Homework 5, Fall 2024

Mohammad Alshurbaji

11/27/2024

Problem 1: Vanilla Logistic Regression for multi Classification:

- k classes. - The input $x \in R^d$

- Prob. to each class is:

$$P(Y=k \mid X=x) = \frac{\exp(w_k^T x)}{1 + \sum_{l=1}^{k-1} \exp(w_l^T x)}, \text{ for } k = 1, 2, \dots, k-1$$

$$P(Y=k \mid X=x) = \frac{1}{1 + \sum_{l=1}^{k-1} \exp(w_l^T x)}$$

If we define $w_k = 0$, Then:

$$P(Y=k \mid X=x) = \frac{\exp(w_k^T x)}{1 + \sum_{l=1}^{k-1} \exp(w_l^T x)}, k = 1, \dots, k$$

Question $\boxed{1}$ what and how many Para. are there to be opt. ?

1. Weight vectors (w_k)
   - → For each class k, there is $w_k \in R^d$.
   - → assuming $w_k = 0$.
   - → means: The # of weight Para.:
     $d \times (k-1)$
2. Bias Parameters: b_k
   - → one bias for each class: $(k-1)$

∴ Total Parameters: $d \times (k-1) + (k-1)$

$= (d+1)(k-1)$ #

Question $\boxed{2}$ $L(w_1, \dots, w_{k-1}) = \sum_{i=1}^{n} \ln P(Y = y_i \mid X = x_i)$

$$= \sum_{i=1}^{n} \ln\left(\frac{\exp(w_k^T x)}{1 + \sum_{l=1}^{k-1} \exp(w_l^T x)} \right)$$

$$= \sum_{i=1}^{n} \left(\ln(\exp(w_k^T x)) - \ln\left(1 + \sum_{l=1}^{k-1} \exp(w_l^T x)\right)\right)$$

$$= \sum_{i=1}^{n} \Big(\underbrace{w_k^T x}_{\mathrm{I}} - \underbrace{\ln\left(1 + \sum_{l=1}^{k-1} \exp(w_l^T x)\right)}_{\mathrm{II}} \Big)$$

Question $\boxed{3}$ The gradient of L w.r.t. w_k:

$$\frac{\partial L}{\partial w_k} = \cancel{\sum_{i=1}^{k}}$$

$\underline{\mathrm{I}}$: $\frac{\partial \mathrm{I}}{\partial w_k} = x$

$\underline{\mathrm{II}}$: $\frac{\partial \mathrm{II}}{\partial w_k} = \frac{\exp(w_l^T x)\, x}{1 + \sum_{l=1}^{k-1} \exp(w_l^T x)}$

$\underline{\therefore \mathrm{I} + \mathrm{II}}$: $\frac{\partial L}{\partial w_k} = \sum_{i=1}^{n} \left(x - \left(\frac{\exp(w_l^T x)\, x}{1 + \sum_{l=1}^{k-1} \exp(w_l^T x)} \right)\right)$

$$= \sum_{i=1}^{n} \left(x - P(Y = k \mid X = x_i)\, x \right)$$

$$= \sum_{i=1}^{n} \left(\left(1(y = k) - P(Y = k \mid X = x_i) \right) x \right)$$ # (In terms of probability)

Question $\boxed{4}$: With adding the regularization term:

$$f(w_1, \dots, w_{k-1}) = \underbrace{L(w_1, \dots, w_{k-1})}_{\mathrm{I}} \underbrace{- \frac{\lambda}{2} \sum_{l=1}^{k-1} \|w_l\|_2^2}_{\mathrm{II}}$$

$\frac{\partial f}{\partial w_k}$:

$$\mathrm{I}: \quad \frac{\partial \mathrm{I}}{\partial w_k} = \sum_{i=1}^{n} \left(\left(1 - P\left(Y = k \mid X = x_i\right)\right) X \right)$$

$$\mathrm{II}: \quad \frac{\partial \mathrm{II}}{\partial w_k} = -\lambda w_k$$

$$\mathrm{I} + \mathrm{II}: \quad \frac{\partial f}{\partial w_k} = \sum_{i=1}^{n} \left(\left(1 - P\left(Y = k \mid X = x_i\right)\right) X \right) - \lambda w_k$$

#

Question $\boxed{5}$: USPS Handwritten recognition digit dataset.

→ Image size: 16×16

→ For each digit (i.e. $0, 1, \dots, 9$) there are 600 training samples. 500 testing ones.

(a)

⇒ Continue in the next pages.

