ELL409 Assignment 2

Pratyush Pandey - 2017EE10938

Developing the classifier using cvxopt library:

The SVM optimization problem is:

$$\min_{w,\xi} \ \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad 1 - \xi_i - y_i (w^T x_i + b) \leq 0 \quad \forall \ i = 1, \ldots, n$$

$$-\xi_i \leq 0$$

To solve this, we make the Lagrangian and apply KKT condition,

Lagrangian

$$L(w, \xi, \mu, \lambda) = \frac{1}{2} w^T w + C \sum \xi_i + \sum \mu_i (1 - \xi_i - y_i (w^T x_i + b)) - \sum \lambda_i \xi_i$$

first set, second set $\{\xi_i\}$

Apply KKT

1) $\nabla_w L = 0 \Rightarrow w + \sum \mu_i (-y_i x_i) = 0 \Rightarrow w^* = \sum_{i=1}^{n} \mu_i y_i x_i$

2) $\nabla_b L = 0 \Rightarrow -\sum \mu_i y_i = 0 \Rightarrow \sum_{i=1}^{n} \mu_i y_i = 0$

3) $\nabla_\xi L = 0 \Rightarrow C - \mu_i - \lambda_i = 0 \ \forall i \Rightarrow \mu_i + \lambda_i = C \ \forall \ i = 1, \ldots n$

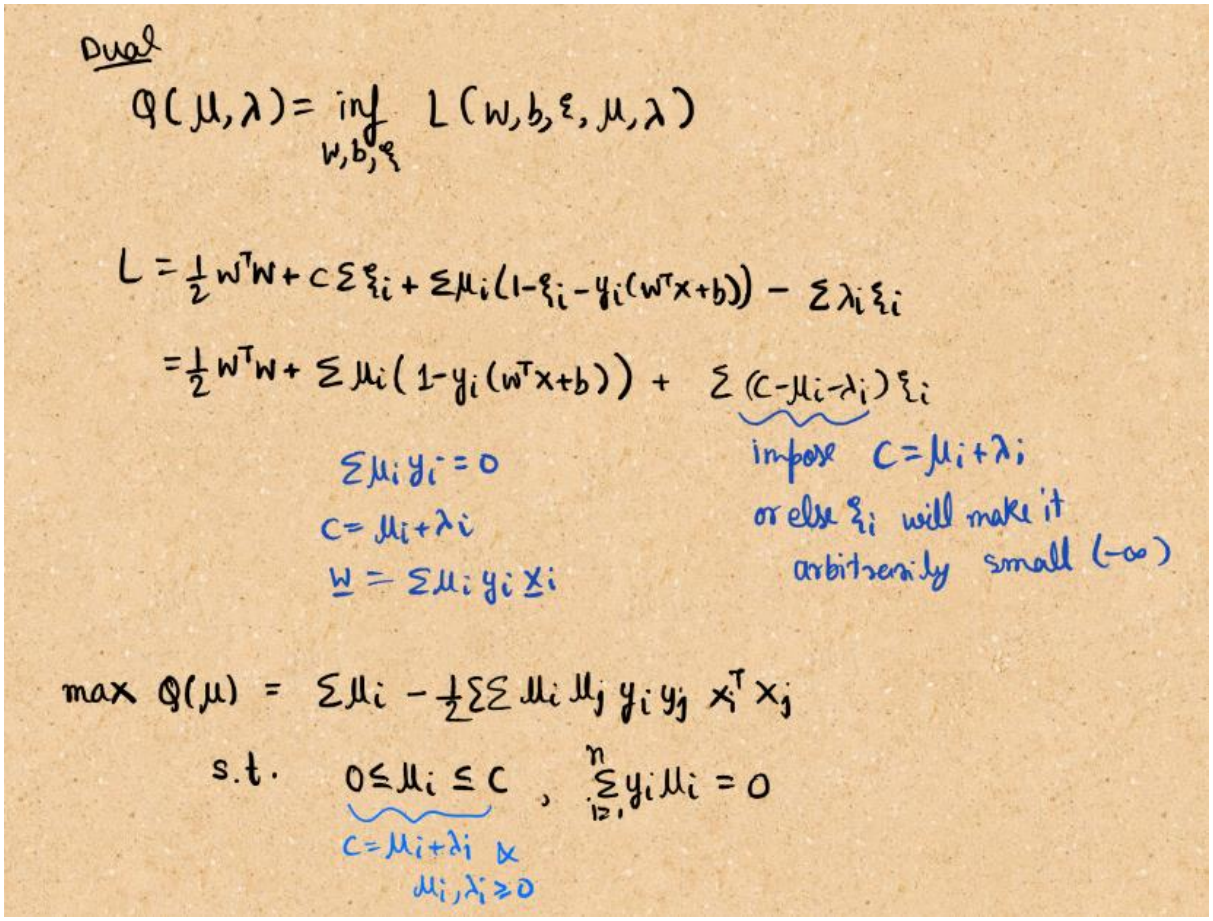
4) $\xi_i \geq 0$, $1 - \xi_i - y_i (w^T x_i + b) \leq 0$

5) $\mu_i , \lambda_i \geq 0 \quad \forall \ i$

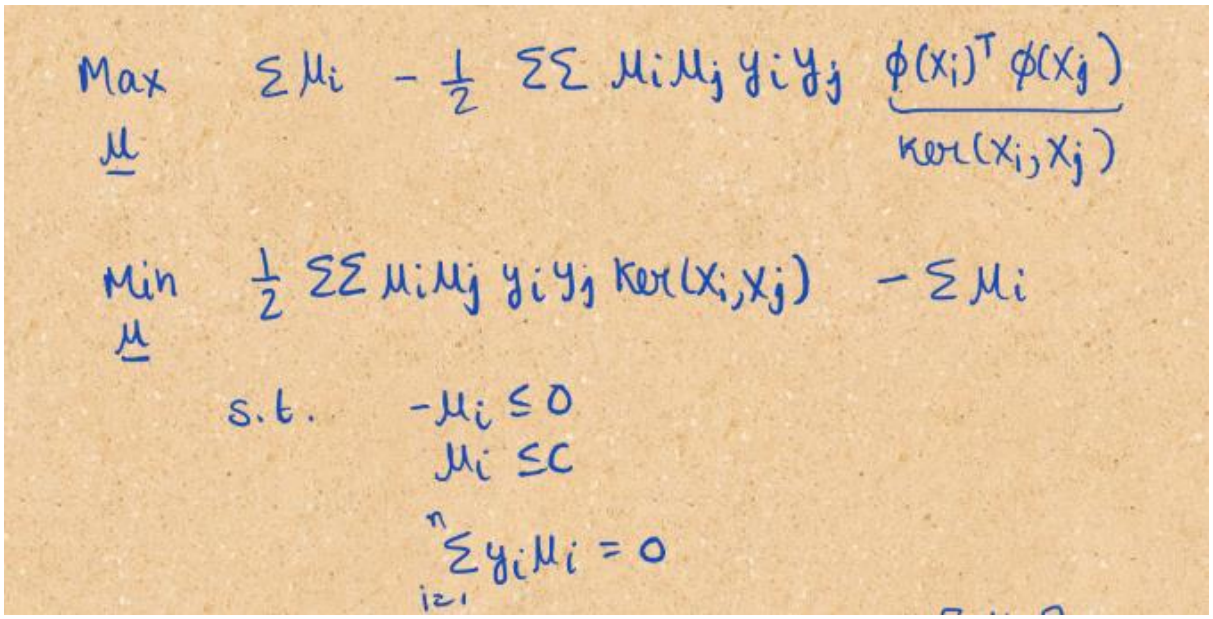
6) $\mu_i (1 - \xi_i - y_i (w^T x_i + b)) = 0$, $\lambda_i \xi_i = 0 \quad \forall \ i$

↳ complementary slackness.

Now, we form the dual of the problem,



Now, to convert it into the form feedable to cvxopt,



Now we introduce a matrix H, to simplify the problem

$$H_{ij} = y_i y_j \, Ker(x_i, x_j)$$

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

$$\frac{1}{2} \sum\sum \mu_i \mu_j H_{ij} - \sum \mu_i$$

$$= \frac{1}{2} \mu^T H \mu - [1 \; 1 \ldots 1] \mu$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n \times 1} \quad n \times 1$$

$$\underset{\mu}{\text{Min}} \; \frac{1}{2} \mu^T H \mu + [-1 \; -1 \ldots -1] \mu$$

$$\text{s.t.} \quad \begin{bmatrix} -I_{n \times n} \\ I_{n \times n} \end{bmatrix} \mu = \begin{bmatrix} [0]_{n \times 1} \\ [C]_{n \times 1} \end{bmatrix}$$

$$y^T \mu = 0$$

← Dual of SVM

Now, the form is feedable to cvxopt library, the parameters to pass are as follows:

cvxopt.solvers.qp(P, q, G, h, A, b)

$$P := H$$

$$q := [-1]_{n \times 1}$$

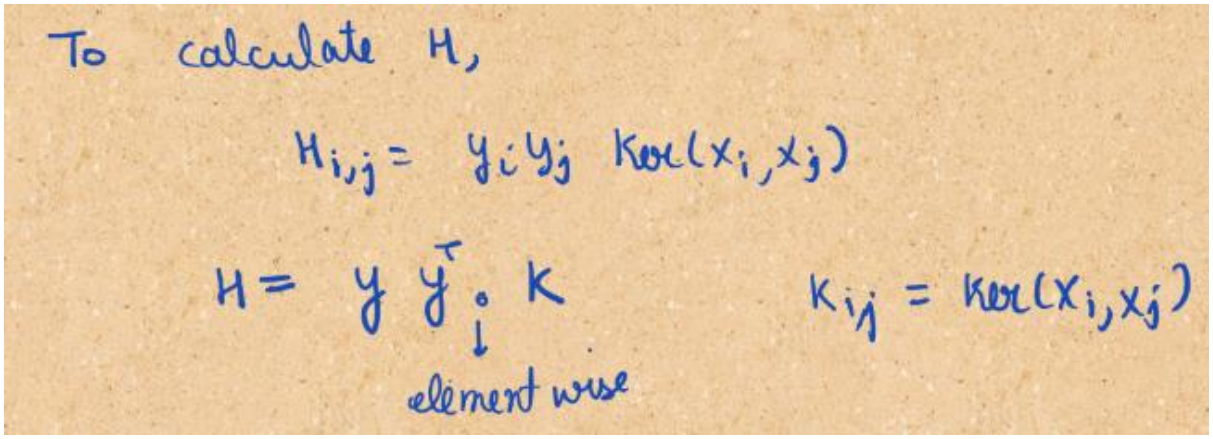
$$G := \begin{bmatrix} -I_{n \times n} \\ I_{n \times n} \end{bmatrix}$$

$$h := \begin{bmatrix} 0_{n \times 1} \\ C_{n \times 1} \end{bmatrix}$$

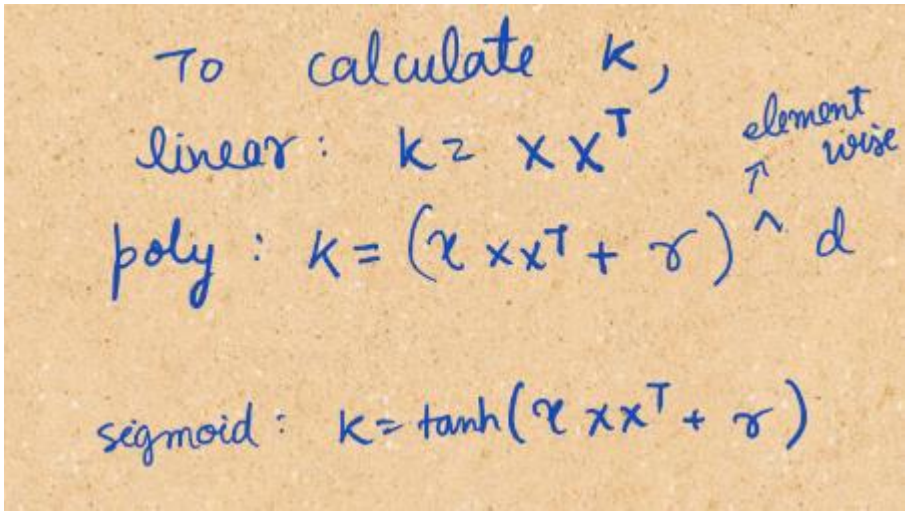
$$A := y^T$$

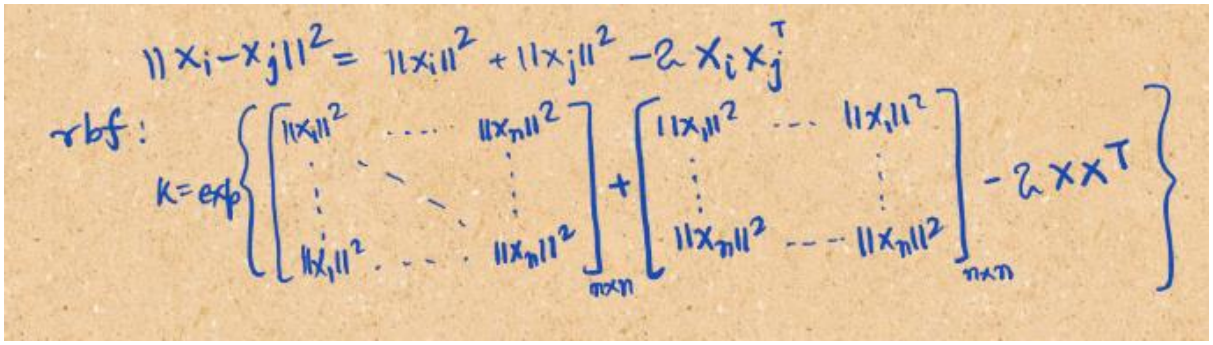
$$b := 0 \; (\text{scalar})$$

To calculate H,



To calculate matrix K,





After the solution has been found by cvxopt,

We can calculate b as follows:

$$b = y_j - \phi(x_j)^T w_{svm}$$
$$= y_j - \sum_{i \in S} \mu_i y_i \phi(x_j)^T \phi(x_i)$$

$$\text{Isl } b = \sum_{j \in S} y_j - \sum_{j \in S} \sum_{i \in S} \mu_i y_i K_{ji}$$
$$= \text{sum}(sy) - \sum_{j \in S} \text{sum}(s\mu \cdot sy \cdot K[j, S])$$

To predict data,

$$h(x_i) = \sum_{j \in S} \mu_j y_j \underbrace{\phi(x_j)^T \phi(x_i)}_{\text{Ker}(x_j, x_i)} + b$$

$$x_j^T X = K_{ji}$$

$$= \text{sum}(s\mu * st * K[S, i])$$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix} \quad \begin{bmatrix} x_1' x_1^T & x_1' x_2^T \\ & \ddots \end{bmatrix}$$

