# Machine Learning and Computational Statistics
# Homework 5: Generalized Hinge Loss and Multiclass SVM

April 11, 2017

## 1 Introduction

## 2 Convex Surrogate Loss Functions

### 2.1 Hinge loss is a convex surrogate for 0/1 loss

(a) For any example $(x, y) \in X \times \{-1, 1\}$, show that $1(y \neq sign(f(x)) \leq max\{0, 1 - yf(x)\}$.

**ANSWER**

If $y \neq sign(f(x))$, $yf(x) \leq 0$, and $1 - yf(x) \geq 1$ therefore, the inequality holds,

If $y = sign(f(x))$, $lhs = 0$ and $rhs \geq 0$ therefore, the inequality holds,

(b) Show that the hinge loss $max\{0, 1 - m\}$ is a convex function of the margin $m$.

**ANSWER**

$f_1(x) = 0$, $f_2(x) = 1 - m$ are convex, so according to the result given their pointwise maximum $f(x) = max\{0, 1 - m\}$ is also convex.

(c) Suppose our prediction score functions are given by $f_w(x) = w^T x$. The hinge loss of $f_w$ on any example $(x, y)$ is then $max\{0, 1 - yw^T x\}$. Show that this is a convex function of w.

**ANSWER**

$f_w(x)$ is an affine function and is a convex function of w. Similarly, $1 - yw^T x$ is also a convex function as it is affine, so the hinge loss $max\{0, 1 - yw^T x\}$ is also a convex function as it is the pointwise maximum of 2 convex functions.

## 2.2 Multiclass Hinge Loss

(1) Suppose we have chosen an $h \in \mathcal{H}$, from which we get $f(x) = argmax_{y \in \mathcal{Y}} h(x, y)$. Justify that for any $x \in X$ and $y \in Y$, we have $h(x, y) \leq h(x, f(x))$.

**ANSWER**

For any $x \in X$ and $y \in Y$,

$h(x, f(x)) = max_{y \in \mathcal{Y}}(h(x, y))$

So, by definition $h(x, f(x)) \geq (h(x, y))$

(2) Justify the following two inequalities:

$$\begin{aligned} \Delta(y, f(x)) &\leq \Delta(y, f(x)) + h(x, f(x)) - h(x, y) \\ &\leq \max_{y' \in \mathcal{Y}} [\Delta(y, y')) + h(x, y') - h(x, y)] \end{aligned}$$

The RHS of the last expression is called the **generalized hinge loss**:

$$\ell(h, (x, y)) = \max_{y_0 \in \mathcal{Y}} [\Delta(y, y_0)) + h(x, y_0) - h(x, y)]$$

We have shown that for any $x \in \mathcal{X}, y \in \mathcal{Y}, h \in \mathcal{H}$ we have

$$\ell(h, (x, y)) \geq \Delta(y, f(x)),$$

where, as usual, $f(x) = arg \max_{y \in \mathcal{Y}} h(x, y)$. [You should think about why we cannot write the generalized hinge loss as $\ell(f, (x, y))$.]

**ANSWER**

Using the solution from the previous part,
$h(x, f(x)) \geq (h(x, y))$
$h(x, f(x)) - h(x, y) \geq 0$

$\therefore \Delta(y, f(x)) + h(x, f(x)) - h(x, y) \geq \Delta(y, f(x))$

In the second inequality we are replacing f(x) with y′ which would maximize the expression, so it can be written as,
$\Delta(y, f(x)) + h(x, f(x)) - h(x, y) \leq \max_{f \in \mathcal{F}} [\Delta(y, f(x)) + h(x, f(x)) - h(x, y)]$

$\leq \max_{y' \in \mathcal{Y}} [\Delta(y, y')) + h(x, y') - h(x, y)]$

(3) We now introduce a specific base hypothesis space H of linear functions. Consider a class sensitive feature mapping $\Psi : X \times Y \mapsto \mathbf{R}^d$, and $\mathcal{H} = \{h_w(x,y) = \langle w, \Psi(x,y)\rangle | w \in \mathbf{R}^d\}$. Show that we can write the generalized hinge loss for $h_w(x,y)$ on example $(x_i, y_i)$ as

$$\ell(h_w, (x_i, y_i)) = \max_{y \in \mathcal{Y}} [\Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i)\rangle].$$