Question 4:

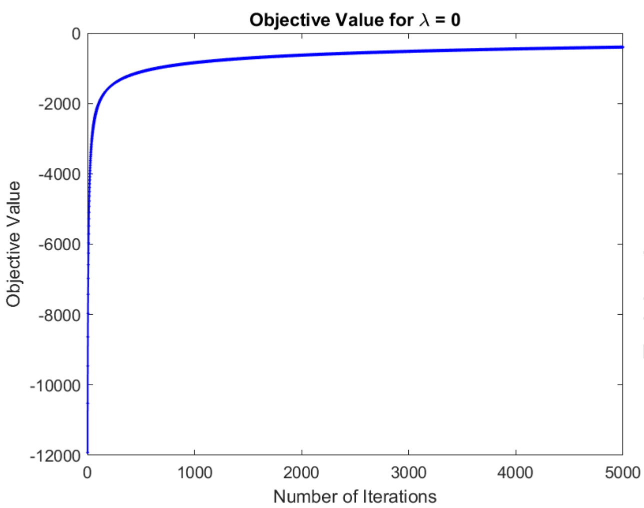
Log_grad.m code:”

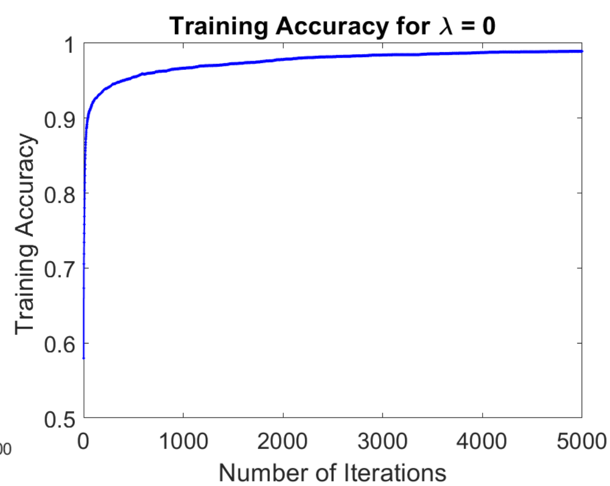
```
function G=log_grad(y, X, B)
    [n,d] = size(X);%n: number of samples, d: number of features
    K = size(B,2) + 1; %Total number of classes

%compute gradient
    XB = X * B;
    expXB = exp(XB);
    prob = expXB ./ (1 + sum(expXB, 2));

    prob = [prob, 1 - sum(prob, 2)];

    G = zeros(d,K-1);
    for k = 1:K-1
        indicator = (y == k);  % Indicator vector for class k
        G(:, k) = X' * (indicator - prob(:, k));  % Gradient for class k
    end


end
```
“