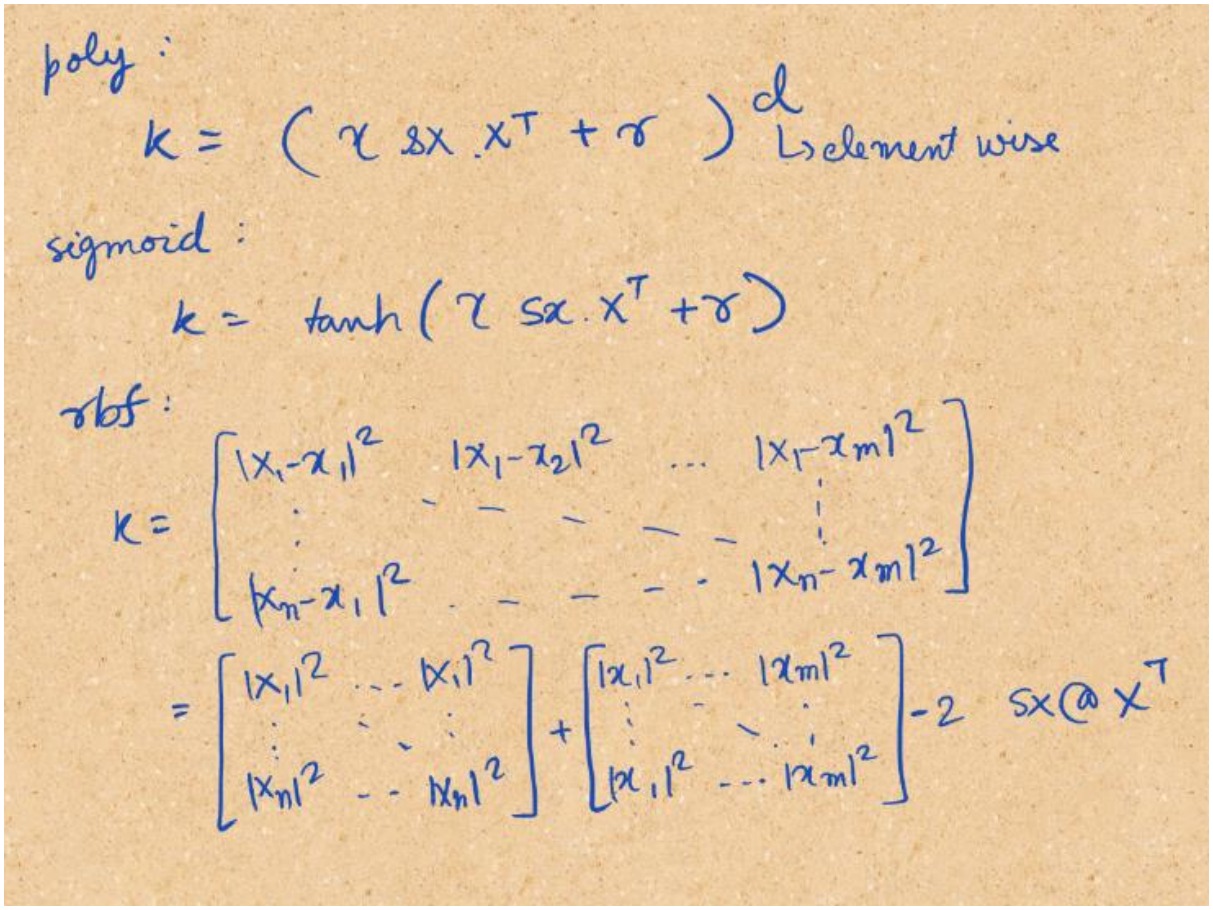
linear: $\langle x, x' \rangle$
poly: $(\gamma \langle x, x' \rangle + r)^d$
rbf: $\exp(-\gamma \|x - x'\|^2)$
sigmoid: $\tanh(\gamma \langle x, x \rangle + r)$

linear:

$$k = \begin{bmatrix} x_1 x_1^T & x_1 x_2^T & \cdots & x_1 x_m^T \\ \vdots & & & \vdots \\ x_n x_1^T & \cdots & \cdots & x_n x_m^T \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} x_1^T & x_2^T & \cdots & x_m^T \end{bmatrix}$$

$$k = sX \,@\, X^T$$



The code which does all the above is shown below:

Code to calculate K:

```
def buildK(this, X1, X2):
    K = X1.dot(X2.T)
    if (this.kernel == 'poly'):
        K = (this.gamma * K + this.coef0)**this.degree
    elif (this.kernel == 'sigmoid'):
        K = np.tanh(this.gamma * K + this.coef0)
    elif (this.kernel == 'rbf'):
        sq1 = np.diag(X1.dot(X1.T)).reshape((-1, 1))*np.ones((1, X2.shape[0]))
        sq2 = np.diag(X2.dot(X2.T)).reshape((1, -1))*np.ones((X1.shape[0], 1))
        K = 2*K-sq1-sq2
        K = np.exp(this.gamma * K)
    # All other kernels are treated to be linear
    this.K = K
```

Code to convert the data and feed to cvxopt solver:

```
def fit(this, X, t):
    this.N = X.shape[0]
    this.buildK(X, X)
    t = t.reshape((-1, 1)) * 1.0
    P = cvxopt.matrix((t.dot(t.T))*this.K)
    q = cvxopt.matrix(-np.ones((this.N, 1)))
    G = cvxopt.matrix(np.concatenate((-np.eye(this.N), np.eye(this.N))))
    h = cvxopt.matrix(np.concatenate((np.zeros((this.N, 1)), this.C * np.ones((this.N, 1)))))
    A = cvxopt.matrix(t.T)
    b = cvxopt.matrix(0.0)

    cvxopt.solvers.options['show_progress'] = False
    sol = cvxopt.solvers.qp(P, q, G, h, A, b)

    mu = np.array(sol['x']) # The lagrange multipliers
    this.sv_idx = np.where(mu > this.threshold)[0] # indices of all the support vectors
```

Code to calculate the support vectors and b,

```
# Extract the support vectors
this.sx = X[this.sv_idx,:]
this.st = t[this.sv_idx]
this.mu = mu[this.sv_idx]

this.b = np.sum(this.st)
for j in this.sv_idx:
    this.b -= np.sum(this.mu * this.st * (this.K[j, this.sv_idx].reshape((-1, 1))))
this.b /= this.sv_idx.shape[0]
```

Code to classify new data:

```
# Make the classifier
def predict(X):
    this.buildK(this.sx, X)
    this.y = np.zeros((X.shape[0],))
    for i in range(this.y.shape[0]):
        this.y[i] = np.sum(this.mu * this.st * this.K[:,i].reshape((-1, 1)))+this.b
    return np.sign(this.y)

this.predict = predict
```

Some utility functions defined:

The function to do crossvalidation:

```
def crossval_score(clf, X, t, cv = 4):
    score = np.array([])
    n = X.shape[0]
    batch_size = n//cv
    for i in range(cv):
        beg, end = i*batch_size, (i+1)*batch_size
        test_X, test_t = X[beg:end,:], t[beg:end]
        train_X, train_t = np.delete(X, range(beg, end), axis = 0), np.delete(t, range(beg, end))
        clf.fit(train_X, train_t)
        predicted_t = clf.predict(test_X)
        f1 = f1_score(test_t, predicted_t, average='weighted')
        score = np.append(score, f1)
    return score
```

The function to grid search the parameters:

```
def findParameters(kernels, X, t, cr = np.logspace(0, 1, 10), gr = np.logspace(0, 1, 10), dr = range(1, 5), op = False):
    # returns the classifier with the best parameters
    best_score = np.array([-1])
    for kernel in kernels:
        if (kernel == 'poly'):
            cnt=0
            allscores = {}
            tscores = {}
            for d in dr:
                allscores[d] = np.array([])
                tscores[d] = np.array([])
        else:
            allscores = np.array([])
            tscores = np.array([])
        if (op):
            print('Kernel = %s' % (kernel))
        for c in cr:
            g2 = False
            for g in gr:
                if (g2 and kernel == 'linear'):
                    break
                g2 = True
                d2 = False
                for d in dr:
                    if (d2 and kernel != 'poly'):
                        break
                    d2 = True
                    clf = svm.SVC(kernel = kernel, C = c, gamma = g, degree = d)
                    score = crossval_score(clf, X, t, cv = cv)
                    if (kernel == 'poly'):
                        allscores[d] = np.append(allscores[d], score.mean())
                        tscores[d] = np.append(tscores[d], clf.fit(X, t).score(X, t))
                    else:
                        allscores = np.append(allscores, score.mean())
                        tscores = np.append(tscores, clf.fit(X, t).score(X, t))
                    if (op):
                        print('c = %0.4f, g = %0.4f, d = %i, score = %0.4f (+/- %0.4f)'%(c, g, d, score.mean(), 2*score.std()))
```

```
                    if (score.mean() > best_score.mean()):
                        best_score = score
                        best_clf = clf
        plotVariations(kernel, cr, gr, dr, allscores, tscores)
    return best_clf.fit(X, t), best_score
```

The function to plot the variations:

```
def plotVariations(kernel, cr, gr, dr, cvscores, tscores):
    if (kernel == 'linear'):
        # Variation with C only
        plt.plot(cr, cvscores, label='cross-validation')
        plt.plot(cr, tscores, label='training')
        plt.xlabel('C')
        plt.ylabel('Accuracy')
        plt.xscale('log')
        plt.title('Accuracy v/s C')
        plt.legend(loc='best')
        plt.show()
```

```
elif (kernel == 'rbf' or kernel == 'sigmoid'):
    # Variation with C and gamma
    gr, cr = np.meshgrid(gr, cr)
    cvscores = cvscores.reshape(cr.shape)
    fig, ax = plt.subplots()
    cs = ax.contourf(cr, gr, cvscores)
    cbar = fig.colorbar(cs)
    plt.xlabel('C')
    plt.ylabel('gamma')
    plt.title('Contour plot for Accuracy v/s C and gamma (cross-validation)')
    plt.xscale('log')
    plt.yscale('log')
    plt.show()
    # training
    tscores = tscores.reshape(cr.shape)
    fig, ax = plt.subplots()
    cs = ax.contourf(cr, gr, tscores)
    cbar = fig.colorbar(cs)
    plt.xlabel('C')
    plt.ylabel('gamma')
    plt.title('Contour plot for Accuracy v/s C and gamma (training)')
    plt.xscale('log')
    plt.yscale('log')
    plt.show()
```

```
else:
    # Variation with C, gamma and degree
    gr, cr = np.meshgrid(gr, cr)
    for d in dr:
        cvscores[d] = cvscores[d].reshape(cr.shape)
        fig, ax = plt.subplots()
        cs = ax.contourf(cr, gr, cvscores[d])
        cbar = fig.colorbar(cs)
        plt.xlabel('C')
        plt.ylabel('gamma')
        plt.title('Contour plot for Accuracy v/s C and gamma, degree=%i (cross-validation)'%(d))
        plt.xscale('log')
        plt.yscale('log')
        plt.show()
        # training
        tscores[d] = tscores[d].reshape(cr.shape)
        fig, ax = plt.subplots()
        cs = ax.contourf(cr, gr, tscores[d])
        cbar = fig.colorbar(cs)
        plt.xlabel('C')
        plt.ylabel('gamma')
        plt.title('Contour plot for Accuracy v/s C and gamma, degree=%i (training)'%(d))
        plt.xscale('log')
        plt.yscale('log')
        plt.show()
```

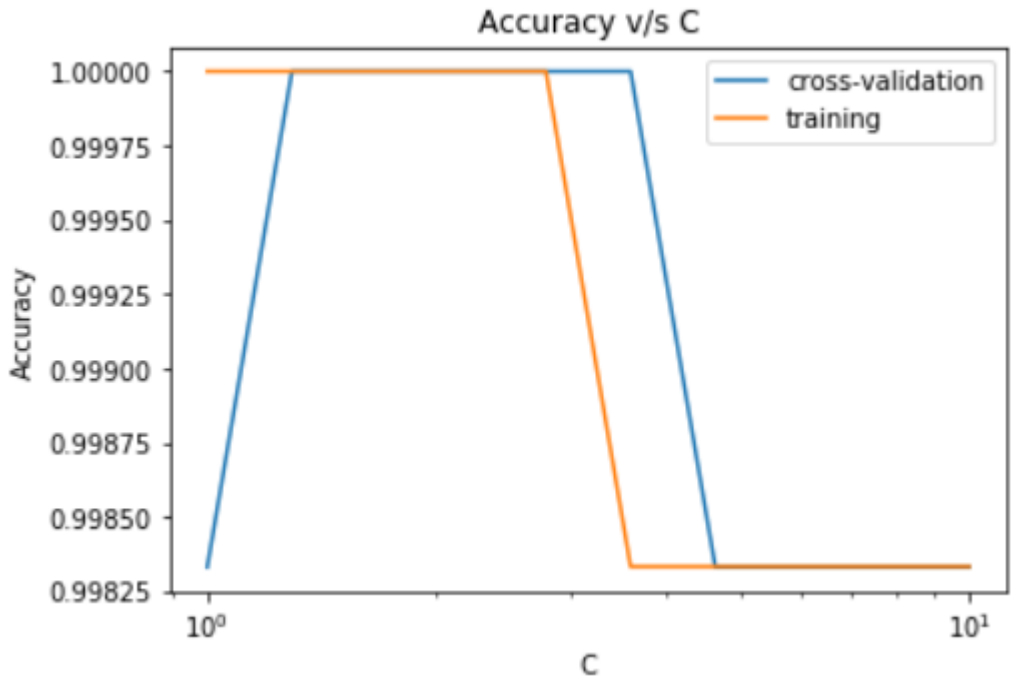
Binary Classification (Using all 25 features):

The following code does the classification:

```
# Part 1 (Binary Classification) using all 25 features
pairs = [(0, 1), (2, 3), (4, 5)]
for i, j in pairs:
    print('Pair: (%i, %i)'%(i, j))
    idx_i = np.where(t == i)[0]
    idx_j = np.where(t == j)[0]
    idx = np.concatenate((idx_i, idx_j))
    np.random.shuffle(idx)
    Xp = X[idx,:]
    tp = t[idx]
    tp = np.where(tp==i, -1, 1)
    for kernel in kernels:
        print('Kernel: %s'%(kernel))
        clf_p, cvscore_p = findParameters([kernel], Xp, tp)
        param_p = clf_p.get_params()
        myclf = mysvm(kernel=kernel, C=param_p['C'], gamma=param_p['gamma'], degree=param_p['degree'], coef0=param_p['coef0'])
        mycvscore = crossval_score(myclf, Xp, tp, cv = cv)
        myclf.fit(Xp, tp)
        print(param_p)
        print('sklearn cross-val score: %0.4f, cvxopt cross-val score: %0.4f'%(cvscore_p.mean(), mycvscore.mean()))
        print('sklearn num of SV: %i, cvxopt num of SV: %i'%(clf_p.support_.shape[0], myclf.sv_idx.shape[0]))
        print('sklearn indices of support vector:', np.sort(clf_p.support_))
        print('cvxopt indices of support vector :', myclf.sv_idx)
    print('\n')
```

Pair (0,1)

Linear Kernel



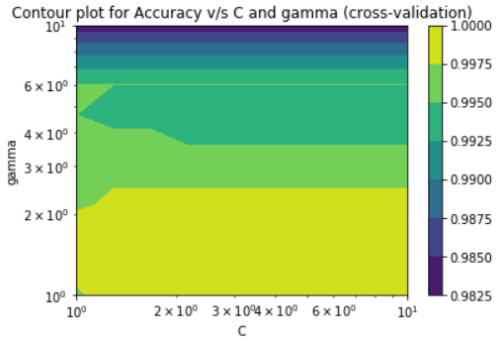
The following hyper-parameters were obtained:

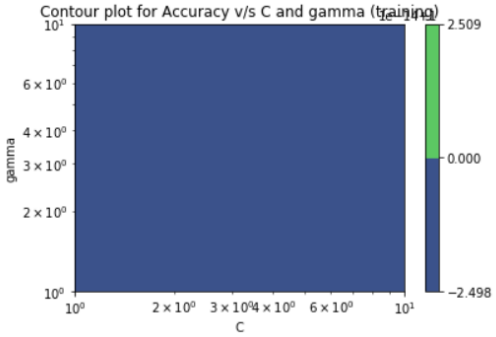
```
{'C': 1.2915496650148839, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 1.0, 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.
001, 'verbose': False}
```

The comparison between results from sklearn.svm library and cvxopt is shown below:

```
sklearn cross-val score: 1.0000, cvxopt cross-val score: 0.9983
sklearn num of SV: 31, cvxopt num of SV: 31
sklearn indices of support vector: [ 23  99 140 144 146 178 199 240 261 264 276 302 303 317 318 327 381 386
 407 408 418 437 452 471 473 485 493 516 545 546 551]
cvxopt indices of support vector : [ 23  99 140 144 146 178 199 240 261 264 276 302 303 317 318 327 381 386
 407 408 418 437 452 471 473 485 493 516 545 546 551]
```

rbf Kernel





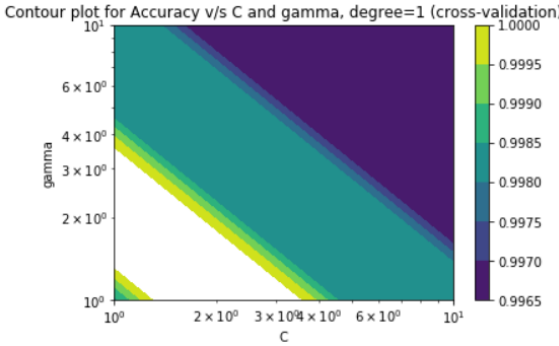
The following hyper-parameters were obtained:

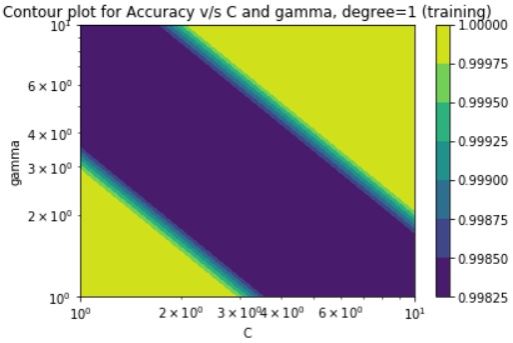
```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 1.2
915496650148839, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.00
1, 'verbose': False}
```