**ANSWER**

$\ell(h_w, (x_i, y_i)) = \max_{y \in \mathcal{Y}} [\Delta(y_i, y) + h(x_i, y) - h(x_i, y_i)]$

Now since $h_w(x,y) = \langle w, \Psi(x,y)$
$= \max_{y \in \mathcal{Y}} [\Delta(y_i, y) + \langle w, \Psi(x_i, y)\rangle - \langle w, \Psi(x_i, y_i)\rangle]$

Using the linearity property of inner product,
$= \max_{y \in \mathcal{Y}} [\Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i)\rangle]$

(4) We will now show that the generalized hinge loss $\ell(h_w, (x_i, y_i))$ is a convex function of $w$.

Justify each of the following steps.

(a) The expression $\Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i \rangle$ is an affine function of $w$.

(b) The expression $\max_{y \in \mathcal{Y}} [\Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$ is a convex function of w.

**ANSWER**

(a)

Since both $\Delta(y_i, y)$and $\Psi(x_i, y) - \Psi(x_i, y_i)$ are constant with respect to w, it would be affine.

(b)

Using the results from the previous part,
$\forall y \in \mathcal{Y}, \Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i \rangle$ is affine and convex.

$\therefore \max_{y \in \mathcal{Y}} [\Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$ is convex.

(5) Conclude that $\ell(h_w, (x_i, y_i))$ is a convex surrogate for $\Delta(y_i, f_w(x_i))$.

**ANSWER**

Since,
$\ell(h_w, (x_i, y_i)) \geq \Delta(y, f(x))$

We proved in the last part that $\ell(h_w, (x_i, y_i))$ is convex,

$\therefore \ell(h_w, (x_i, y_i))$ is the convex surrogate for $\Delta(y, f(x))$

# 3 SGD for Multiclass SVM

## 3.1 Question 1

For a training set $(x_1, y_1), \cdots, (x_n, y_n)$, let $J(w)$ be the $\ell_2$-regularized empirical risk function for the multiclass hinge loss. We can write this as

$$J(w) = \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} \max_{y \in \mathcal{Y}} [\Delta(y_i, y)) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle].$$

We will now show that that $J(w)$ is a convex function of $w$. Justify each of the following steps. As we've shown it in a previous problem, you may use the fact that $w \mapsto \max_{y \in \mathcal{Y}} [\Delta(y_i, y)) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$ is a convex function.

(a) $\frac{1}{n} \sum_{i=1}^{n} \max_{y \in \mathcal{Y}} [\Delta(y_i, y)) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$ is a convex function of $w$.

(b) $\|w\|^2$ is a convex function of $w$.

(c) $J(w)$ is a convex function of $w$.

**ANSWER**

(a) Let $f(w) = \max_{y \in \mathcal{Y}} [\Delta(y_i, y)) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$

We know that $f(w)$ is convex, so all the functions in the summation are convex. And the sum of convex functions is also convex.

Therefore, $\frac{1}{n} \sum_{i=1}^{n} \max_{y \in \mathcal{Y}} [\Delta(y_i, y)) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$ is a convex function of $w$.

(b)

$\|w\|^2 = w^T w$
$\nabla\|w\|^2 = 2$ and therefore $\|w\|^2$ is a convex function of w.

(c)

Using the last 2 parts,

Both $\lambda_k\|w_k\|^2$ and $\frac{1}{n}\sum_{i=1}^{n} \max_{y\in\mathcal{Y}}[\Delta(y_i, y)) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i)\rangle]$ are convex,

Therefore, $J(w)$ is a convex function of w.

### 3.2 Question 2

Since $J(w)$ is convex, it has a subgradient at every point. Give an expression for a subgradient of $J(w)$. You may use any standard results about subgradients, including the result from an earlier homework about subgradients of the pointwise maxima of functions. (Hint: It may be helpful to refer to $\hat{y} = arg\max_{y \in \mathcal{Y}} [\Delta(y_i, y) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$.)