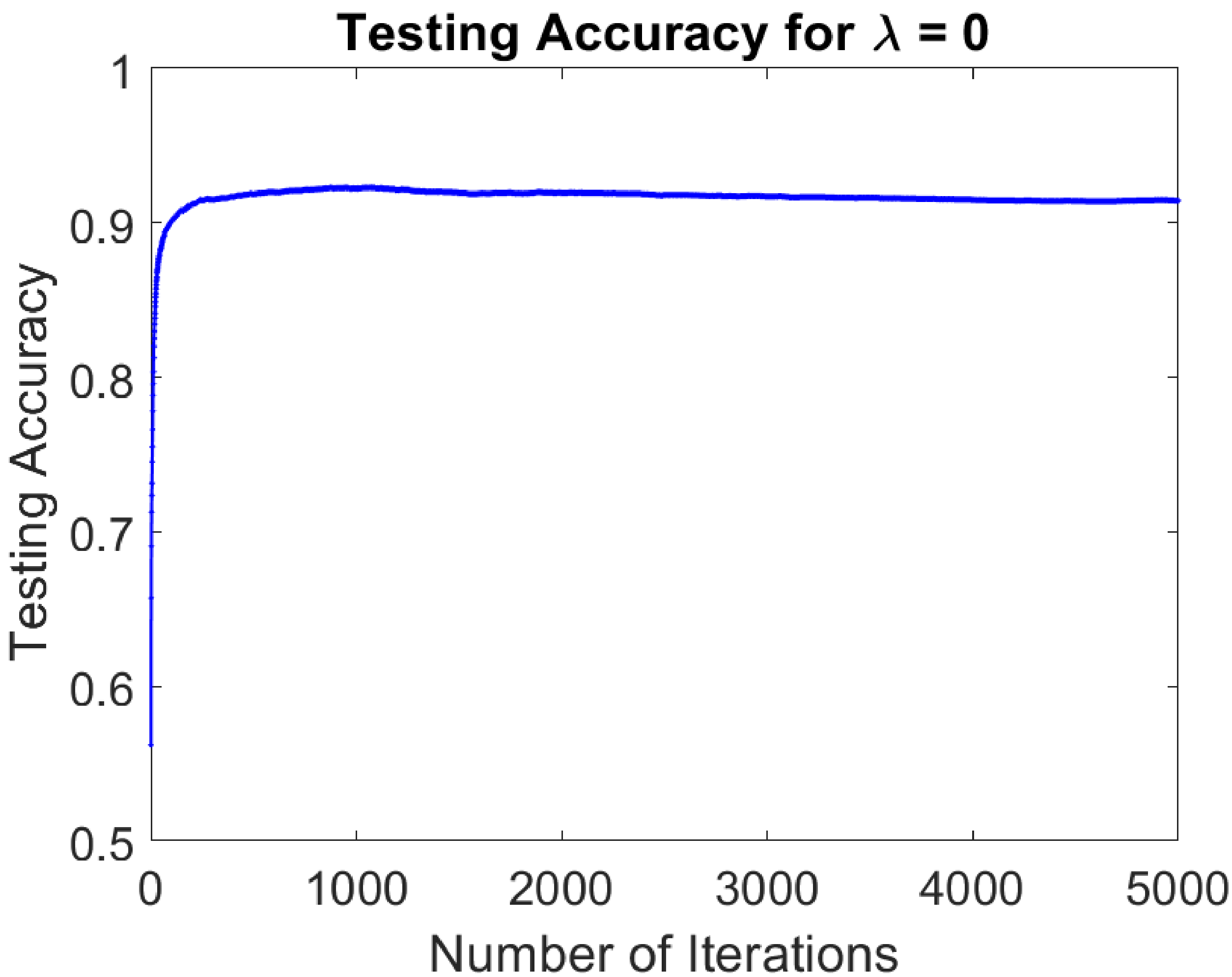
```
function G=log_grad(y, X, B)
    [n,d] = size(X);%n: number of samples, d: number of features
    K = size(B,2) + 1; %Total number of classes

%compute gradient
    XB = X * B;
    expXB = exp(XB);
    prob = expXB ./ (1 + sum(expXB, 2));

    prob = [prob, 1 - sum(prob, 2)];

    G = zeros(d,K-1);
    for k = 1:K-1
        indicator = (y == k);  % Indicator vector for class k
        G(:, k) = X' * (indicator - prob(:, k));  % Gradient for class k
    end


end
```

Following this code and implementing the logistic_classify .m and generate these graphs:

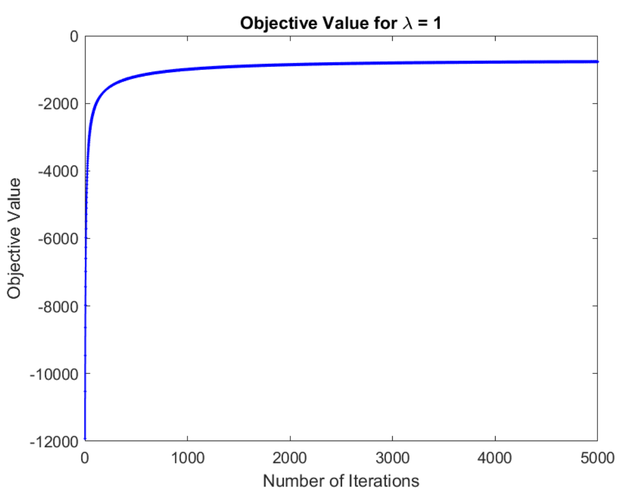
When lambda = 0

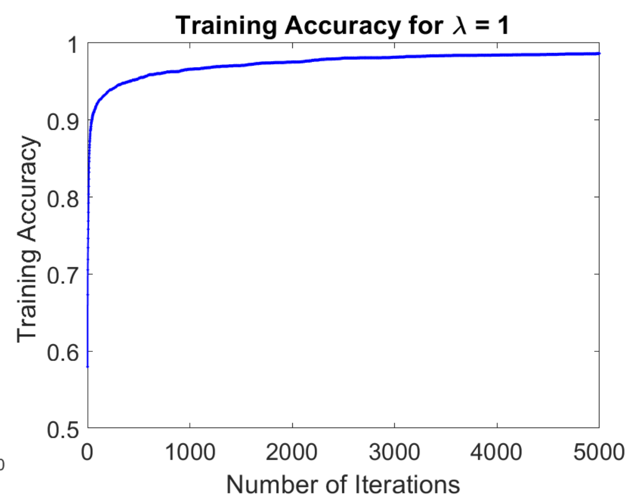


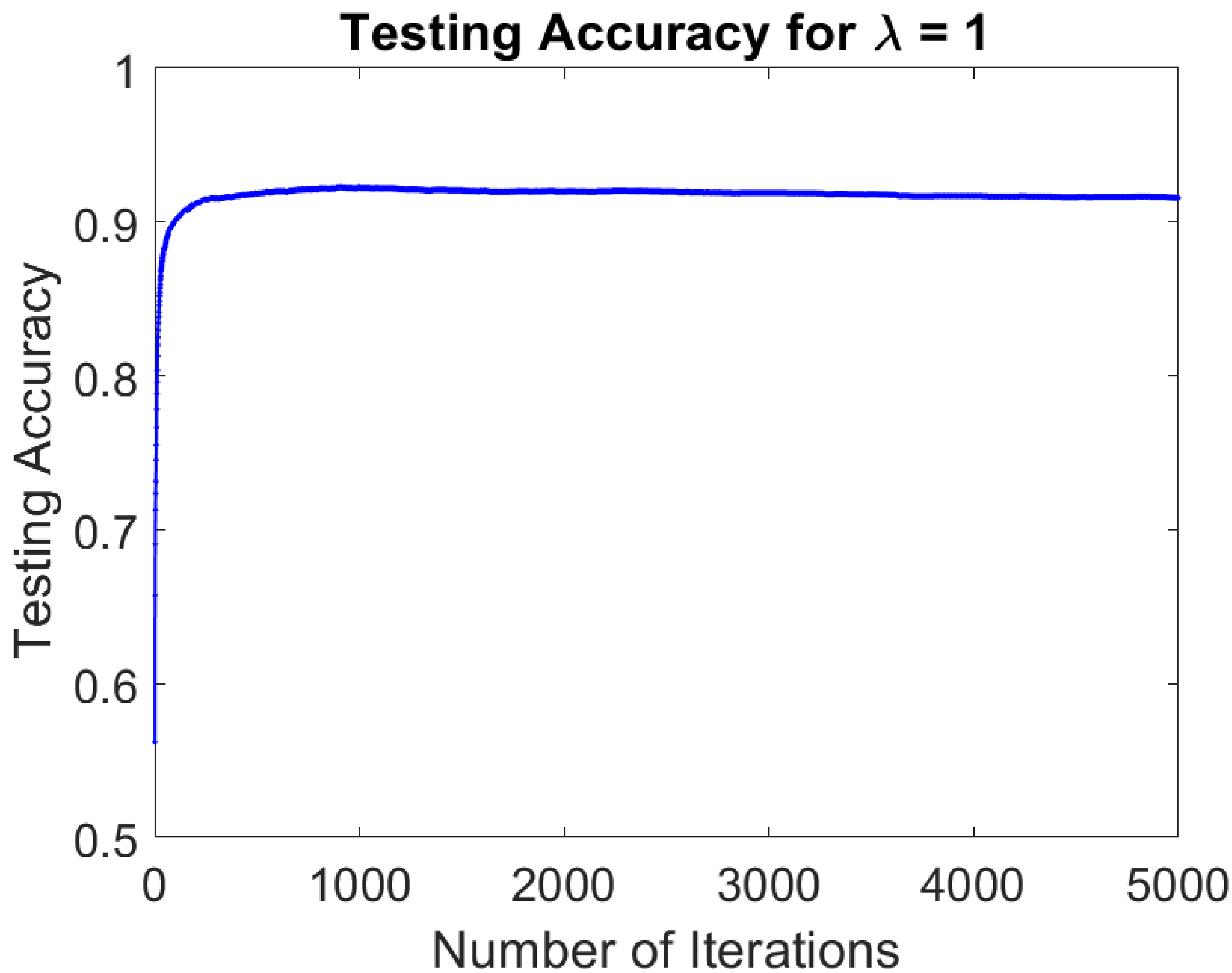




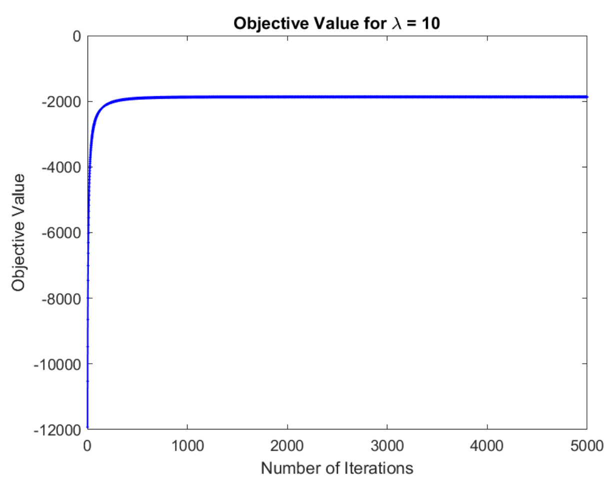
When lambda = 1

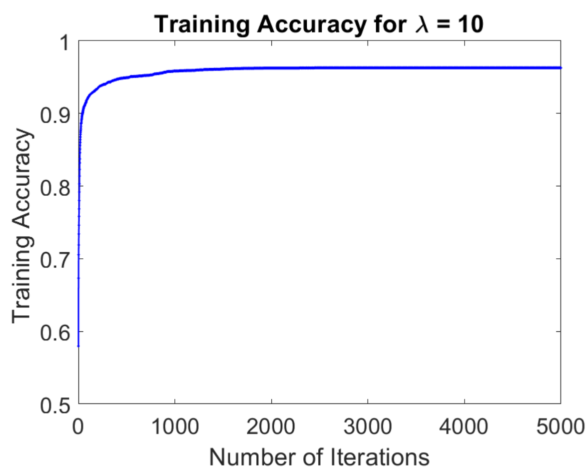


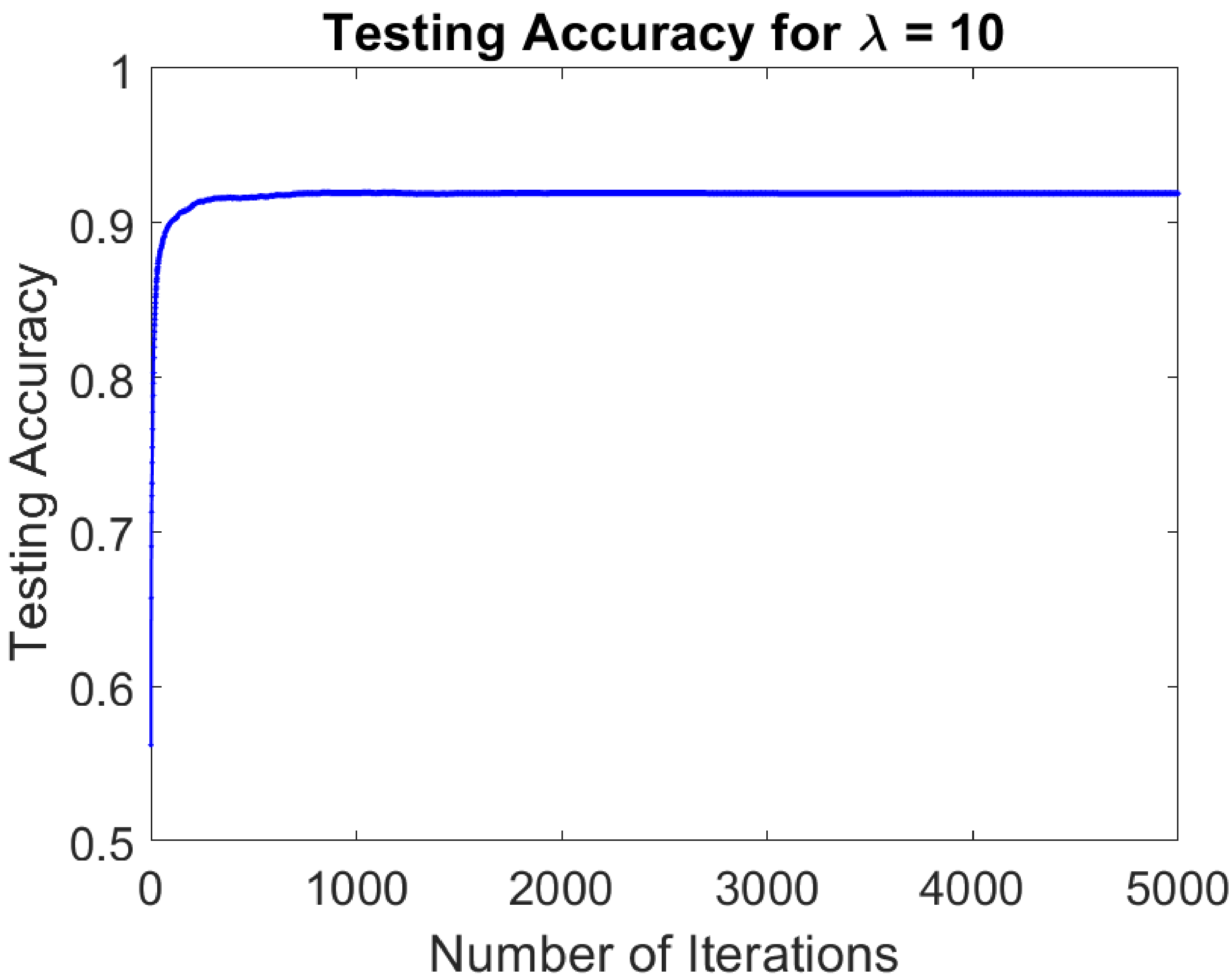




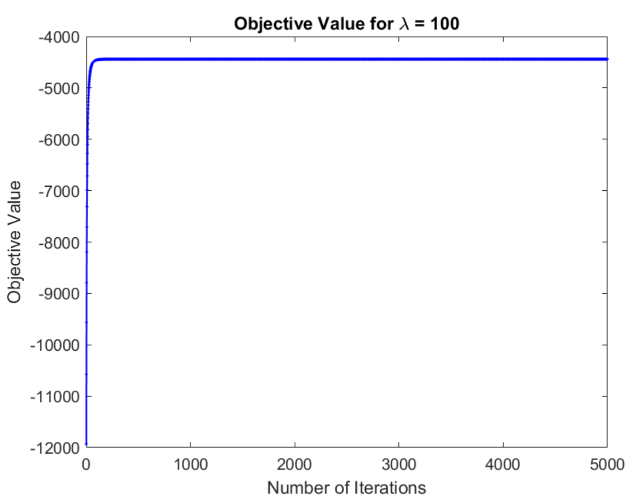