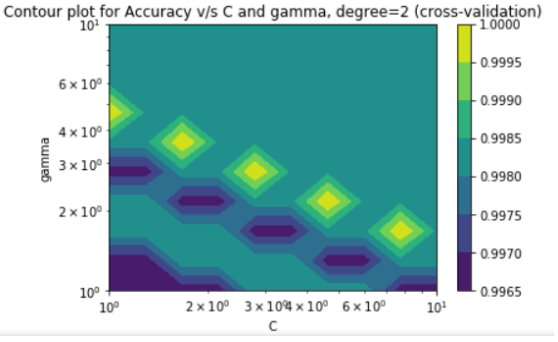
The comparison between sklearn and cvxopt:

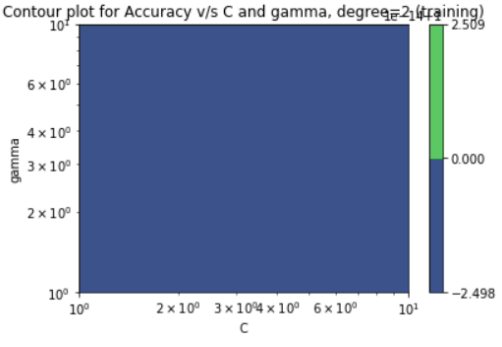
```
sklearn cross-val score: 1.0000, cvxopt cross-val score: 0.9983
sklearn num of SV: 42, cvxopt num of SV: 42
sklearn indices of support vector: [ 16  23  64  86  99 140 144 146 178 191 199 240 261 264 276 298 302 303
 317 318 327 381 386 407 408 418 436 437 452 469 471 473 474 485 489 493
 516 540 545 546 551 575]
cvxopt indices of support vector : [ 16  23  64  86  99 140 144 146 178 191 199 240 261 264 276 298 302 303
 317 318 327 381 386 407 408 418 436 437 452 469 471 473 474 485 489 493
 516 540 545 546 551 575]
```

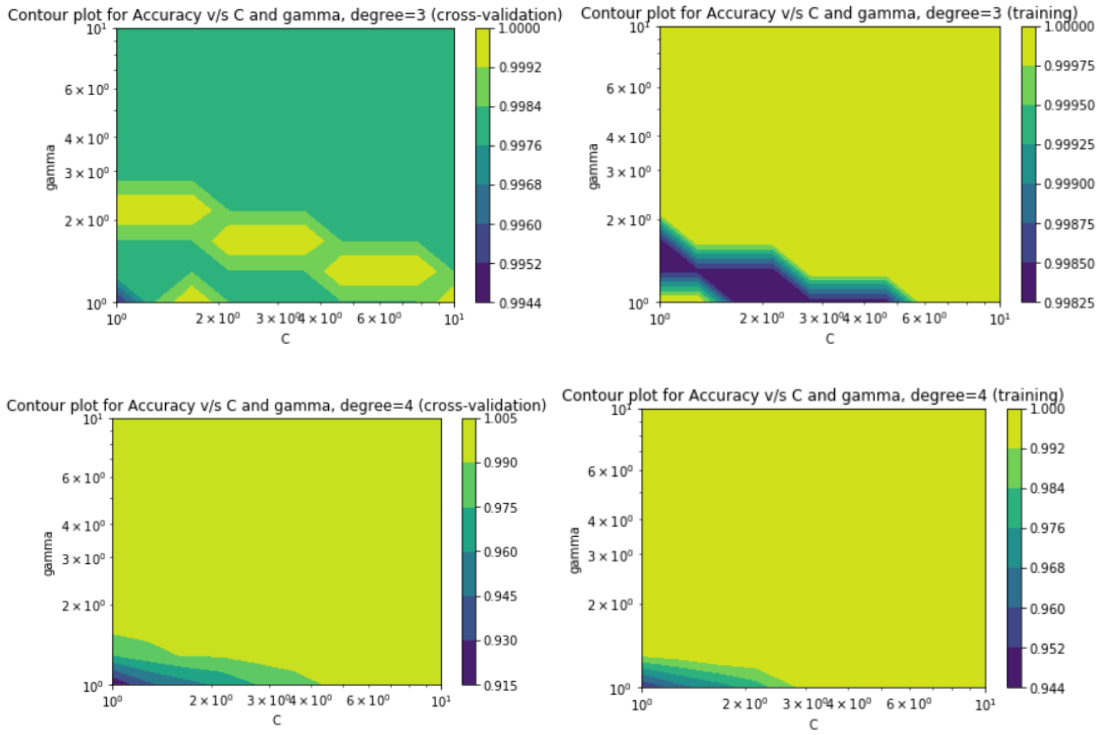
Poly kernel











The following hyper-parameters were obtained:

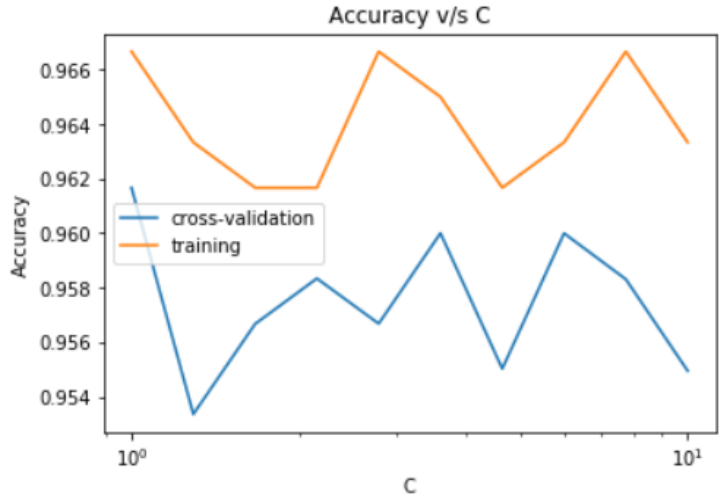
```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 1.2
915496650148839, 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.00
1, 'verbose': False}
```

The comparison:

```
sklearn cross-val score: 1.0000, cvxopt cross-val score: 0.9983
sklearn num of SV: 31, cvxopt num of SV: 31
sklearn indices of support vector: [ 23  99 140 144 146 178 199 240 261 264 276 302 303 317 318 327 381 386
 407 408 418 437 452 471 473 485 493 516 545 546 551]
cvxopt indices of support vector : [ 23  99 140 144 146 178 199 240 261 264 276 302 303 317 318 327 381 386
 407 408 418 437 452 471 473 485 493 516 545 546 551]
```

Pair (2, 3)

Linear Kernel



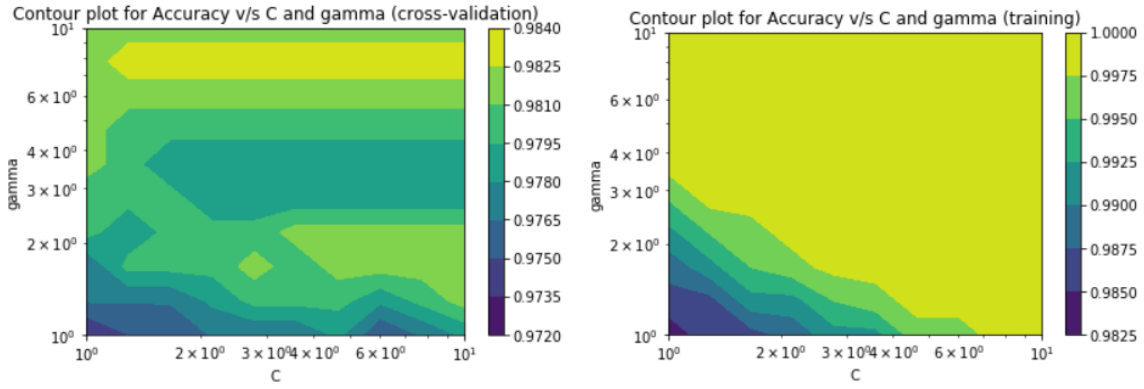
The following hyperparameters were obtained:

```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 1.
0, 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbos
e': False}
```

The comparison:

```
sklearn cross-val score: 0.9617, cvxopt cross-val score: 0.9550
sklearn num of SV: 139, cvxopt num of SV: 143
sklearn indices of support vector: [  3  12  18  19  22  30  39  44  46  51  52  56  57  60  61  62  64  67
  71  72  81  86 102 105 107 112 113 114 116 121 125 134 138 144 148 153
 155 156 158 162 164 167 168 172 175 182 184 187 188 190 196 206 207 208
 214 220 223 224 228 233 236 240 246 248 250 253 257 259 260 261 262 263
 264 266 268 279 287 302 305 309 310 312 318 330 333 339 340 345 357 362
 363 364 379 389 397 401 403 406 409 411 413 419 424 432 468 471 473 482
 483 484 486 488 489 490 493 495 496 506 507 508 512 516 519 524 533 535
 552 563 565 573 576 579 586 587 590 592 593 594 597]
cvxopt indices of support vector : [  3  12  18  19  22  30  39  44  46  51  52  56  57  60  61  62  64  67
  71  72  81  86 102 105 107 112 113 114 116 121 125 134 138 144 148 153
 155 156 158 162 164 167 168 172 175 182 184 187 188 190 196 206 207 208
 214 220 223 224 228 233 236 240 246 248 250 253 257 259 260 261 262 263
 264 266 268 279 281 287 302 305 309 310 312 318 330 333 339 340 345 350
 357 362 363 364 379 380 389 397 401 403 406 409 411 413 419 424 432 468
 471 473 482 483 484 486 488 489 490 493 495 496 506 507 508 512 516 519
 524 533 535 549 552 563 565 573 576 579 586 587 590 592 593 594 597]
```

Rbf Kernel



The following hyperparameters were obtained:

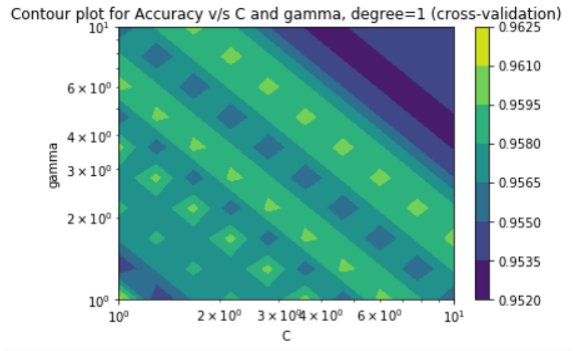
```
{'C': 1.2915496650148839, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 7.742636826811269, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tru
e, 'tol': 0.001, 'verbose': False}
```

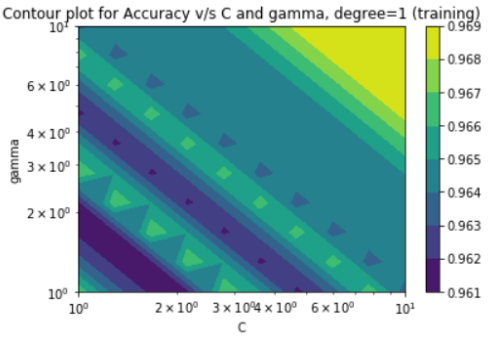
The comparison:

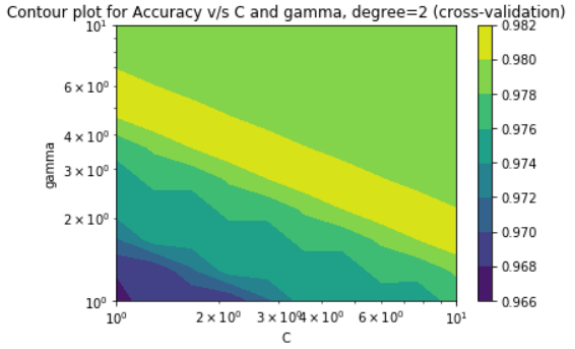
```
sklearn cross-val score: 0.9833, cvxopt cross-val score: 0.9833
sklearn num of SV: 453, cvxopt num of SV: 457
sklearn indices of support vector: [  0   1   2   4   6   9  10  11  12  13  14  16  17  18  19  22  25  26
  27  28  30  31  32  34  36  38  39  40  41  42  44  45  46  47  48  49
  50  51  52  53  54  55  56  57  59  60  61  62  64  66  67  69  71  72
  73  74  75  76  79  81  82  84  85  87  88  90  91  92  94  95  96  97
  99 102 103 104 105 106 107 108 110 111 112 113 114 115 116 117 118 119
 120 121 124 125 126 130 131 132 133 134 136 138 139 140 142 144 145 146
 148 149 150 151 152 153 154 155 158 160 161 162 163 164 165 167 168 169
 170 171 172 173 175 176 177 179 180 181 182 183 184 186 187 188 189 190
 191 192 193 194 195 196 198 199 200 201 202 203 204 205 206 207 208 209
 210 211 214 216 217 218 220 221 223 224 225 227 228 229 232 233 234 235
 236 238 239 240 241 242 244 245 246 247 248 249 250 253 255 256 257 259
 260 261 262 263 264 265 266 267 268 271 272 274 275 277 279 281 284 285
 286 287 290 291 292 293 294 296 297 298 299 301 302 304 305 307 309 310
```

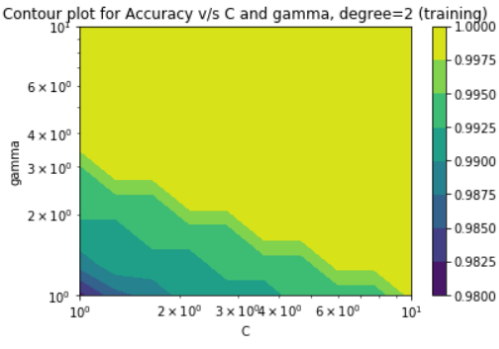
```
 312 313 314 316 317 319 321 322 326 327 328 329 330 331 332 333 335 336
 337 338 339 340 342 343 345 346 347 348 349 350 351 355 356 357 359 360
 362 363 364 366 367 369 370 371 372 373 374 375 377 378 379 381 382 383
 385 386 388 389 390 391 392 393 394 395 396 398 399 400 401 402 403 404
 405 406 408 409 410 411 412 413 414 415 416 418 419 421 422 423 424 425
 427 429 432 434 435 438 440 441 442 443 444 445 446 447 451 453 454 455
 460 461 462 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479
 482 483 484 486 487 488 489 490 491 492 493 495 496 497 498 500 501 502
 504 505 506 507 508 509 510 511 512 513 514 515 516 518 519 520 523 524
 525 528 530 531 532 534 535 538 540 542 543 544 545 546 547 548 549 550
 551 552 553 554 555 556 557 558 559 562 563 564 565 566 567 568 572 573
 576 577 579 580 581 583 584 585 586 587 588 589 590 591 592 593 594 596
 597 598 599]
cvxopt indices of support vector : [  0   1   2   4   6   9  10  11  12  13  14  16  17  18  19  22  25  26
  27  28  29  30  31  32  34  36  38  39  40  41  42  44  45  46  47  48
  49  50  51  52  53  54  55  56  57  59  60  61  62  64  66  67  69  71
  72  73  74  75  76  79  81  82  84  85  87  88  90  91  92  94  95  96
  97  99 102 103 104 105 106 107 108 110 111 112 113 114 115 116 117 118

119 120 121 124 125 126 127 130 131 132 133 134 136 138 139 140 142 144
145 146 148 149 150 151 152 153 154 155 158 160 161 162 163 164 165 167
168 169 170 171 172 173 175 176 177 179 180 181 182 183 184 186 187 188
189 190 191 192 193 194 195 196 198 199 200 201 202 203 204 205 206 207
208 209 210 211 214 216 217 218 220 221 223 224 225 227 228 229 232 233
234 235 236 238 239 240 241 242 244 245 246 247 248 249 250 253 255 256
257 259 260 261 262 263 264 265 266 267 268 271 272 274 275 277 279 281
284 285 286 287 290 291 292 293 294 296 297 298 299 301 302 304 305 307
309 310 312 313 314 316 317 319 321 322 326 327 328 329 330 331 332 333
335 336 337 338 339 340 342 343 345 346 347 348 349 350 351 355 356 357
359 360 362 363 364 366 367 369 370 371 372 373 374 375 377 378 379 381
382 383 385 386 388 389 390 391 392 393 394 395 396 398 399 400 401 402
403 404 405 406 408 409 410 411 412 413 414 415 416 418 419 421 422 423
424 425 427 429 432 434 435 438 440 441 442 443 444 445 446 447 451 453
454 455 456 460 461 462 465 466 467 468 469 470 471 472 473 474 475 476
477 478 479 480 482 483 484 486 487 488 489 490 491 492 493 495 496 497
498 500 501 502 504 505 506 507 508 509 510 511 512 513 514 515 516 518
519 520 523 524 525 528 530 531 532 534 535 538 540 542 543 544 545 546
547 548 549 550 551 552 553 554 555 556 557 558 559 562 563 564 565 566

567 568 572 573 576 577 579 580 581 583 584 585 586 587 588 589 590 591
592 593 594 596 597 598 599]
```