**ANSWER**

$J(w) = \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} \max_{y \in \mathcal{Y}} [\Delta(y_i, y)) + \langle w, \Psi(x_i, y) - \Psi(x_i, y_i) \rangle]$

$\partial J(w) = 2\lambda w + \partial [\frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, \hat{y}) + \langle w, \Psi(x_i, \hat{y}) \rangle - \langle w, \Psi(x_i, y_i) \rangle)]$

Using the linearity property of inner products,
$= 2\lambda w + \frac{1}{n} \sum_{i=1}^{n} \partial \langle w, (\Psi(x_i, \hat{y}) - \Psi(x_i, y_i)) \rangle$

$\partial J(w) = 2\lambda w + \frac{1}{n} \sum_{i=1}^{n} (\Psi(x_i, \hat{y}) - \Psi(x_i, y_i))$

### 3.3 Question 3

Give an expression the stochastic subgradient based on the point $(x_i, y_i)$.

**ANSWER**

At point $(x_i, y_i)$ the subgradient can be written as,
g = $2\lambda w + \frac{1}{n}\sum_{i=1}^{n}(\Psi(x_i, \hat{y}) - \Psi(x_i, y_i))$

For updating w, it can be expressed as
$w_t = w_{t-1} - \eta_t \nabla_w J(w_{t-1})$

$= w_{t-1} - \eta_t g_{t-1}$

### 3.4 Question 4

Give an expression for a minibatch subgradient, based on the points $(x_i, y_i), \cdots, (x_{i+m-1}, y_{i+m-1})$

**ANSWER**

The minibatch subgradient can be written as,
$g = 2\lambda w + \frac{1}{m}\sum_{j=i}^{i+m-1}(\Psi(x_j, \hat{y}) - \Psi(x_j, y_j))$

For the update on w, it can be expressed as,

$w_{t-1} - \eta_t(2\lambda w_{t-1} + \frac{1}{n}\sum_{j=1}^{m}\sum_{i=1}^{n}(\Psi(x_{i+j-1}, \hat{y}) - \Psi(x_{i+j-1}, y_{i+j-1})))$

# 4 Another Formulation of Generalized Hinge Loss

## 4.1 Question 1

Show that $\ell(h, (x_i, y_i)) = \max_{y' \in \mathcal{Y}} [\Delta(y_i, y')) - m_{i,y'}(h)]$.

**ANSWER**

The generalized hinge loss is

$$\ell(h, (x_i, y_i)) = \max_{y' \in \mathcal{Y}} [\Delta(y_i, y') + h(x_i, y') - h(x_i, y_i)]$$

$$= \max_{y' \in \mathcal{Y}} [\Delta(y_i, y')) - m_{i,y'}(h)]$$

### 4.2 Question 2

Suppose $\Delta(y, y_0) \geq 0$ for all $y, y_0 \in \mathcal{Y}$. Show that for any example $(x_i, y_i)$ and any score function $h$, the multiclass hinge loss we gave in lecture and the generalized hinge loss presened above are equivalent, in the sense that

$$\max_{y \in \mathcal{Y}} [(\Delta(y_i, y) - m_i, y(h)))_+] = \max_{y \in \mathcal{Y}} (\Delta(y_i, y) - m_{i,y'}(h))).$$

(Hint: This is easy by piecing together other results we have already attained regarding the relationship between $\ell$ and $\Delta$.)

**ANSWER**

$\Delta(y_i, y) - m_{i,y'} \geq \Delta(y_i, y) \geq 0$

Since the term in the maximum is non-negative as shown above,

$\max_{y \in \mathcal{Y}} (\Delta(y_i, y) - m_{i,y'}(h))) = \max_{y \in \mathcal{Y}} [(\Delta(y_i, y) - m_i, y(h)))_+]$

### 4.3 Question 3

In the context of the generalized hinge loss, $\Delta(y, y_0)$ is like the "target margin" between the score for true class $y$ and the score for class $y_0$. Suppose that our prediction function $f$ gets the correct class on $x_i$. That is, $f(x_i) = arg\max_{y_0 \in \mathcal{Y}} h(x_i, y_0) = y_i$. Furthermore, assume that all of our target margins are reached or exceeded. That is

$$m_{i,y}(h) = h(x_i, y_i) - h(x_i, y) \geq \Delta(y_i, y),$$

for all $y \neq y_i$. Show that $\ell(h, (x_i, y_i)) = 0$ if we assume that $\Delta(y, y) = 0$ for all $y \in \mathcal{Y}$.