When lambda = 10

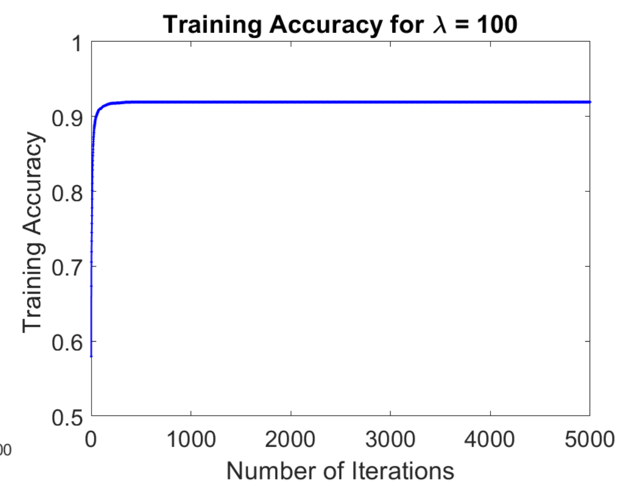


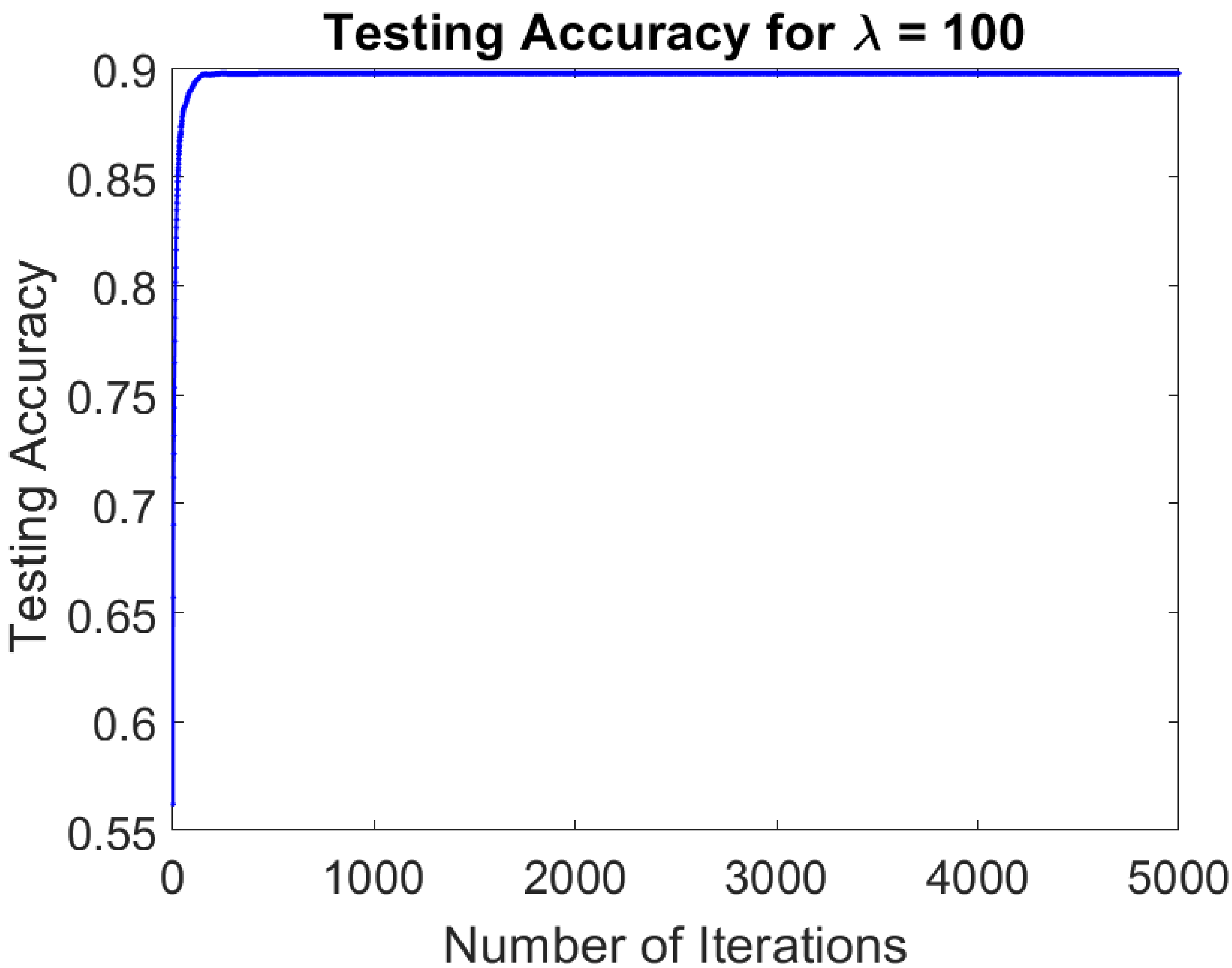




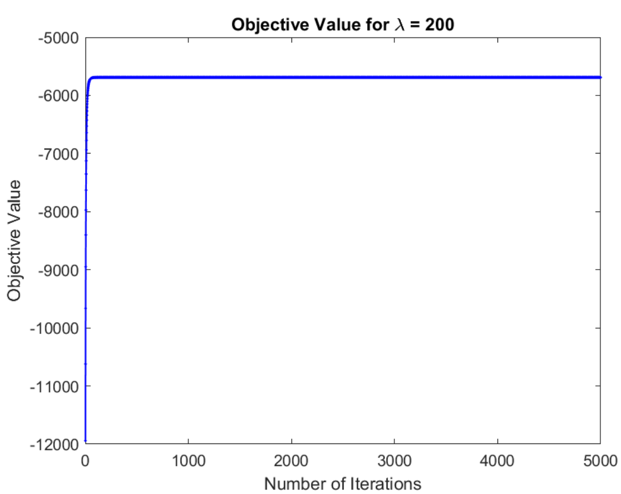
When lambda =100

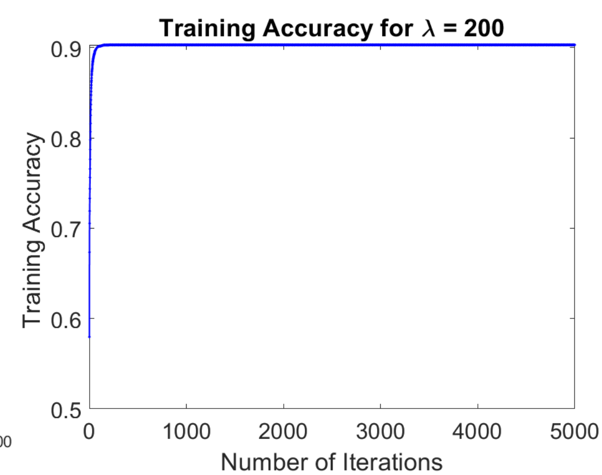


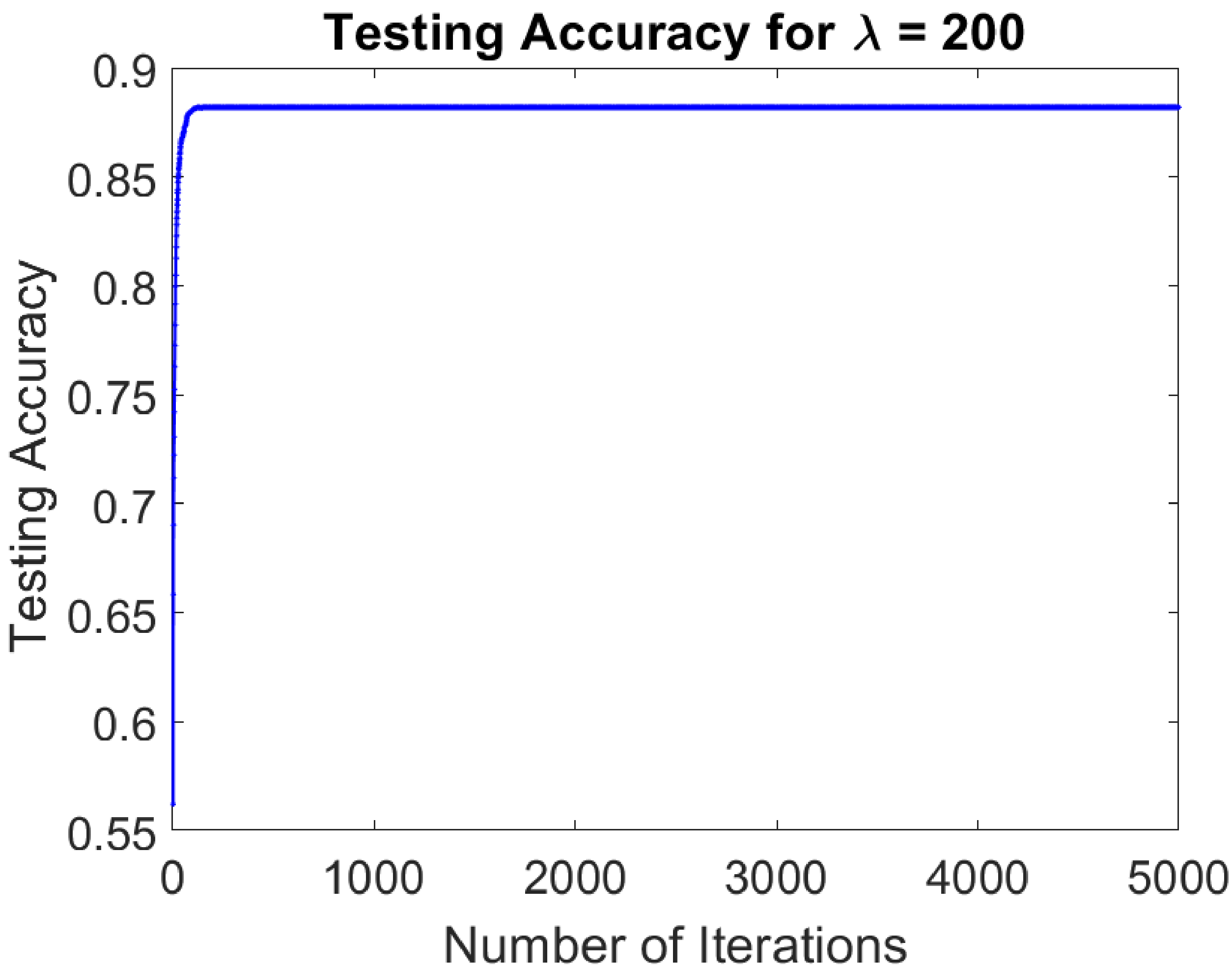




When lambda =200







From the above charts we recognize that:

- Objective value:
  - In general the obj value decreases over iterations, showing convergence for all lambda values.
  - While for smaller lambda values (lambda = 0), the convergence is faster and reach a lower final value.
  - And for larger values, the convergence is slower and the final obj is higher.
  - To wrap up: a moderate lambda balances fast convergence and regularization.

- Training Accuracy:
  - For smaller lambda values, training accuracy is higher as the model overfits to the training data. While for larger values, training accuracy decreases because the model becomes over regularized.
  - To wrap up, moderate lambda has in between better training accuracy.
- Testing Accuracy:
  - This improves with moderate values as the model geralization better to unseen data.

- While for small values, testing accuracy is lower since the model overfits to the training sets.
- And for larger values, it declines again due to underfitting since the model becomes too simplistic.