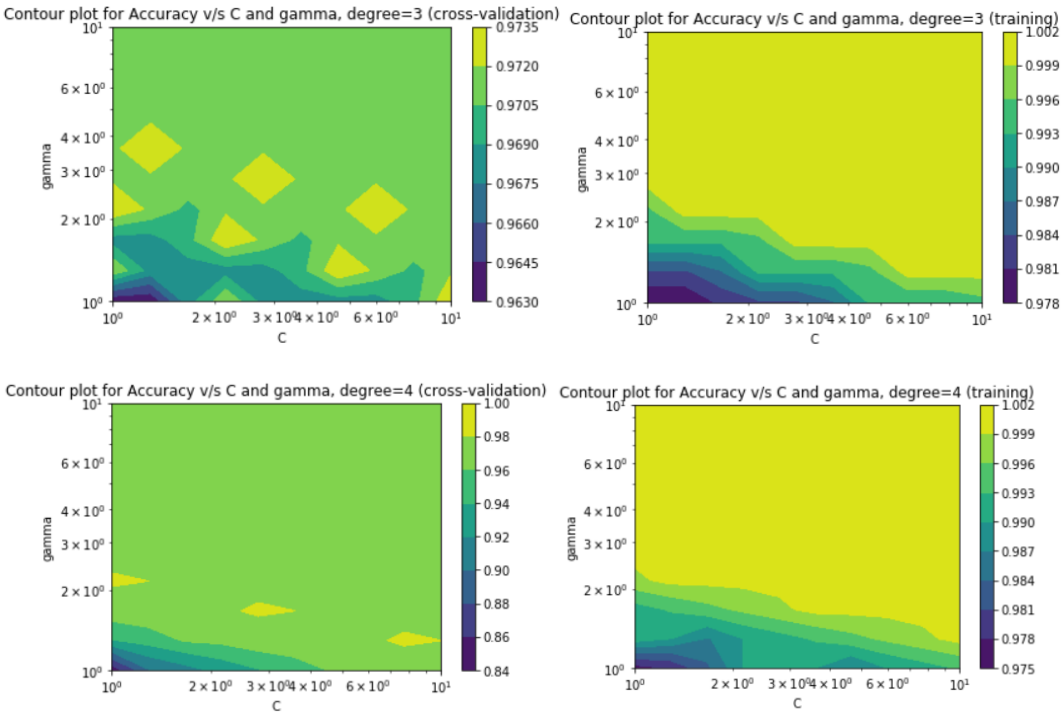
Poly Kernel











The following hyperparameters were obtained:

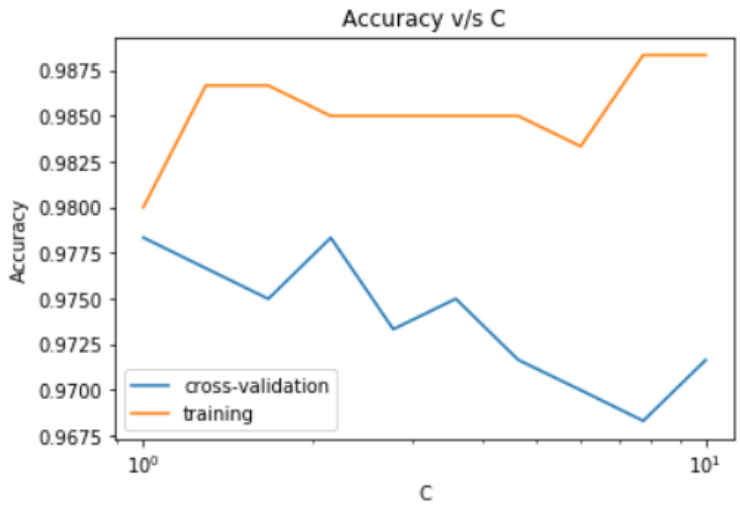
```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 2, 'gamma': 5.9
94842503189409, 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.00
1, 'verbose': False}
```

The comparison:

```
sklearn cross-val score: 0.9817, cvxopt cross-val score: 0.9817
sklearn num of SV: 122, cvxopt num of SV: 124
sklearn indices of support vector: [  4   6  12  14  18  35  51  52  56  60  62  64  66  71  76  86  92  94
 102 104 105 106 107 108 113 114 125 136 138 144 145 158 161 165 167 168
 171 172 176 180 181 183 194 196 199 202 205 206 216 220 229 231 232 236
 240 246 249 256 259 260 261 262 263 264 265 268 278 281 290 293 302 310
 328 336 338 340 343 345 346 357 362 363 364 369 371 377 385 392 399 403
 409 413 424 460 469 470 471 472 473 475 476 483 489 493 500 505 506 508
 518 519 528 534 538 547 552 560 563 568 573 579 587 591]
cvxopt indices of support vector : [  4   6  12  14  18  35  51  52  56  60  62  64  66  71  76  86  92  94
 102 104 105 106 107 108 113 114 125 136 138 144 145 158 161 165 167 168
 171 172 176 180 181 183 194 196 199 202 205 206 216 220 229 231 232 236
 240 246 249 256 259 260 261 262 263 264 265 268 278 281 290 293 302 310
 328 336 338 340 343 345 346 357 362 363 364 366 369 371 377 385 392 399
 403 409 413 424 460 469 470 471 472 473 475 476 483 489 493 496 500 505
 506 508 518 519 528 534 538 547 552 560 563 568 573 579 587 591]
```

Pair (4, 5)

Linear kernel



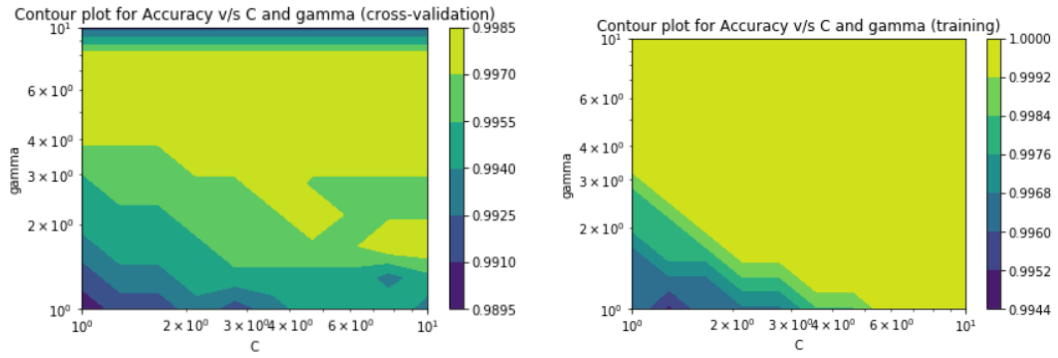
The following hyperparameters were obtained:

```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 1.
0, 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbos
e': False}
```

The comparison:

```
sklearn cross-val score: 0.9783, cvxopt cross-val score: 0.9733
sklearn num of SV: 125, cvxopt num of SV: 126
sklearn indices of support vector: [  0   7  24  26  28  29  30  37  41  42  44  46  49  50  51  53  62  66
  69  73  78  79  83  98 102 104 105 120 124 130 133 140 144 154 159 168
 176 180 183 186 188 196 198 201 204 205 207 213 220 224 225 229 230 232
 240 256 265 266 268 278 281 291 301 303 307 314 315 318 337 342 345 346
 347 349 356 361 379 387 392 400 403 406 407 411 413 417 420 430 433 434
 436 441 442 447 448 456 466 475 484 491 493 494 499 505 510 515 519 524
 525 530 544 549 554 559 561 565 569 571 572 573 579 584 587 595 599]
cvxopt indices of support vector : [  0   7  24  26  28  29  30  37  41  42  44  46  49  50  51  53  62  66
  69  73  78  79  83  85  98 102 104 105 120 124 130 133 140 144 154 159
 168 176 180 183 186 188 196 198 201 204 205 207 213 220 224 225 229 230
 232 240 256 265 266 268 278 281 291 301 303 307 314 315 318 337 342 345
 346 347 349 356 361 379 387 392 400 403 406 407 411 413 417 420 430 433
 434 436 441 442 447 448 456 466 475 484 491 493 494 499 505 510 515 519
 524 525 530 544 549 554 559 561 565 569 571 572 573 579 584 587 595 599]
```

Rbf kernel



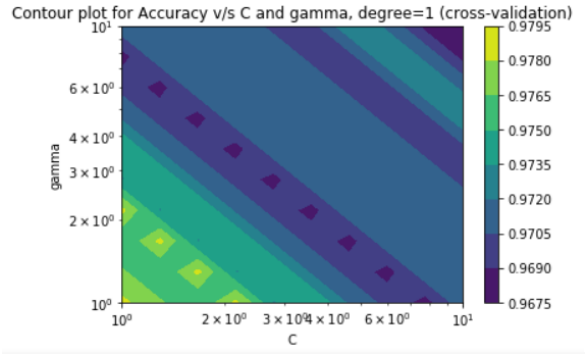
The following hyperparameters were obtained:

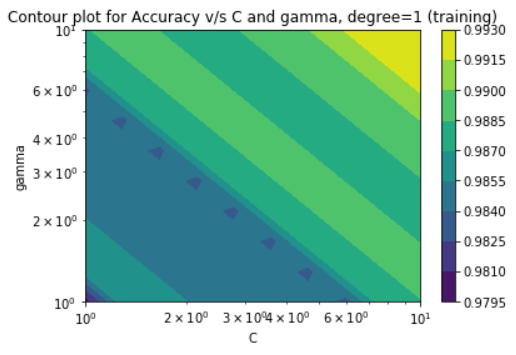
```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 4.6
41588833612778, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001,
'verbose': False}
```

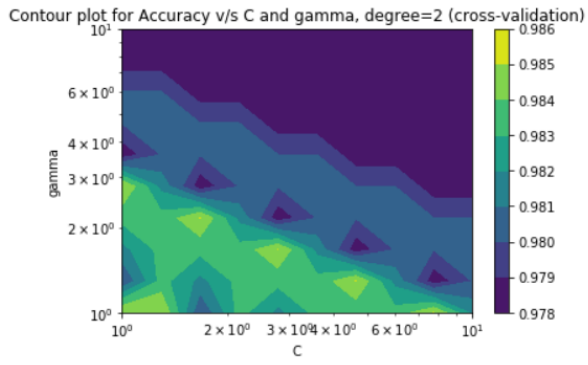
The comparison:

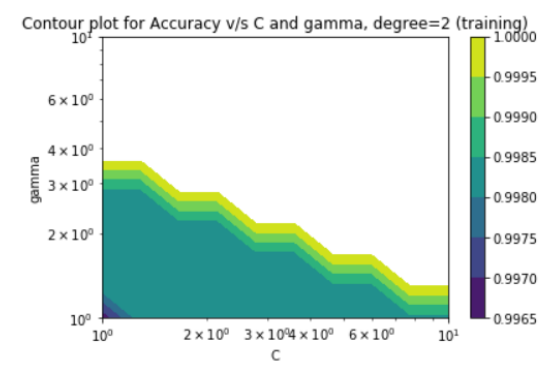
```
sklearn cross-val score: 0.9983, cvxopt cross-val score: 0.9983
sklearn num of SV: 258, cvxopt num of SV: 270
sklearn indices of support vector: [  6   8   9  11  17  24  25  26  28  29  30  31  35  39  41  42  44  46
  49  50  51  53  54  59  60  62  66  69  71  72  73  74  77  78  79  81
  93  97  98 102 104 105 106 107 111 112 113 114 115 120 121 122 123 124
 127 128 131 133 140 141 144 145 154 156 159 160 168 170 172 180 183 184
 185 186 190 191 194 196 198 201 203 204 206 207 208 217 219 224 225 226
 229 235 236 239 240 241 242 244 245 248 249 250 259 260 263 265 266 272
 273 278 279 280 285 287 289 291 292 294 295 297 298 299 300 301 302 303
 304 307 310 314 315 317 320 321 324 327 328 330 334 336 337 342 345 346
 349 351 352 356 361 363 364 365 373 374 379 385 386 387 391 392 394 399
 400 401 404 405 407 408 411 413 417 420 421 425 428 430 432 433 436 437
 438 439 442 445 446 447 448 451 456 457 459 460 462 463 464 466 467 470
 472 473 481 482 484 485 491 492 493 494 498 499 504 505 510 511 512 515
 519 522 524 525 526 530 535 536 538 544 545 547 548 549 551 553 554 555
 558 559 561 563 564 565 567 568 570 571 572 574 576 578 580 582 583 584
 587 588 591 595 597 599]

cvxopt indices of support vector : [  2   6   7   8   9  11  17  24  25  26  28  29  30  31  35  37  39  41
  42  44  46  47  49  50  51  53  54  59  60  62  65  66  69  71  72  73
  74  77  78  79  81  93  97  98 102 104 105 106 107 111 112 113 114 115
 120 121 122 123 124 127 128 131 133 140 141 144 145 154 156 159 160 168
 170 172 180 183 184 185 186 190 191 194 196 198 199 201 203 204 206 207
 208 217 219 224 225 226 229 235 236 239 240 241 242 244 245 248 249 250
 259 260 263 265 266 272 273 278 279 280 285 287 289 291 292 294 295 297
 298 299 300 301 302 303 304 307 310 314 315 317 320 321 324 327 328 330
 331 334 336 337 342 345 346 349 351 352 356 361 363 364 365 373 374 379
 385 386 387 391 392 394 399 400 401 403 404 405 407 408 411 413 417 420
 421 425 428 430 432 433 436 437 438 439 442 445 446 447 448 451 456 457
 459 460 462 463 464 466 467 470 472 473 481 482 484 485 491 492 493 494
 498 499 504 505 509 510 511 512 515 518 519 522 524 525 526 530 535 536
 538 544 545 547 548 549 551 553 554 555 558 559 561 563 564 565 567 568
 570 571 572 574 576 578 580 582 583 584 587 588 589 591 595 596 597 599]
```

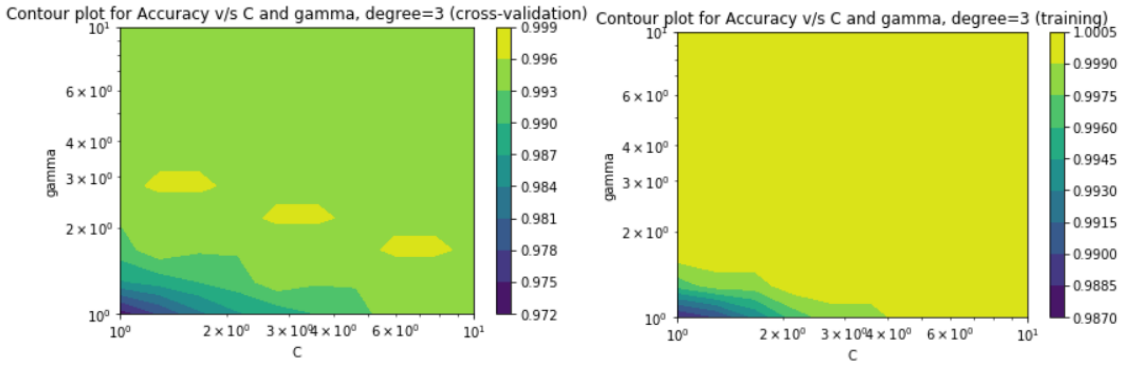
Poly kernel

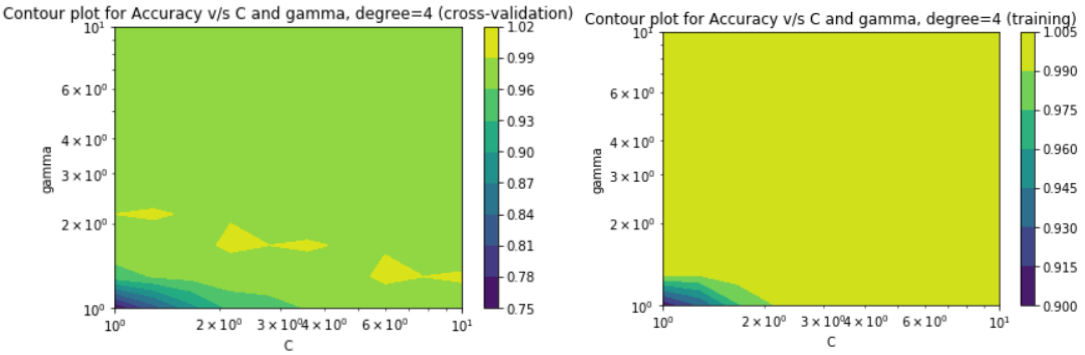












The hyperparameters obtained:

```
{'C': 1.2915496650148839, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
3, 'gamma': 2.7825594022071245, 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tr
ue, 'tol': 0.001, 'verbose': False}
```

