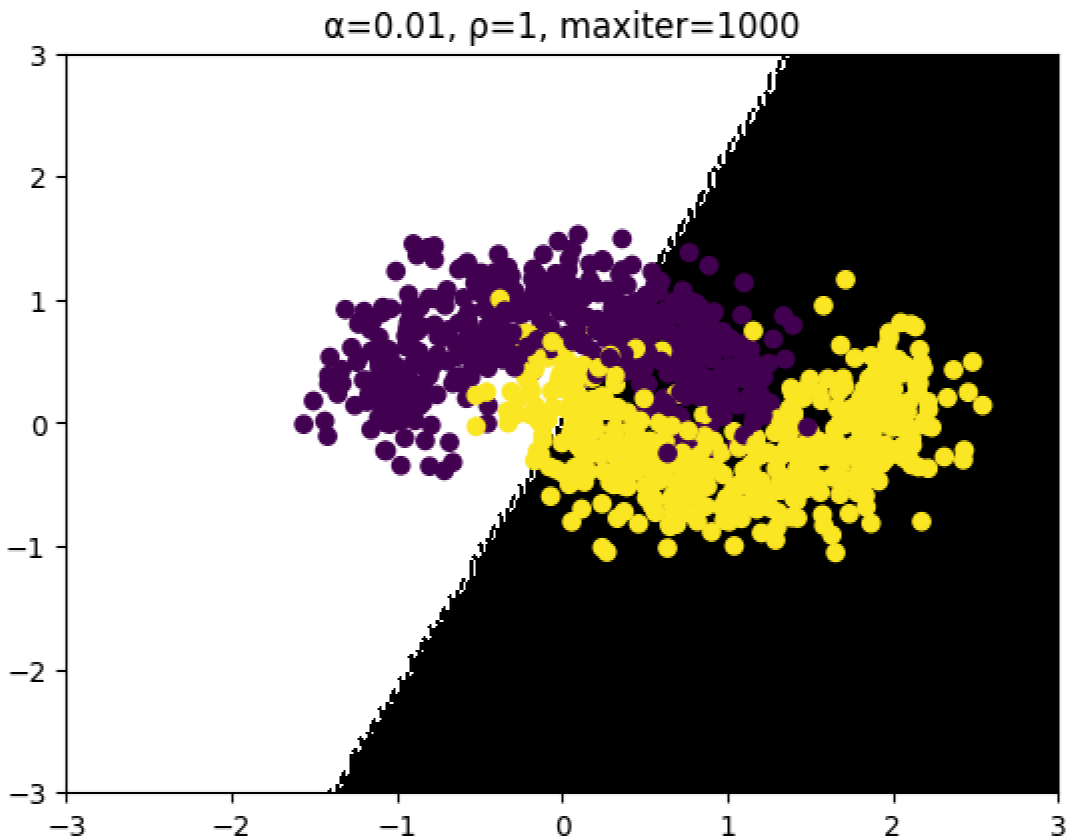
(b).

```
def fit_data(X,y,Kfun,rho,maxiter=1000):
    K = Kfun(X,X)
    def get_grad(u):
        return np.ones_like(u) - (y * np.dot(u * y, K))

    u = np.ones(X.shape[0])
    for _ in range(maxiter):
        u = np.clip(u - 0.01 * get_grad(u), 0, rho)

    def inference(x):
        Kt = Kfun(x,X)
        return np.sign(np.dot(y * u, Kt.T))
    return inference

def LinearKfun(X1,X2):
    return np.dot(X1,X2.T)

rhos = [0.01, 0.1, 1, 10]
for rho in rhos:
    inference = fit_data(X, y, LinearKfun, rho)
    print(f"rho = {rho}, Train error: {np.mean(inference(X) != y):.4f}, 
Test error: {np.mean(inference(Xt) != yt):.4f}")
```

```
rho = 0.01, Train error: 0.2580, Test error: 0.2900
rho = 0.1, Train error: 0.2540, Test error: 0.2900
rho = 1, Train error: 0.2530, Test error: 0.2900
rho = 10, Train error: 0.2530, Test error: 0.2900
```

(c).

```
inference = fit_data(X, y, LinearKfun, 1)
Xp, Yp, Zp = plot_model_contour(inference)
plt.contourf(Xp, Yp, Zp, colors=['#FFFFFF', '#000000'])
plt.scatter(*X.T, c=y)
plt.title("α=0.01, ρ=1, maxiter=1000")
plt.show()
```