**ANSWER**

Using the results from the previous problem

$\ell(h, (x_i, y_i)) = \max_{y \in \mathcal{Y}} (\Delta(y_i, y) - m_{i,y'}(h)))$

Since $m_{i,y}(h) = h(x_i, y_i) - h(x_i, y) \geq \Delta(y_i, y)$,
the above expression for loss would be maximum when $y = y_i$, in all the other cases it would be negative.

$\therefore \ell(h, (x_i, y_i)) = \Delta(y_i, y_i)$

$= 0$

## 5 Hinge Loss is a Special Case of Generalized Hinge Loss

Let $Y = \{-1, 1\}$. Let $\Delta(y, \hat{y}) = 1(y \neq \hat{y})$. If $g(x)$ is the score function in our binary classification setting, then define our compatibility function as

$$h(x, 1) = g(x)/2$$

$$h(x, -1) = -g(x)/2.$$

Show that for this choice of $h$, the multiclass hinge loss reduces to hinge loss: $\ell(h, (x, y)) = \max_{y_0 \in \mathcal{Y}}[\Delta(y, y_0)) + h(x, y_0) - h(x, y)] = max\{0, 1 - yg(x)\}$

**ANSWER**

If $y = y_0$,

$\ell(h, (x, y)) = \Delta(y_0, y_0) + h(x, y_0) - h(x, y_0)$

$= 0$

If $y \neq y_0$,

$\ell(h, (x, y)) = \Delta(y, y_0)) + h(x, y_0) - h(x, y)$

=$1(y \neq y_0) + 1/2(-g(x) - g(x)$ (case y = 1) or $1/2(g(x) + g(x))$ (case y=-1)

= $1 + (-g(x))$(case y = 1) or $g(x)(casey = -1)$

$= 1 - yg(x)$

$\therefore, \ell(h(x, y)) = \max\{0, 1 - yg(x)\}$

# 6 Multiclass Classification - Implementation

In this problem we will work on a simple three-class classification example, similar to the one given in lecture. The data is generated and plotted for you in the skeleton code.

## 6.1 One-vs-All (also known as One-vs-Rest)

In this problem we will implement one-vs-all multiclass classification. Our approach will assume we have a binary base classifier that returns a score, and we will predict the class that has the highest score.

1. Complete the class OneVsAllClassifier in the skeleton code. Following the OneVsAllClassifier code is a cell that extracts the results of the fit and plots the decision region. Include these results in your submission.

```
class OneVsAllClassifier(BaseEstimator, ClassifierMixin):

    def __init__(self, estimator, n_classes):

        self.n_classes = n_classes
        self.estimators = [clone(estimator) for _ in range(n_classes)]
        self.fitted = False

    def fit(self, X, y=None):

        for i in range(self.n_classes):

            tempy = np.copy(y)

            for j in range(len(tempy)):

                if tempy[j] == i:
                    tempy[j] = 1

                else:
                    tempy[j] = -1

                self.estimators[i].fit(X, tempy)

        self.fitted = True
        return self
```

```
    def decision_function(self, X):

        if not self.fitted:
            raise RuntimeError("You must train classifer before predicting data.")

        if not hasattr(self.estimators[0], "decision_function"):
            raise AttributeError(
                "Base estimator doesn't have a decision_function attribute.")


        s = (X.shape[0], self.n_classes)
        dec = np.zeros(s)

        for i in range(self.n_classes):

            dec[:,i] = self.estimators[i].decision_function(X)

        return(dec)


    def predict(self, X):

        prediction = np.zeros(X.shape[0])
        dec = self.decision_function(X)

        for i in range(X.shape[0]):

            prediction[i] = np.argmax(dec[i])

        return prediction
```

Coeffs 0 [[-1.21554249 -0.83295331]] Coeffs 1 [[ 0.42090396 -0.30060302]] Coeffs 2 [[ 0.89162796 -0.82467394]]