The comparison:

```
sklearn cross-val score: 0.9967, cvxopt cross-val score: 0.9967
sklearn num of SV: 168, cvxopt num of SV: 170
sklearn indices of support vector: [  9  11  17  22  24  26  28  30  36  42  44  46  47  49  50  51  53  59
  62  66  69  72  73  81  83  88  90 102 104 105 106 110 120 124 126 134
 140 143 144 152 154 159 170 172 176 180 186 191 198 201 206 208 216 224
 225 226 229 233 235 240 243 247 250 252 265 266 269 278 279 280 281 284
 285 287 289 291 294 297 301 302 303 307 315 317 324 327 330 336 337 341
 342 345 346 349 352 356 371 379 387 392 396 398 400 403 405 406 407 411
 413 414 416 420 428 429 430 433 436 439 441 442 446 447 448 454 462 466
 478 479 484 486 491 492 494 495 499 505 510 511 512 515 516 517 518 522
 524 525 538 544 547 549 551 554 558 559 560 561 563 567 570 572 579 584
 587 588 595 597 598 599]

cvxopt indices of support vector : [  9  11  17  22  24  26  28  30  36  42  44  46  47  49  50  51  53  59
  62  66  69  72  73  81  83  88  90 102 104 105 106 110 120 124 126 134
 140 143 144 152 154 159 170 172 176 180 186 191 198 201 206 208 216 224
 225 226 229 233 235 240 243 247 250 252 265 266 269 278 279 280 281 284
 285 287 289 291 294 297 301 302 303 307 315 317 324 327 330 336 337 341
 342 345 346 349 352 356 371 379 385 387 392 396 398 400 403 405 406 407
 411 413 414 416 420 428 429 430 433 436 439 441 442 446 447 448 454 462
 463 466 478 479 484 486 491 492 494 495 499 505 510 511 512 515 516 517
 518 522 524 525 538 544 547 549 551 554 558 559 560 561 563 567 570 572
 579 584 587 588 595 597 598 599]
```

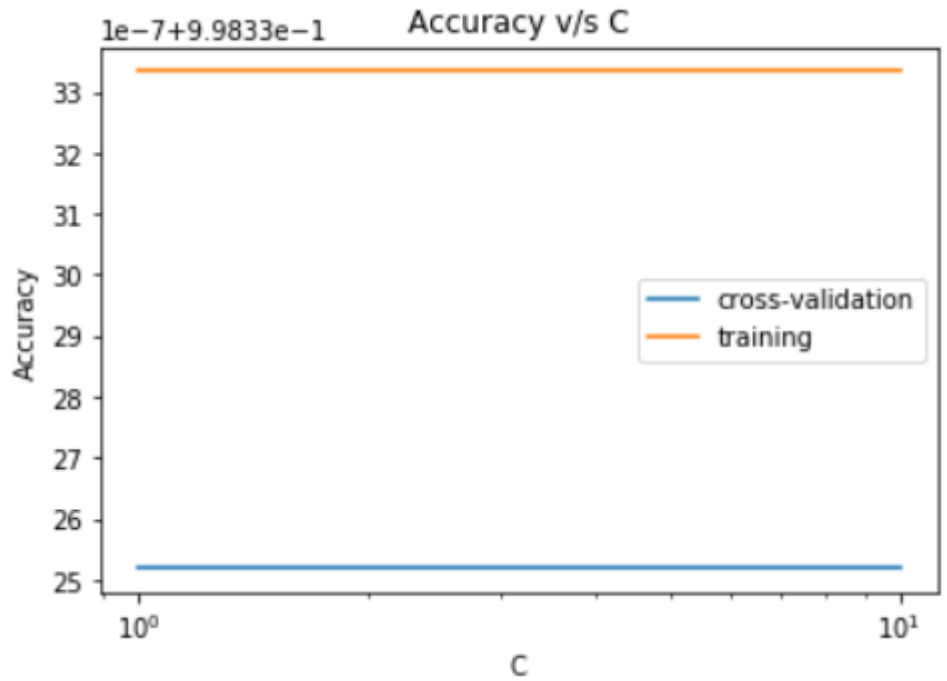
Binary Classification (Using 10 features):

The following code does the analyses:

```
# Part 1 (Binary Classification) using 10 features
pairs = [(0, 1), (2, 3), (4, 5)]
for i, j in pairs:
    print('Pair: (%i, %i)'%(i, j))
    idx_i = np.where(t == i)[0]
    idx_j = np.where(t == j)[0]
    idx = np.concatenate((idx_i, idx_j))
    np.random.shuffle(idx)
    Xp = X[idx,:10]
    tp = t[idx]
    tp = np.where(tp==i, -1, 1)
    for kernel in kernels:
        print('Kernel: %s'%(kernel))
        clf_p, cvscore_p = findParameters([kernel], Xp, tp)
        param_p = clf_p.get_params()
        myclf = mysvm(kernel=kernel, C=param_p['C'], gamma=param_p['gamma'], degree=param_p['degree'], coef0=param_p['coef0'])
        mycvscore = crossval_score(myclf, Xp, tp, cv = cv)
        myclf.fit(Xp, tp)
        print(param_p)
        print('sklearn cross-val score: %0.4f, cvxopt cross-val score: %0.4f'%(cvscore_p.mean(), mycvscore.mean()))
        print('sklearn num of SV: %i, cvxopt num of SV: %i'%(clf_p.support_.shape[0], myclf.sv_idx.shape[0]))
        print('sklearn indices of support vector:', np.sort(clf_p.support_))
        print('cvxopt indices of support vector :', myclf.sv_idx)
    print('\n')
```

Pair (0,1)

Linear Kernel



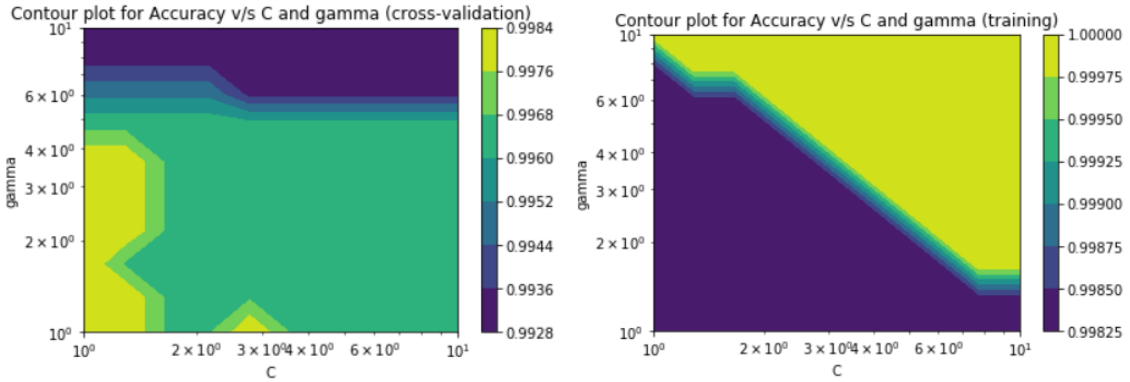
The following hyperparameters were obtained:

```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 1.
0, 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbos
e': False}
```

The comparison:

```
sklearn cross-val score: 0.9983, cvxopt cross-val score: 0.9983
sklearn num of SV: 40, cvxopt num of SV: 40
sklearn indices of support vector: [  7  26  96 115 136 168 177 183 204 216 220 242 251 259 275 293 294 312
 338 342 348 355 361 366 372 381 386 390 392 394 395 415 450 453 470 473
 478 526 528 537]
cvxopt indices of support vector : [  7  26  96 115 136 168 177 183 204 216 220 242 251 259 275 293 294 312
 338 342 348 355 361 366 372 381 386 390 392 394 395 415 450 453 470 473
 478 526 528 537]
```

Rbf kernel



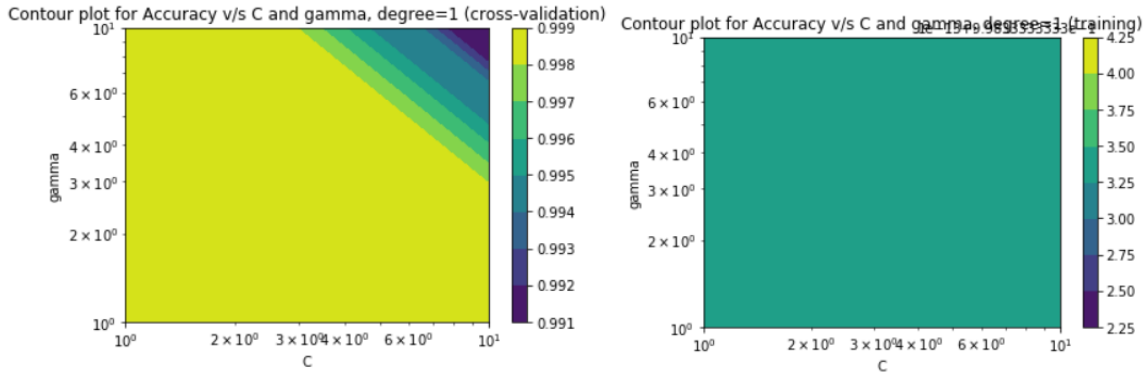
The following hyperparameters were obtained:

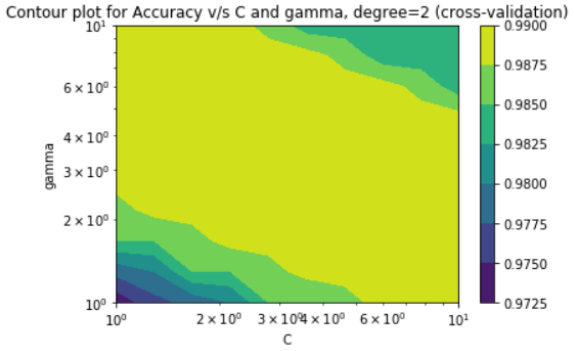
```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 1.
0, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbose': F
alse}
```

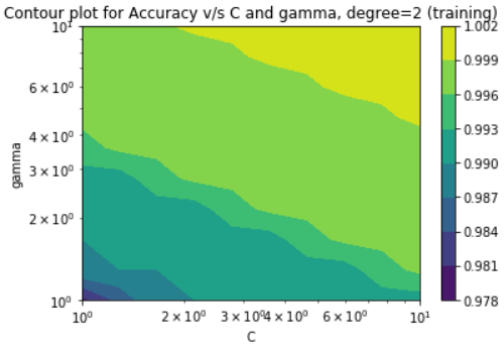
The comparison:

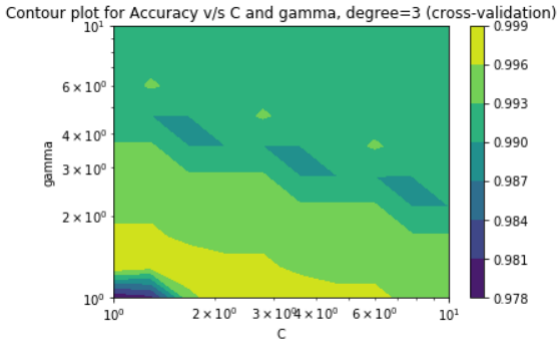
```
sklearn cross-val score: 0.9983, cvxopt cross-val score: 0.9983
sklearn num of SV: 38, cvxopt num of SV: 38
sklearn indices of support vector: [  1   7  26  96 136 168 177 183 204 214 216 242 275 293 294 304 312 342
 348 355 361 366 381 386 390 392 394 395 415 450 453 463 470 473 478 526
 528 537]
cvxopt indices of support vector : [  1   7  26  96 136 168 177 183 204 214 216 242 275 293 294 304 312 342
 348 355 361 366 381 386 390 392 394 395 415 450 453 463 470 473 478 526
 528 537]
```

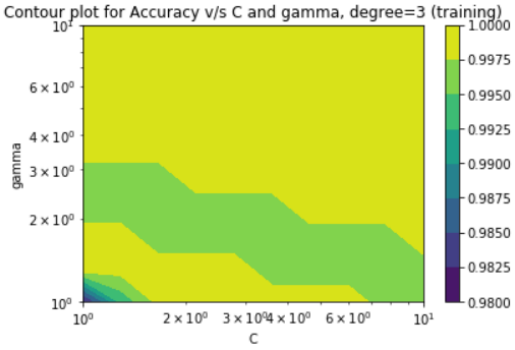
Poly kernel

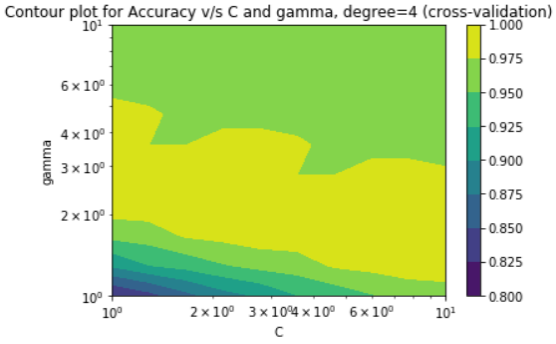


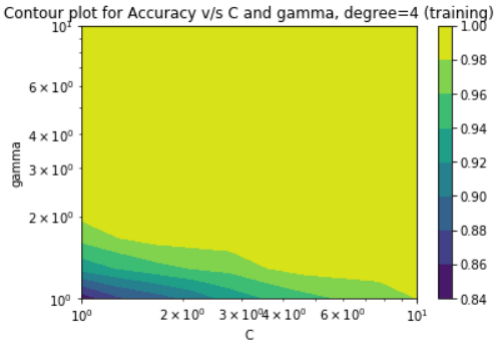












The following hyperparameters were obtained:

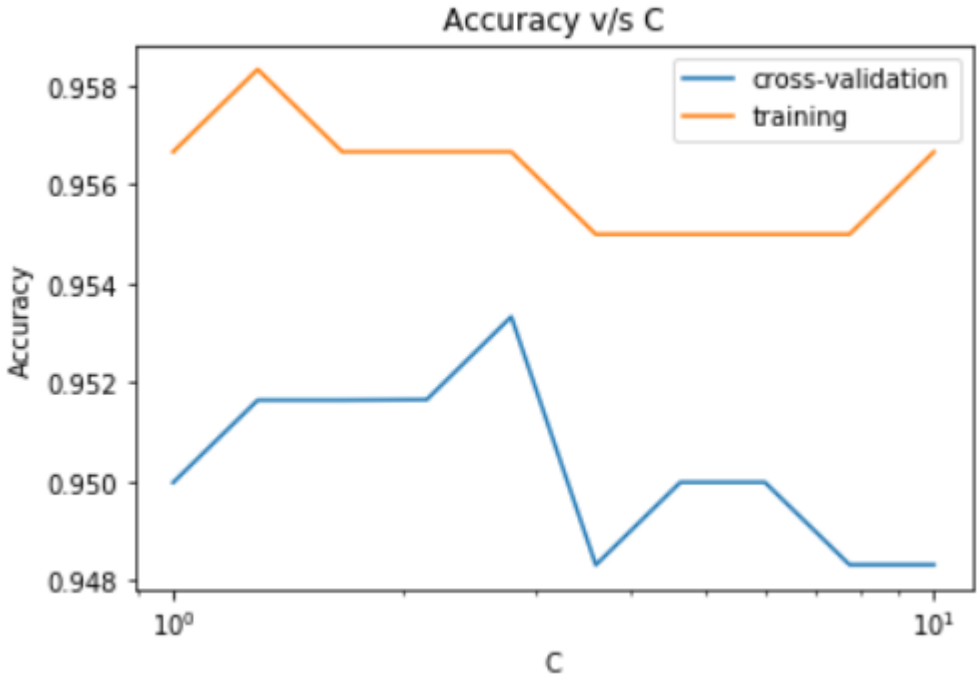
```
{'C': 1.6681005372000588, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
3, 'gamma': 1.2915496650148839, 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tr
ue, 'tol': 0.001, 'verbose': False}
```

