1 Exercise 1

In the general case of solving a linear SVM with slack variables without a regularizer, the objective function is:

$$\begin{aligned}
\underset{\alpha}{\text{minimize}} \quad & -\sum_{i=1}^{n} \alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)}) && \text{(1a)} \\
\text{subject to} \quad & 0 \le \alpha_i \le C \quad \forall i\,, && \text{(1b)} \\
& \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 && \text{(1c)}
\end{aligned}$$

Using the Python CVXOPT package, the general form of the objective function is:

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & \frac{1}{2} x^T P x + q^T x && \text{(2a)} \\
\text{subject to} \quad & Gx \le h && \text{(2b)} \\
& Ax = b && \text{(2c)}
\end{aligned}$$

The general form for converting our slack variable objective function in Equation 1 to the CVXOPT objective function in Equation 2 is described in Table 1.

CVXOPT	Conversion from Equation 1
x	If there are n points in the training set, an $n \times 1$ vector equal to the values of x in the training data
P	An $n \times n$ matrix which is the kernel matrix between all pairs of training data x weighted by the corresponding value of y from the training data
q	An $n \times 1$ vector of -1s
G	A $2n \times n$ matrix where the top $n \times n$ is the identity matrix and the bottom $n \times n$ is the negative identity matrix
h	A $2n \times 1$ vector with the top $n \times 1$ vector of Cs and the bottom $n \times 1$ vector of 0s
A	An $1 \times n$ vectors with elements $y^{(i)}$ from the training set for all values of i
b	An 1×1 vector of 0s

Table 1: Conversion rule for deriving CVXOPT constraints

For the small example with $(1, 2), (2, 2)$ as positive examples and $(0, 0), (-2, 3)$ as negative examples, the constraints from CVXOPT as written above are written in Equation 3.

$$P = \begin{bmatrix} 5 & 6 & 0 & -4 \\ 6 & 8 & 0 & -2 \\ 0 & 0 & 0 & 0 \\ -4 & -2 & 0 & 13 \end{bmatrix} \qquad q = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \qquad h = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \end{bmatrix}$$

The decision boundary generated by the SVM code for the small example is shown in Figure 1.

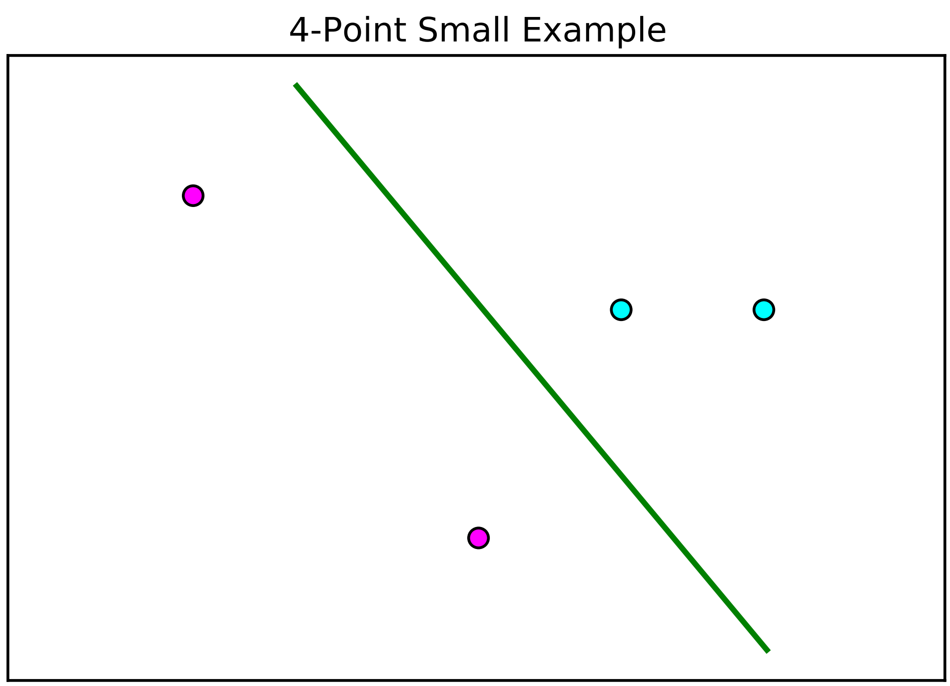

Figure 1: Decision boundary for 4 points using CVXOPT and SVM with slack variables

Setting $C = 1$,

The error rates for the training and validation sets is shown in Table 2.

1.2 - since smallOverlap was the same, took half the data for test and half for train

2.2: * do training set over some values of lambda * use that to pick lambda * test on 'validation' data and report an error value