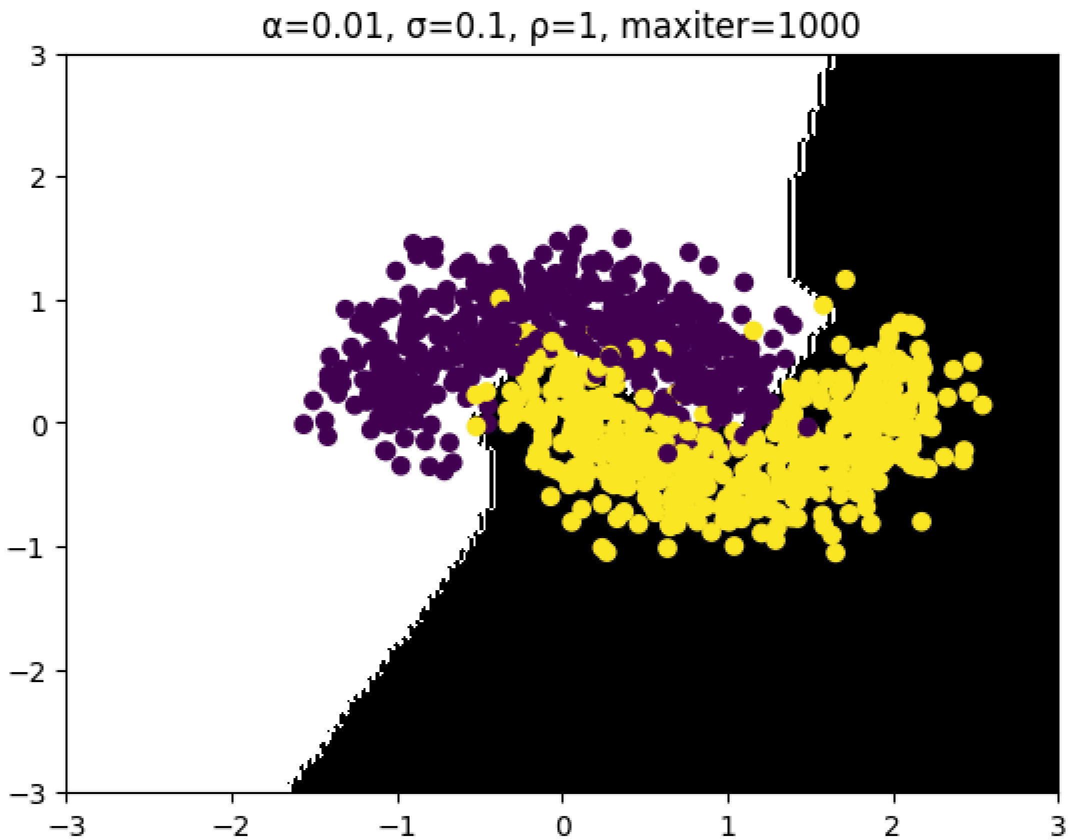
(d).

```
def fit_data(X,y,Kfun,sigma,rho,maxiter=1000):
   K = Kfun(X,X,sigma)
   def get_grad(u):
       return np.ones_like(u) - (y * np.dot(u * y, K))

   u = np.ones(X.shape[0])
   for _ in range(maxiter):
       u = np.clip(u - 0.01 * get_grad(u), 0, rho)

   def inference(x):
       Kt = Kfun(x,X,sigma)
       return np.sign(np.dot(y * u, Kt.T))
   return inference

def RBFKfun(X1,X2,sigma):
   sq = np.sum(X1**2, axis=1).reshape(-1, 1) - 2 * np.dot(X1, X2.T) +
np.sum(X2**2, axis=1)
   return np.exp(- sq / (2 * sigma**2))

sigmas = [1, 0.1, 0.001, 0.0001]
for sigma in sigmas:
   inference = fit_data(X, y, RBFKfun, sigma, 1)
   print(f"Sigma: {sigma}, Train error: {np.mean(inference(X) != y):.4f},
Test error: {np.mean(inference(Xt) != yt):.4f}")
```

```
Sigma: 1, Train error: 0.1990, Test error: 0.2200
Sigma: 0.1, Train error: 0.0520, Test error: 0.0400
Sigma: 0.001, Train error: 0.0020, Test error: 0.5200
Sigma: 0.0001, Train error: 0.0000, Test error: 0.9700
```

(e).

```
inference = fit_data(X, y, RBFKfun, 0.1, 1)
Xp, Yp, Zp = plot_model_contour(inference)
plt.contourf(Xp, Yp, Zp, colors=['#FFFFFF', '#000000'])
plt.scatter(*X.T, c=y)
plt.title("α=0.01, σ=0.1, ρ=1, maxiter=1000")
plt.show()
```



(f).

When σ -> 0, SVM is too sensitive to a single data point, causing the model to fit the training data very closely but lose generalization ability, leading to overfitting. This is why although Train error -> 0, the Test error is getting bigger and bigger.