The comparison:

```
sklearn cross-val score: 0.9983, cvxopt cross-val score: 0.9967
sklearn num of SV: 118, cvxopt num of SV: 118
sklearn indices of support vector: [  1   7   8  10  20  26  38  46  48  59  60  68  69  75  77  79  80  84
  89  92  93  96 108 115 125 136 142 148 152 160 161 164 165 166 168 171
 173 177 183 191 196 204 205 216 220 226 237 242 244 251 258 259 269 274
 275 284 290 292 293 294 300 303 304 312 317 319 324 326 332 335 338 342
 348 351 355 361 364 366 372 378 381 386 390 392 394 395 399 407 411 414
 415 418 420 421 435 447 449 450 453 463 464 470 473 474 478 491 496 500
 505 516 524 526 528 537 541 560 561 568]
cvxopt indices of support vector : [  1   7   8  10  20  26  38  46  48  59  60  68  69  75  77  79  80  84
  89  92  93  96 108 115 125 136 142 148 152 160 161 164 165 166 168 171
 173 177 183 191 196 204 205 216 220 226 237 242 244 251 258 259 269 274
 275 284 290 292 293 294 300 303 304 312 317 319 324 326 332 335 338 342
 348 351 355 361 364 366 372 378 381 386 390 392 394 395 399 407 411 414
 415 418 420 421 435 447 449 450 453 463 464 470 473 474 478 491 496 500
 505 516 524 526 528 537 541 560 561 568]
```

Pair (2, 3)

Linear kernel



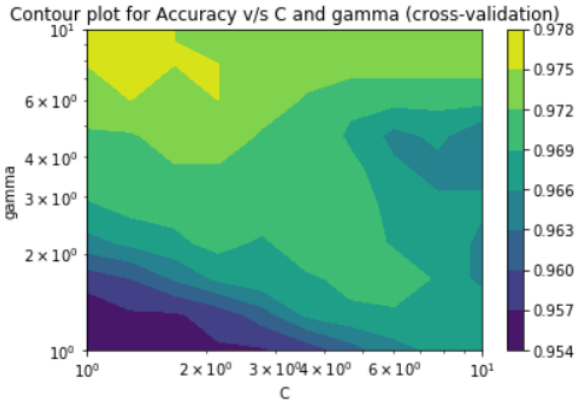
The following hyperparameters were obtained:

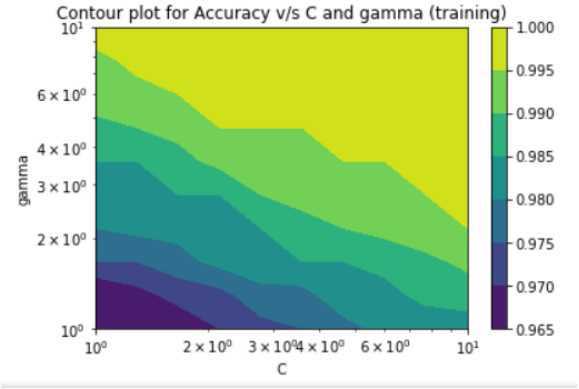
```
{'C': 2.7825594022071245, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 1.0, 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.
001, 'verbose': False}
```

The comparison:

```
sklearn cross-val score: 0.9533, cvxopt cross-val score: 0.9467
sklearn num of SV: 113, cvxopt num of SV: 114
sklearn indices of support vector: [  5   6  10  15  19  20  24  26  29  32  40  51  52  67  75  79  83  85
  86  94  97 107 110 116 118 119 122 128 130 133 136 143 151 159 164 167
 177 201 203 204 205 206 214 217 221 223 224 228 229 230 238 244 254 256
 259 260 276 277 289 291 299 300 302 318 329 331 340 347 350 352 357 363
 370 374 376 386 396 397 399 432 441 445 448 449 455 458 462 479 486 493
 497 502 506 515 523 526 527 528 540 543 550 554 557 558 561 565 570 572
 574 577 582 596 598]
cvxopt indices of support vector : [  5   6  10  15  19  20  24  26  29  32  40  51  52  67  75  79  83  85
  86  94  97 107 110 116 118 119 122 128 130 133 136 143 151 159 164 167
 177 201 203 204 205 206 214 217 221 223 224 228 229 230 238 244 254 256
 259 260 276 277 289 291 295 299 300 302 318 329 331 340 347 350 352 357
 363 370 374 376 386 396 397 399 432 441 445 448 449 455 458 462 479 486
 493 497 502 506 515 523 526 527 528 540 543 550 554 557 558 561 565 570
 572 574 577 582 596 598]
```

Rbf kernel





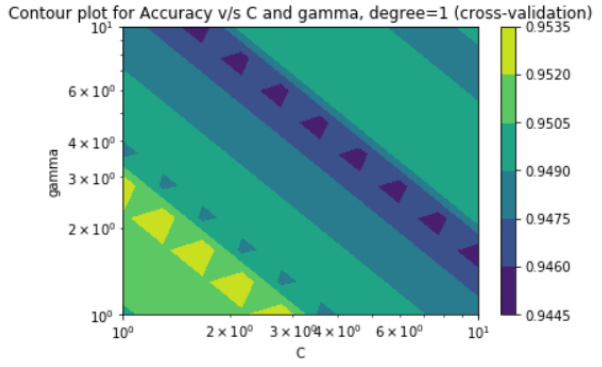
The following hyperparameters were obtained:

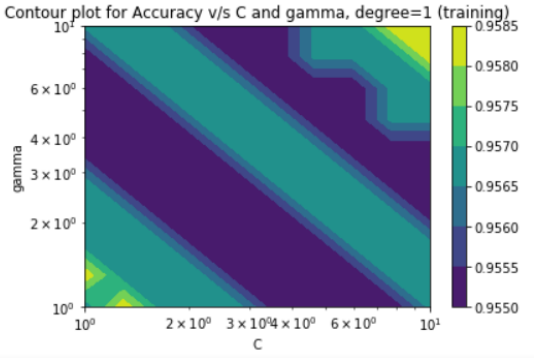
```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 10.
0, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001, 'verbose': F
alse}
```

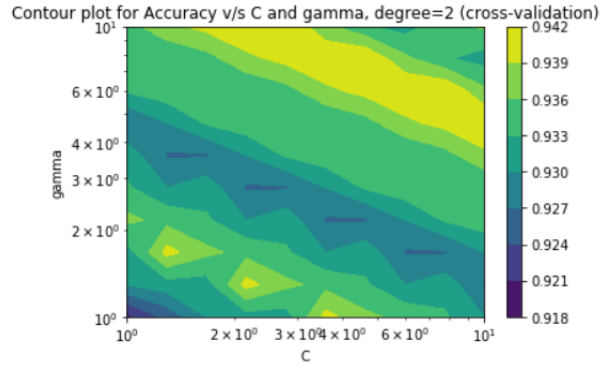
The comparison:

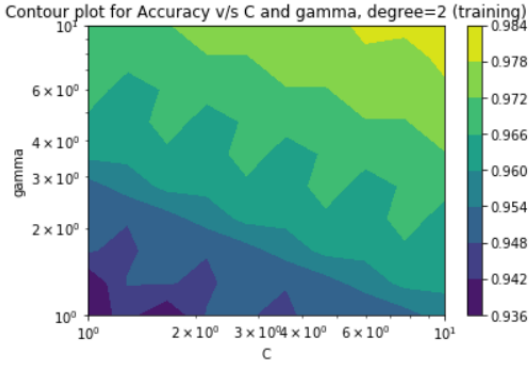
```
sklearn cross-val score: 0.9767, cvxopt cross-val score: 0.9767
sklearn num of SV: 274, cvxopt num of SV: 277
sklearn indices of support vector: [  4   8   9  10  12  17  18  19  21  22  26  29  30  32  36  39  40  41
  47  48  49  50  51  52  57  59  60  62  63  66  67  71  72  73  75  76
  77  79  80  81  82  83  85  86  91  92  93  94  95  97  98  99 100 101
 105 107 109 110 113 116 117 118 119 122 125 126 127 128 130 131 132 133
 134 135 136 138 143 146 148 150 151 152 155 157 159 161 163 165 166 167
 175 177 178 183 185 188 189 190 191 196 199 202 203 204 205 206 207 208
 211 212 214 215 216 217 218 220 221 223 224 228 229 230 235 237 243 244
 245 248 249 251 254 256 259 260 264 267 269 276 277 289 291 298 299 300
 302 304 305 310 312 314 315 316 318 320 322 323 326 328 330 331 338 340
 344 345 347 348 349 350 351 352 357 363 366 367 370 371 372 374 376 377
 381 382 383 386 387 388 389 391 392 394 395 397 399 400 403 404 406 411
 413 414 416 418 419 422 432 434 435 439 441 444 445 446 448 449 450 455
 457 460 461 462 463 468 473 474 475 477 478 479 484 487 488 490 491 492
 493 497 501 502 506 507 510 515 519 521 526 527 528 534 537 539 540 541
 545 547 550 554 555 557 558 560 564 566 567 568 569 572 573 574 576 577
 583 585 591 596]

cvxopt indices of support vector : [  4   8   9  10  12  17  18  19  21  22  26  29  30  32  36  39  40  41
  47  48  49  50  51  52  56  57  59  60  62  63  66  67  71  72  73  75
  76  77  79  80  81  82  83  85  86  91  92  93  94  95  97  98  99 100
 101 105 107 109 110 113 116 117 118 119 122 125 126 127 128 130 131 132
 133 134 135 136 138 143 146 148 150 151 152 155 157 159 161 163 165 166
 167 170 175 177 178 183 185 188 189 190 191 196 199 202 203 204 205 206
 207 208 211 212 214 215 216 217 218 220 221 223 224 228 229 230 235 237
 243 244 245 248 249 251 254 256 259 260 264 267 269 276 277 289 291 298
 299 300 302 304 305 310 312 314 315 316 318 320 322 323 326 328 330 331
 338 340 344 345 347 348 349 350 351 352 357 363 366 367 370 371 372 374
 376 377 381 382 383 386 387 388 389 391 392 394 395 397 399 400 403 404
 406 411 413 414 416 418 419 422 432 434 435 439 441 444 445 446 448 449
 450 455 457 460 461 462 463 468 473 474 475 477 478 479 484 487 488 490
 491 492 493 497 501 502 506 507 510 515 519 521 526 527 528 534 537 539
 540 541 545 547 550 554 555 557 558 560 564 566 567 568 569 572 573 574
 576 577 583 585 591 594 596]
```

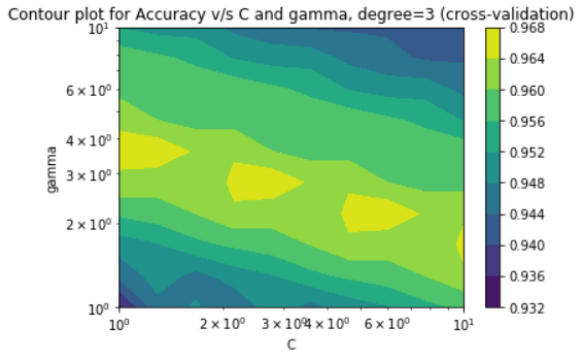
Poly kernel

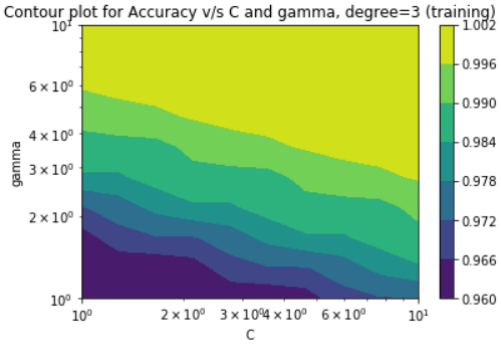


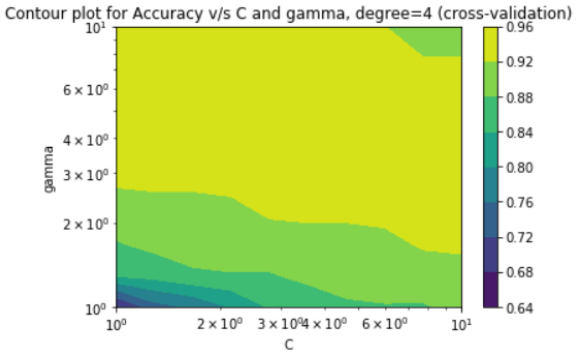


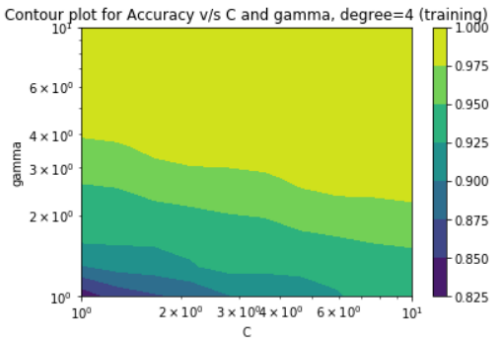












The following hyperparameters were obtained:

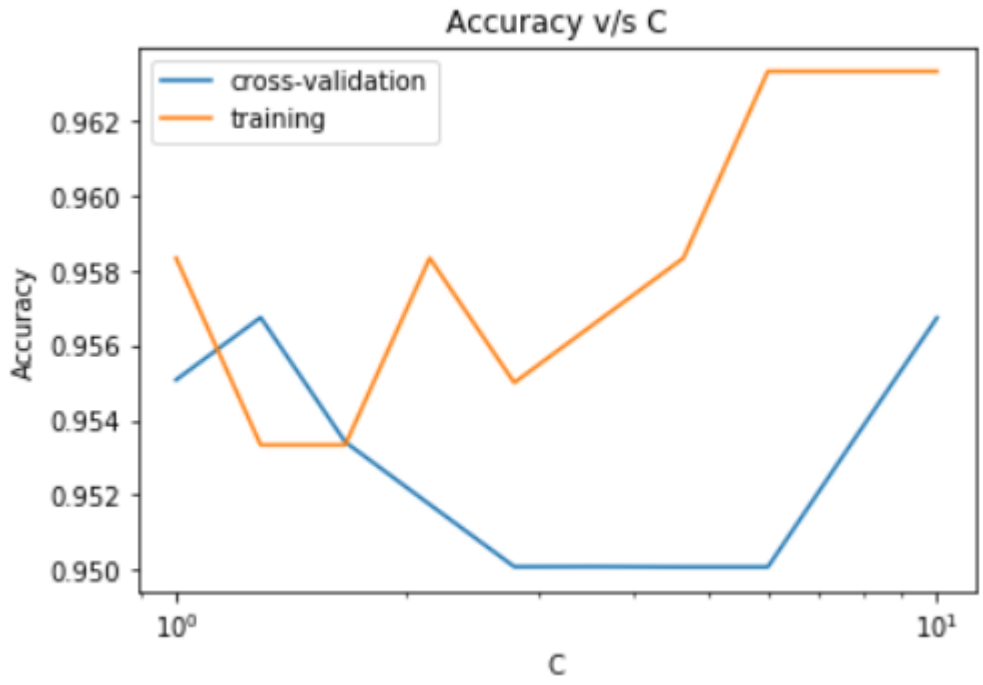

```
{'C': 1.2915496650148839, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
3, 'gamma': 3.5938136638046276, 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tr
ue, 'tol': 0.001, 'verbose': False}
```


The comparison:

```
sklearn cross-val score: 0.9667, cvxopt cross-val score: 0.9633
sklearn num of SV: 143, cvxopt num of SV: 143
sklearn indices of support vector: [  9  10  19  22  26  29  32  42  46  50  51  52  53  67  75  79  82  83
  85  86  91  96  98 107 113 115 116 118 119 122 127 130 132 141 143 159
 163 167 171 174 177 178 202 203 204 205 206 208 214 216 217 219 221 223
 224 226 228 230 236 238 245 251 254 256 258 259 264 274 276 277 289 297
 299 300 301 302 312 315 318 319 320 323 325 331 336 341 347 349 350 366
 367 369 370 373 374 380 383 386 391 393 397 414 421 432 435 439 441 445
 448 450 455 461 462 464 473 481 487 492 493 494 497 502 506 510 517 519
 523 528 540 554 557 558 564 568 572 576 577 578 584 590 593 596 598]
cvxopt indices of support vector : [  9  10  19  22  26  29  32  42  46  50  51  52  53  67  75  79  82  83
  85  86  91  96  98 107 113 115 116 118 119 122 127 130 132 141 143 159
 163 167 171 174 177 178 202 203 204 205 206 208 214 216 217 219 221 223
 224 226 228 230 236 238 245 251 254 256 258 259 264 274 276 277 289 297
 299 300 301 302 312 315 318 319 320 323 325 331 336 341 347 349 350 366
 367 369 370 373 374 380 383 386 391 393 397 414 421 432 435 439 441 445
 448 450 455 461 462 464 473 481 487 492 493 494 497 502 506 510 517 519
 523 528 540 554 557 558 564 568 572 576 577 578 584 590 593 596 598]
```

Pair (4, 5)