2.3: will need to plot lambda versus sparsity

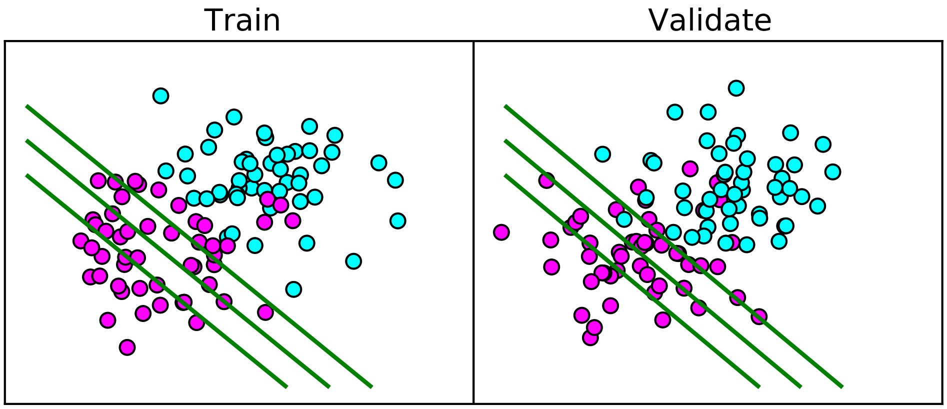

Figure 2: smallOverlap

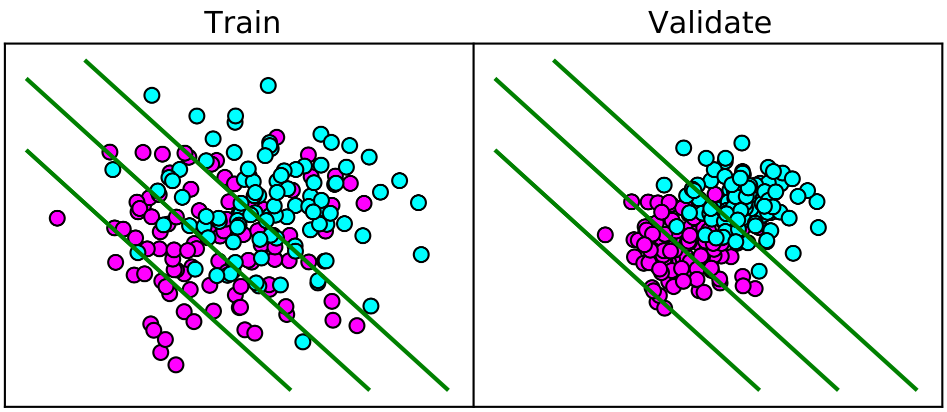

Figure 3: bigOverlap

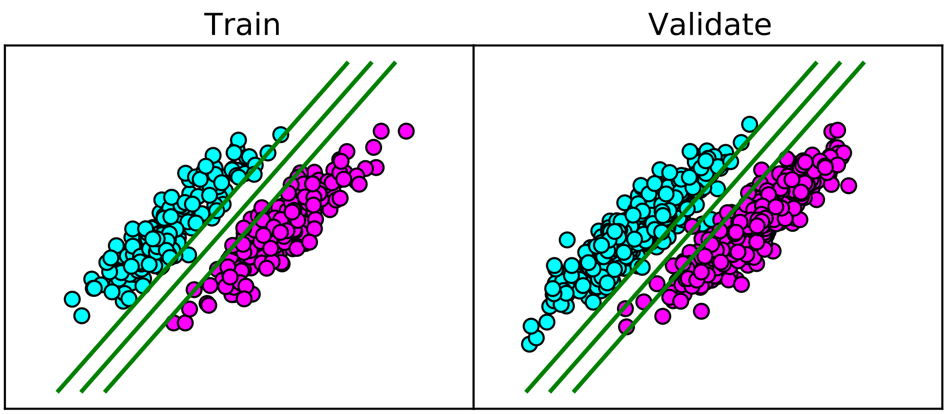

Figure 4: ls

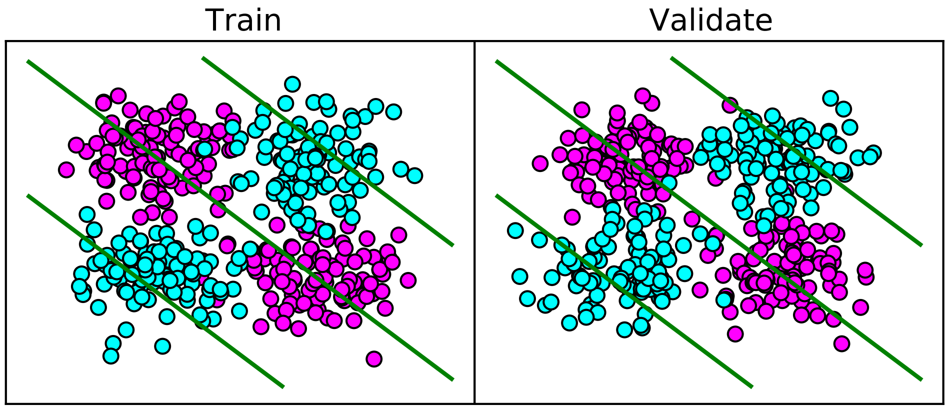

Figure 5: nonSep2

	Training	Validation
smallOverlap	.24	.24
bigOverlap	.305	.255
ls	0	0.00375
nonSep2	.485	.495

Table 2: Error rates for training and validation sets

2.4: * do training set over some values of bandwidth with lambda = 0 * do validation set over some values of lambda * test on test and report results

2.5: * will need to think about this a little bit more.

3.1: * https://piazza.com/class/hzdfawvtilo7hf?cid=434 * L2 regularization, linear case * http://blog.datumbox.com/ma learning-tutorial-the-multinomial-logistic-regression-softmax-regression/

* maybe need to save coefficients or predictions or something?

3.1 - LR: * 2 features, 100 pts, 2 classes, l = 0.01 -¿ error = .485 * might need to randomly select subsets of data points

* bigOverlap - error is .26 on test, ,.305 on training with l = 0.1 * smallOverlap - error is .26 on test, .27 on training with l = 0.1

* tips for getting numerics to work better: * normalize data beforehand (then don't forget to re-normalize later) *

3.2 - multiclass SVM: * used http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html * can do this SO FAST it doesn't even make sense to try to do this by myself. * tried an array of l values with L1 loss (multiclasssvm.py), best L with random partitioning of data into 3 sets. rigorous stopping criteria * hinge loss, l2 regularization: * validation error: .187 * test error: .261 * l: 0.01 * squared loss, l1 regularization: * validation error: .143 * test error: .143 * l: 7e-7