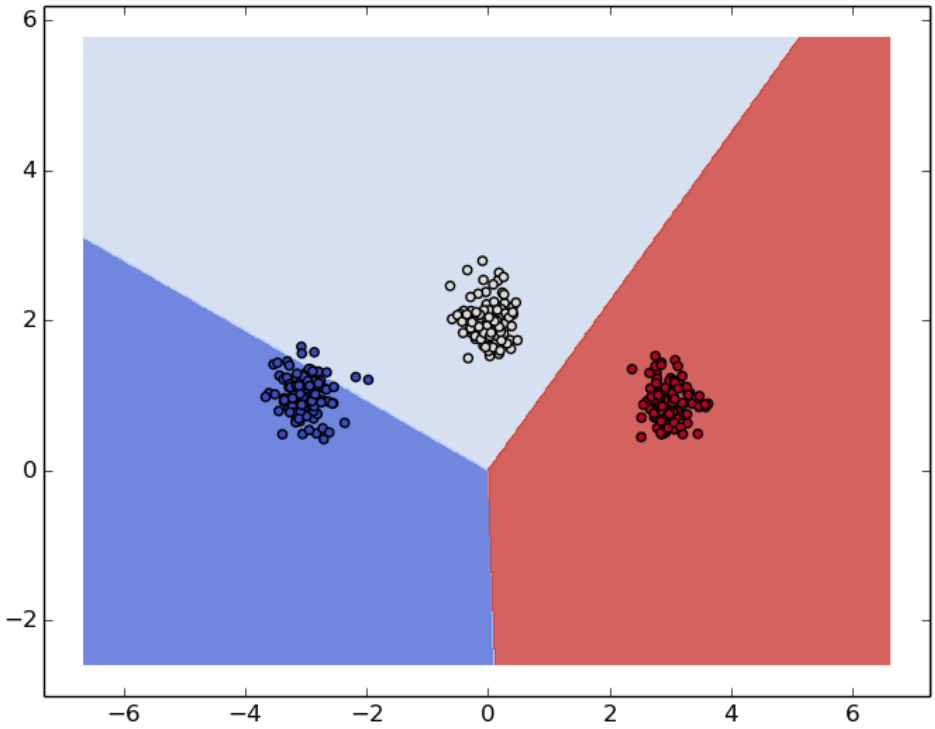


Figure 1: Classification produced after running one vs all

2. [Optional] Normalize the vectors corresponding to each of the linear SVM classifiers so that they have unit norm. Evaluate the results and plot the decision regions for these normalized vectors.