Linear kernel



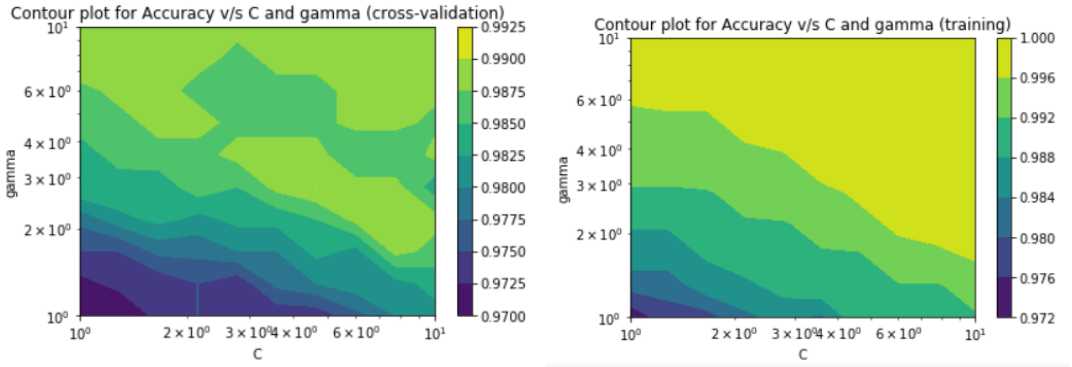
The following hyperparameters were obtained:

```
{'C': 1.2915496650148839, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 1.0, 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.
001, 'verbose': False}
```

The comparison:

```
sklearn cross-val score: 0.9567, cvxopt cross-val score: 0.9567
sklearn num of SV: 136, cvxopt num of SV: 136
sklearn indices of support vector: [  1   3   8  10  12  20  25  29  31  35  37  40  42  44  56  57  63  72
  82  88  98 104 106 107 116 119 125 136 138 139 140 144 146 147 156 162
 163 166 167 172 176 181 194 196 201 207 208 211 212 225 227 230 232 233
 240 242 248 250 253 260 265 276 284 285 293 298 301 306 309 310 311 313
 315 320 324 329 337 341 343 348 350 355 357 365 367 374 379 383 390 398
 406 412 418 419 420 428 432 450 452 456 461 466 470 479 482 483 484 487
 488 491 494 496 497 502 505 508 516 517 520 522 523 524 525 531 544 545
 546 548 568 574 578 579 584 586 587 597]
cvxopt indices of support vector : [  1   3   8  10  12  20  25  29  31  35  37  40  42  44  56  57  63  72
  82  88  98 104 106 107 116 119 125 136 138 139 140 144 146 147 156 162
 163 166 167 172 176 181 194 196 201 207 208 211 212 225 227 230 232 233
 240 242 248 250 253 260 265 276 284 285 293 298 301 306 309 310 311 313
 315 320 324 329 337 341 343 348 350 355 357 365 367 374 379 383 390 398
 406 412 418 419 420 428 432 450 452 456 461 466 470 479 482 483 484 487
 488 491 494 496 497 502 505 508 516 517 520 522 523 524 525 531 544 545
 546 548 568 574 578 579 584 586 587 597]
```

Rbf kernel



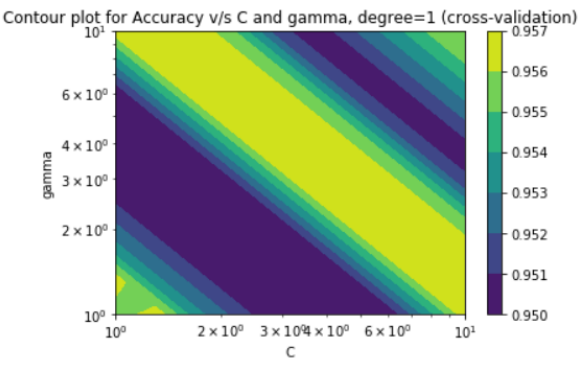
The obtained hyperparameters:

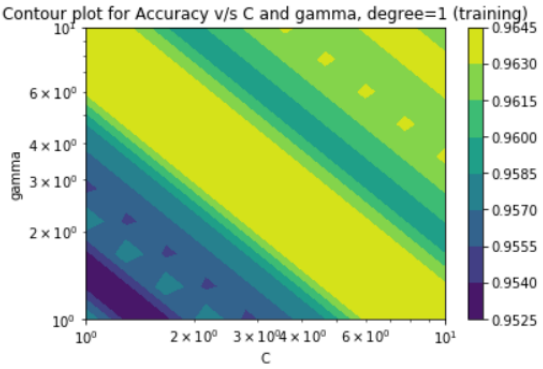
```
{'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 1, 'gamma': 7.7
42636826811269, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.001,
'verbose': False}
```

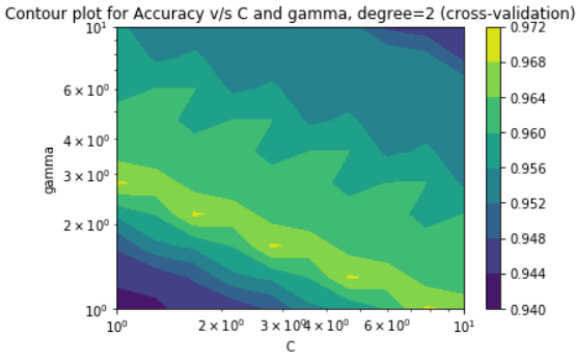
The comparison:

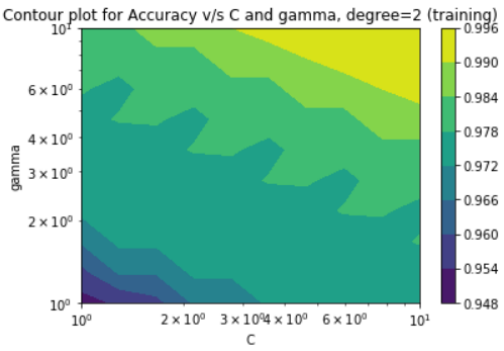
```
sklearn cross-val score: 0.9900, cvxopt cross-val score: 0.9900
sklearn num of SV: 185, cvxopt num of SV: 191
sklearn indices of support vector: [  0   4   6   7   8  10  19  20  21  25  29  35  36  40  43  44  46  51
  56  62  63  67  72  73  81  82  88  89  93 100 104 110 112 120 124 125
 129 136 138 139 142 144 147 162 163 164 167 172 173 174 177 181 187 190
 194 197 201 207 211 212 217 219 222 224 227 232 235 236 238 240 245 248
 250 252 253 259 260 265 266 274 276 279 281 284 288 295 296 298 301 303
 304 306 311 313 315 319 320 324 325 327 329 337 338 343 344 347 348 349
 352 355 356 357 359 362 364 365 372 373 374 379 381 383 392 393 394 400
 406 409 412 414 418 420 421 423 426 428 429 433 438 439 443 448 450 451
 452 456 457 458 461 466 468 476 479 481 482 484 487 488 491 494 497 505
 510 512 516 517 520 522 524 531 532 536 540 541 544 548 553 559 568 579
 587 591 594 595 597]
cvxopt indices of support vector : [  0   4   6   7   8  10  14  15  19  20  21  25  26  29  35  36  40  43
  44  46  51  56  62  63  67  72  73  81  82  88  89  93 100 104 110 112
 120 124 125 129 136 138 139 142 144 147 162 163 164 167 172 173 174 177
 181 187 190 194 197 201 207 211 212 217 219 222 224 227 232 235 236 238
 240 245 248 250 252 253 259 260 265 266 274 276 279 281 284 288 295 296
 298 301 303 304 306 311 313 315 319 320 324 325 327 329 337 338 343 344
 347 348 349 352 355 356 357 359 362 364 365 372 373 374 379 380 381 383
 392 393 394 400 406 409 412 414 418 420 421 423 426 428 429 433 438 439
 443 448 450 451 452 456 457 458 461 466 468 472 476 479 481 482 484 487
 488 491 494 495 497 505 510 512 516 517 520 522 524 531 532 536 540 541
 544 548 553 559 568 579 587 591 594 595 597]
```

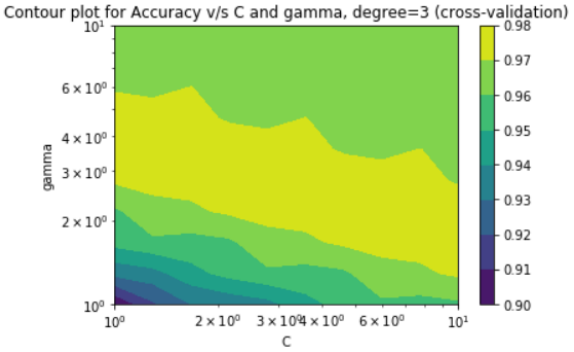
Poly kernel

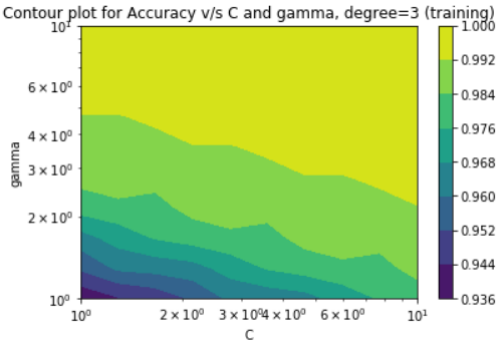


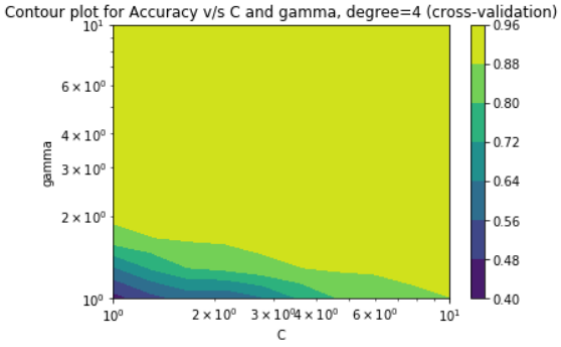


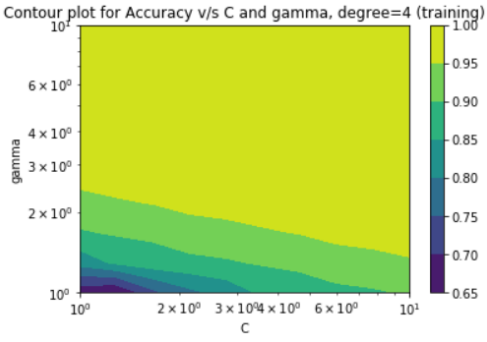












The obtained hyperparameters:


```
{'C': 1.2915496650148839, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
3, 'gamma': 3.5938136638046276, 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tr
ue, 'tol': 0.001, 'verbose': False}
```


The comparison:

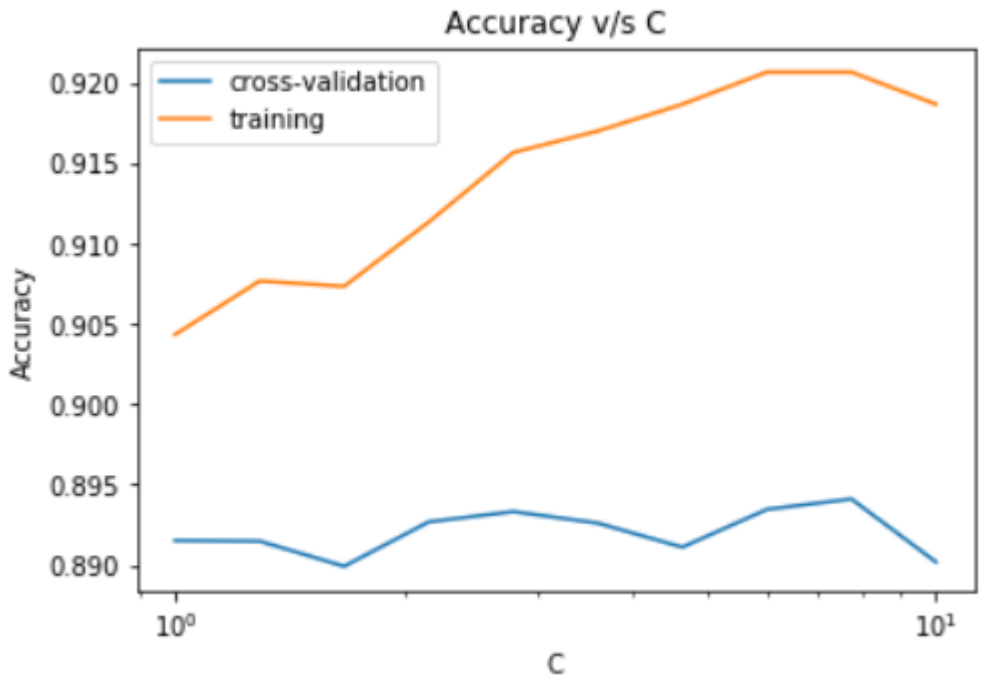
```
sklearn cross-val score: 0.9784, cvxopt cross-val score: 0.9784
sklearn num of SV: 139, cvxopt num of SV: 139
sklearn indices of support vector: [  0   2  10  11  20  25  35  37  40  44  45  56  63  72  82  87  88  90
  91  96 104 107 110 111 123 124 125 130 137 138 139 144 146 147 156 158
 162 163 164 166 167 172 177 184 194 201 207 208 211 212 227 232 239 240
 248 250 253 260 276 284 296 298 301 303 304 306 309 310 311 313 315 324
 329 331 337 343 347 348 349 352 356 357 367 380 381 390 393 394 406 410
 412 419 420 424 426 428 433 447 450 456 460 461 466 470 479 482 484 485
 487 488 490 491 494 497 499 505 508 516 517 520 523 524 531 536 541 544
 545 546 548 555 559 571 579 584 587 590 594 597 598]
cvxopt indices of support vector : [  0   2  10  11  20  25  35  37  40  44  45  56  63  72  82  87  88  90
  91  96 104 107 110 111 123 124 125 130 137 138 139 144 146 147 156 158
 162 163 164 166 167 172 177 184 194 201 207 208 211 212 227 232 239 240
 248 250 253 260 276 284 296 298 301 303 304 306 309 310 311 313 315 324
 329 331 337 343 347 348 349 352 356 357 367 380 381 390 393 394 406 410
 412 419 420 424 426 428 433 447 450 456 460 461 466 470 479 482 484 485
 487 488 490 491 494 497 499 505 508 516 517 520 523 524 531 536 541 544
 545 546 548 555 559 571 579 584 587 590 594 597 598]
```

Multiclass Classification (using all 25 features):

The following code does the analyses

```
# Part 1 (Multiclass Classification) using all 25 features
for kernel in kernels:
    print('Kernel: %s'%(kernel))
    clf_p1, cvscore_p1 = findParameters([kernel], X, t)
    print("Score: %0.4f (+/- %0.4f)" % (cvscore_p1.mean(), 2*cvscore_p1.std()))
    print(clf_p1.get_params())
```

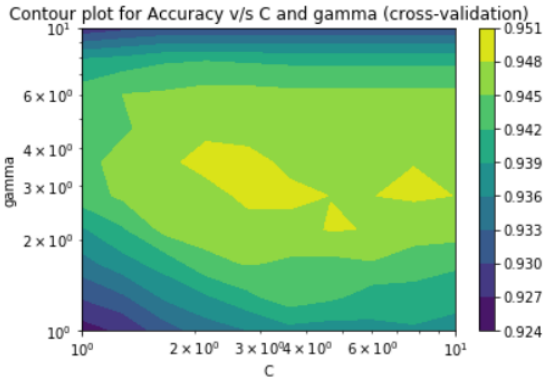
Linear Kernel:

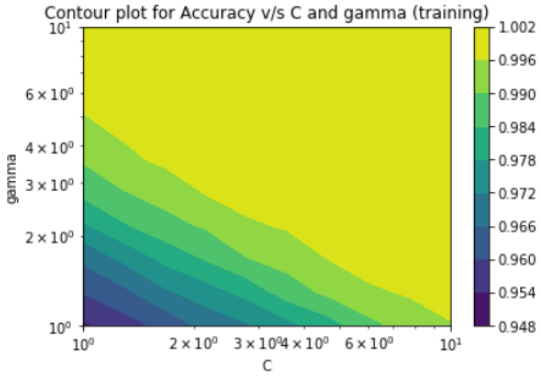


Obtained hyperparameters:

```
Score: 0.8941 (+/- 0.0311)
{'C': 7.742636826811269, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 1.0, 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.
001, 'verbose': False}
```

Rbf kernel

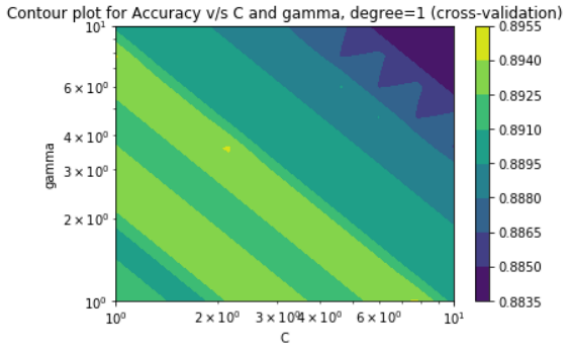


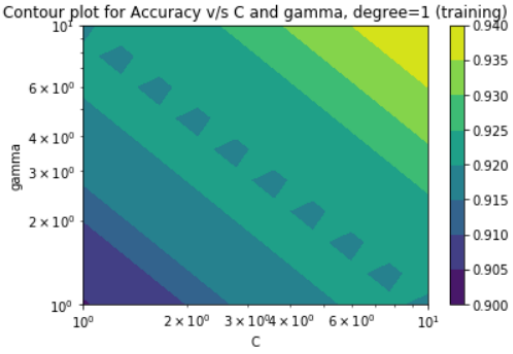


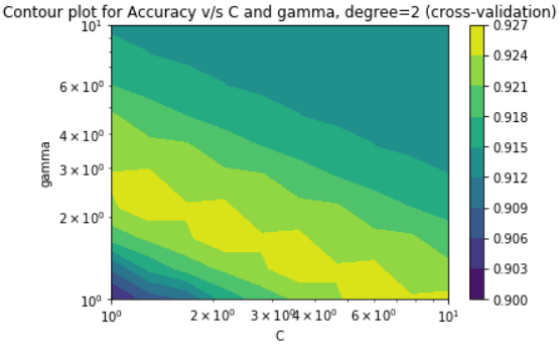
Obtained hyperparameters:

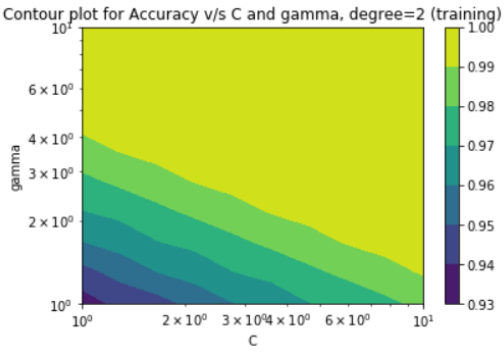
```
Score: 0.9493 (+/- 0.0122)
{'C': 2.154434690031884, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 3.5938136638046276, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tru
e, 'tol': 0.001, 'verbose': False}
```

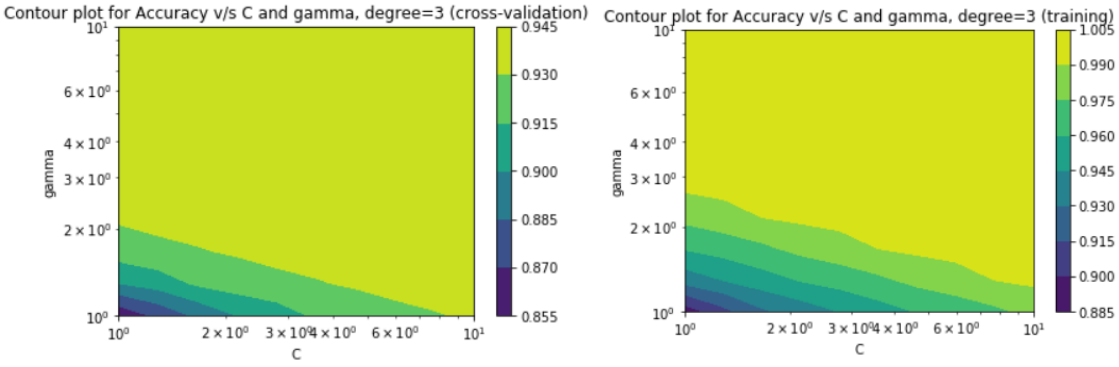
Poly kernel

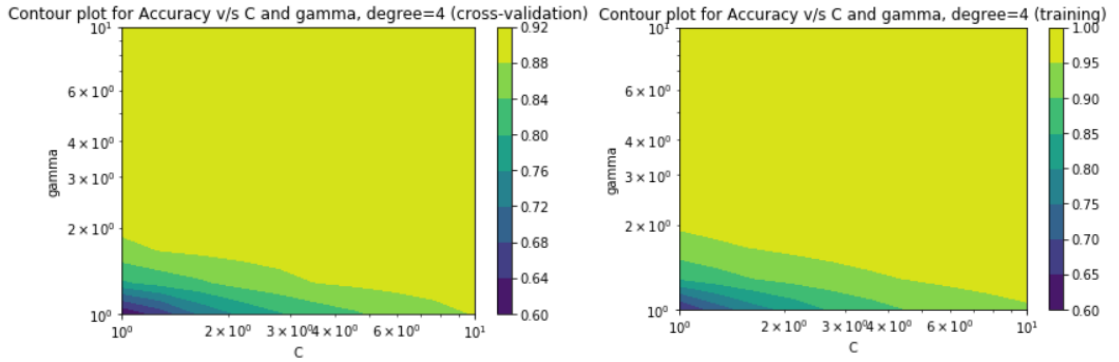












Obtained hyperparameters:

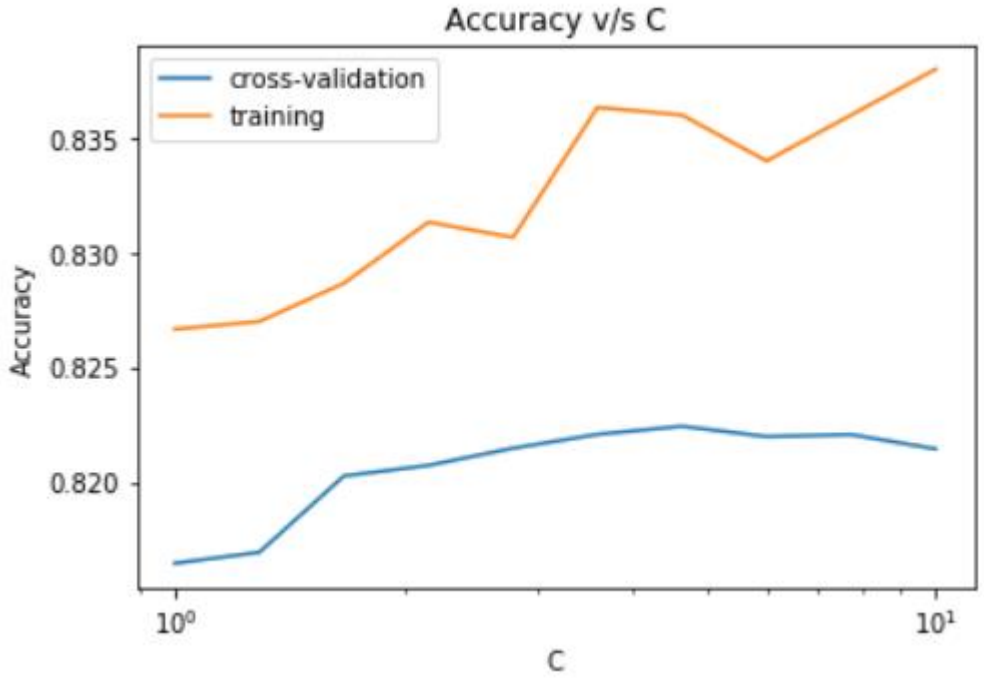
```
Score: 0.9411 (+/- 0.0112)
{'C': 1.6681005372000588, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
3, 'gamma': 2.7825594022071245, 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tr
ue, 'tol': 0.001, 'verbose': False}
```

Multiclass Classification (using 10 features):

The following code does the analyses:

```
# Part 1 (Multiclass Classification) using 10 features
for kernel in kernels:
    print('Kernel: %s'%(kernel))
    clf_p1, cvscore_p1 = findParameters([kernel], X[:,:10], t)
    print("Score: %0.4f (+/- %0.4f)" % (cvscore_p1.mean(), 2*cvscore_p1.std()))
    print(clf_p1.get_params())
```

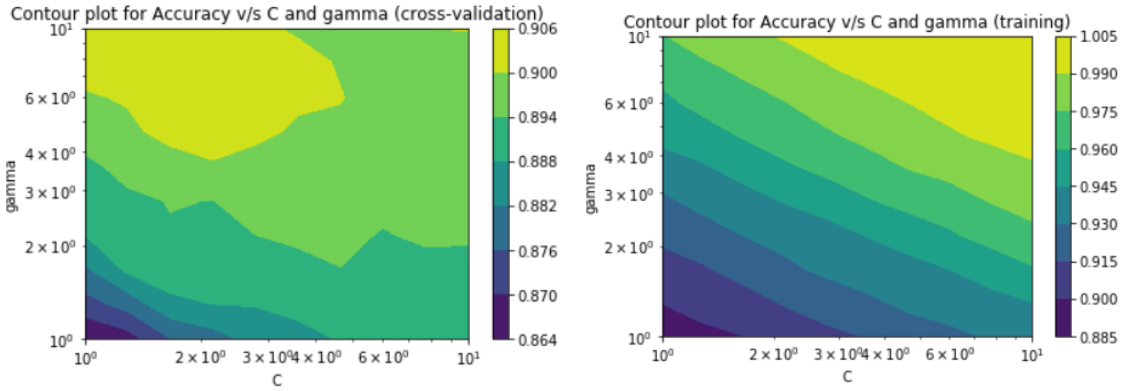
Linear kernel



Obtained hyperparameters:

```
Score: 0.8224 (+/- 0.0250)
{'C': 4.641588833612778, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 1.0, 'kernel': 'linear', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': True, 'tol': 0.
001, 'verbose': False}
```

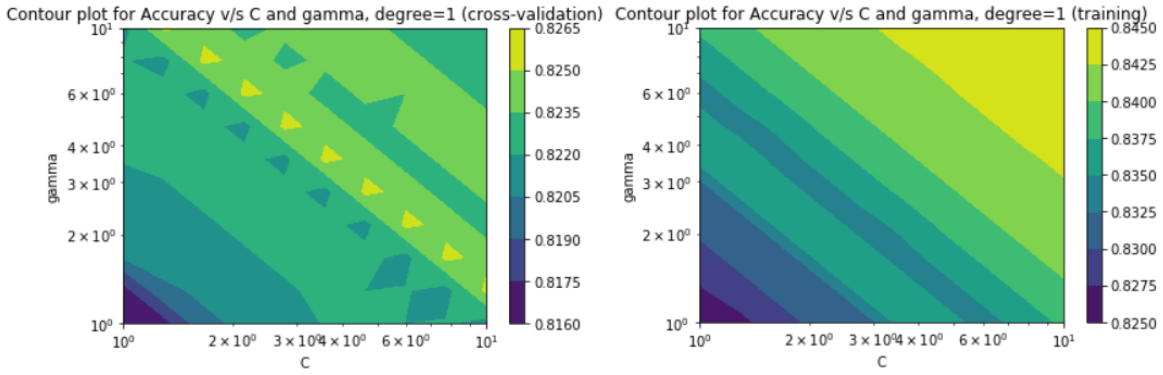
Rbf kernel

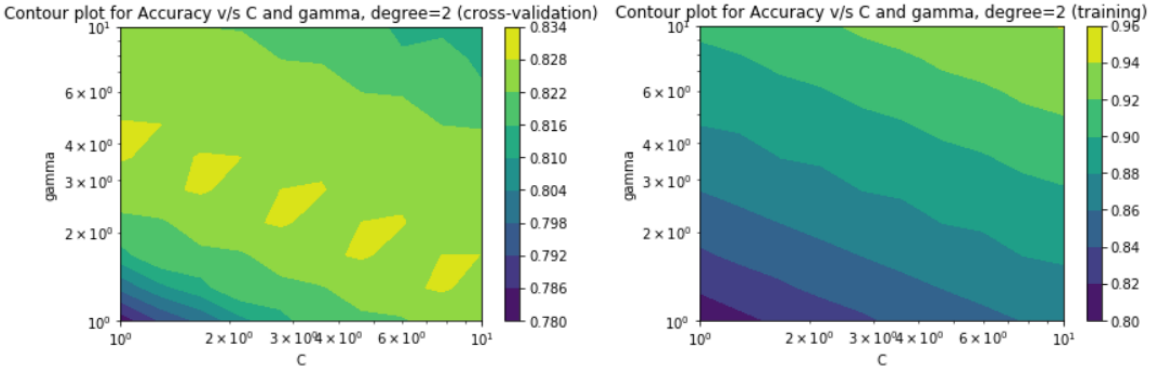


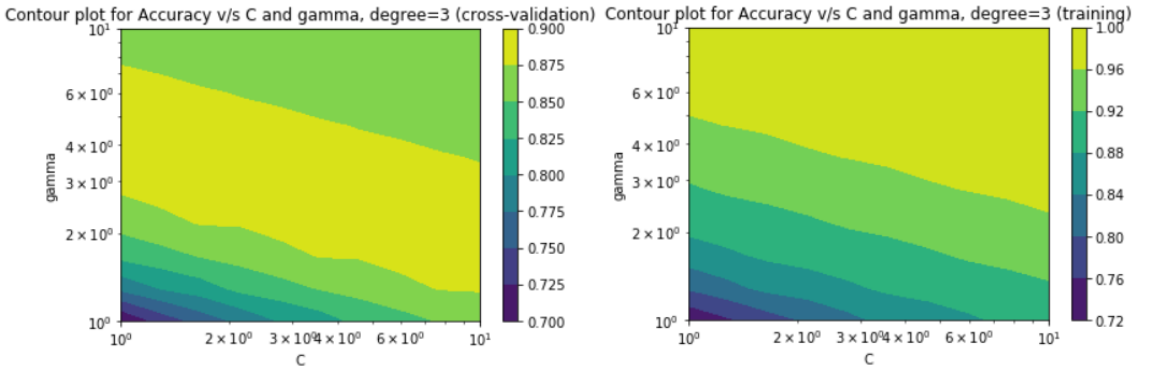
Obtained hyperparameters:

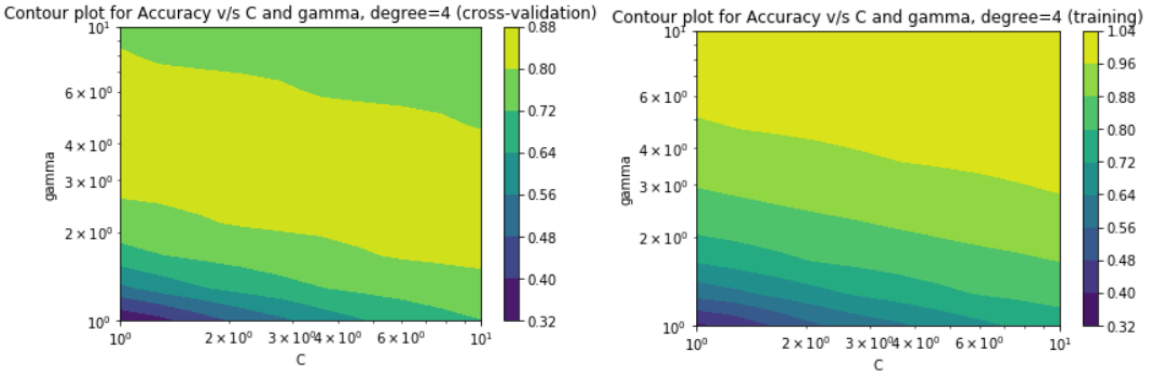
```
Score: 0.9058 (+/- 0.0121)
{'C': 2.154434690031884, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 7.742636826811269, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tru
e, 'tol': 0.001, 'verbose': False}
```

Poly kernel









Obtained hyperparameters:

```
Score: 0.8869 (+/- 0.0102)
{'C': 1.6681005372000588, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
3, 'gamma': 3.5938136638046276, 'kernel': 'poly', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tr
ue, 'tol': 0.001, 'verbose': False}
```

Part2: Training and Prediction (Kaggle)

The following code is used for training:

```
# Part 2 (training)
clf_p2, cvscore_p2 = findParameters(kernels, train_X, train_t)
print("Score: %0.4f (+/- %0.4f)" % (cvscore_p2.mean(), 2*cvscore_p2.std()))
print(clf_p2.get_params())
```

The following parameters were obtained:

```
Score: 0.9720 (+/- 0.0078)
{'C': 5.994842503189409, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree':
1, 'gamma': 4.641588833612778, 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': None, 'shrinking': Tru
e, 'tol': 0.001, 'verbose': False}
```

The following code does the prediction:

```
# Part 2 (prediction)
prediction_t = clf_p2.predict(prediction_X)
print(prediction_t)
df = pd.DataFrame({'class': prediction_t})
df.to_csv('prediction.csv', index_label = 'id')
```