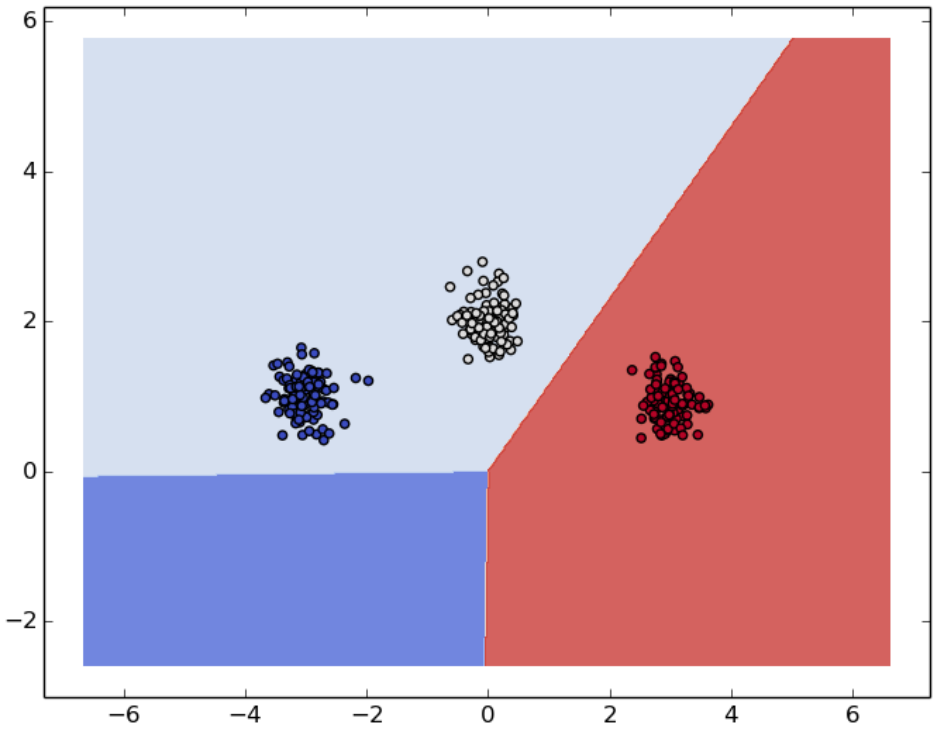


Figure 2: Classification produced after normalizing

3. [Optional] You may notice that every time you run the cell that fits the models and plots the decision regions, you get slightly different results. This is more pronounced when $C$ is larger, e.g. $C = 200$. Investigate and propose an explanation for this. You may use any means necessary, including google searching and asking other people (just like in real life). [It's generally good to investigate things that look odd – you'll almost always learn something, and sometimes you'll uncover a more serious underlying problem.]

## 6.2 Multiclass SVM

In this question, we will implement stochastic subgradient descent for the linear multiclass SVM described in lecture and in this problem set. We will use the class-sensitive feature mapping approach with the "multivector construction", as described in our multiclass classification lecture and in SSBD Section 17.2.1.

1. Complete the skeleton code for multiclass SVM. Following the multiclass SVM implementation, we have included another block of test code. Make sure to include the results from these tests in your assignment, along with your code.

```
def zeroOne(y,a) :

    return int(y != a)

def featureMap(X,y,num_classes) :

    dim = (num_samples, num_outFeatures)
    phi = np.zeros(dim)

    if num_samples == 1:

        new_row = np.zeros(num_outFeatures)
        new_row[y*num_inFeatures:(y+1)*num_inFeatures] = X
        return new_row

    for i in range(X.shape[0]):

        new_row = np.zeros(num_outFeatures)
        new_row[y[i]*num_inFeatures:(y[i]+1)*num_inFeatures] = X[i]
        phi[i,:] = new_row

    return phi



def sgd(X, y, num_outFeatures, subgd, eta = 0.01, T = 10000):

    num_samples = X.shape[0]

    w = np.zeros(num_outFeatures)
    meanw = np.zeros(num_outFeatures)

    ind_arr = list(range(num_samples))

    for i in range(T):
```

```
            # np.random.shuffle(ind_arr)
            j = np.random.randint(num_samples)
            # for j in ind_arr:

            gradient = subgd(X[j],y[j],w)
            w = w - eta * gradient
            meanw += w

        return meanw/T

    class MulticlassSVM(BaseEstimator, ClassifierMixin):

        def __init__(self, num_outFeatures, lam=1.0, num_classes=3, Delta=zeroOne, Psi=

            self.num_outFeatures = num_outFeatures
            self.lam = lam
            self.num_classes = num_classes
            self.Delta = Delta
            self.Psi = lambda X,y : Psi(X,y,num_classes)
            self.fitted = False

        def subgradient(self,x,y,w):

            y_cap = np.zeros(self.num_classes)

            for p in range(self.num_classes):

                y_cap[p] = self.Delta(y, p) + np.dot(w, (self.Psi(x,p) - self.Psi(x,y))

            res = 2 * self.lam * w + self.Psi(x, np.argmax(y_cap)) - self.Psi(x,y)

            return res


        def fit(self,X,y,eta=0.1,T=10000):

            self.coef_ = sgd(X,y,self.num_outFeatures,self.subgradient,eta,T)
            self.fitted = True
            return self

        def decision_function(self, X):

            if not self.fitted:
                raise RuntimeError("You must train classifer before predicting data.")
```

```
        s = (X.shape[0], self.num_classes)
        dec = np.zeros(s)

        for i in range(X.shape[0]):

            new_row = np.zeros(self.num_classes)

            for j in range(self.num_classes):

                new_row[j] = np.dot(self.coef_, self.Psi(X[i],j))

            dec[i] = new_row

        return(dec)

    def predict(self, X):

        dec = self.decision_function(X)
        prediction = np.zeros(X.shape[0])

        for i in range(X.shape[0]):

            prediction[i] = np.argmax(dec[i])

        return prediction
```

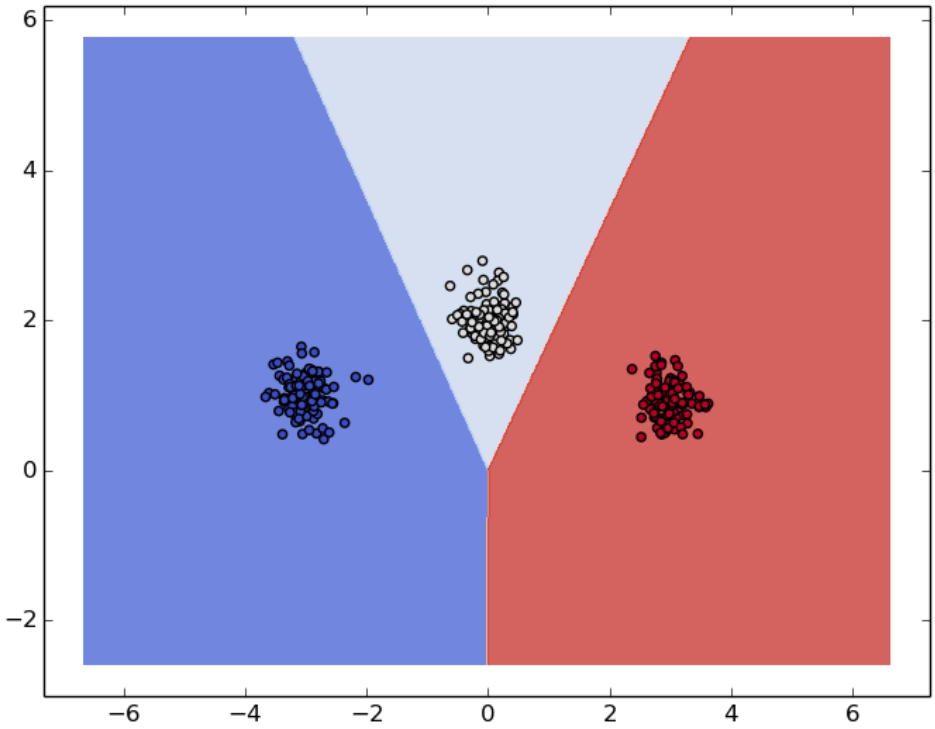


Figure 3: Classification produced using Multiclass SVM

# 7 [Optional] Audio Classification

In this problem, we will work on the urban sound dataset URBANSOUND8K from the Center for Urban Science and Progress (CUSP) at NYU. We will first extract features from raw audio data using the LibROSA package, and then we will train multiclass classifiers to classify the sounds into 10 sound classes. URBANSOUND8K dataset contains 8732 labeled sound excerpts. For this problem, you may use the file UrbanSound8K.csv to randomly sample 2000 examples for training and 2000 examples for validation.

1. In LibROSA, there are many functions for visualizing audio waves and spectra, such as display.waveplot() and display.specshow(). Load a random audio file from each class as a floating point time series with librosa.load(), and plot their waves and linear-frequency power spectrogram. If you are interested, you can also play the audio in the notebook with functions display() and Audio() in IPython.display.

2. Mel-frequency cepstral coefficients (MFCC) are a commonly used feature for sound processing. We will use MFCC and its first and second differences (like discrete derivatives) as our features for classfication. First, use function feature.mfcc() from LibROSA to extract MFCC features from each audio sample. (The first MFCC coefficient is typically discarded in sound analysis, but you do not need to. You can test whether this helps in the optional problem below.) Next, use function feature.delta() to calculate the first and second differences of MFCC. Finally, combine these features and normalize each feature to zero mean and unit variance.

3. Train a linear multiclass SVM on your 2000 example training set. Evaluate your results on the validation set in terms of 0/1 error and generate a confusion table. Compare the results to a one-vs-all classifier using a binary linear SVM as the base classifier. For each model, may use your code from the previous problem, or you may use another implementation (e.g. from sklearn).

4. [More Optional] Compare results to any other multiclass classification methods of your choice.

5. [More Optional] Try different feature sets and see how they affect performance.