CS231N Assignment 1 - SVM

1 Soft SVM - Introduction

In contrast with the hard SVM that we learned in CS3244.

1.1 What's wrong with the Hard SVM?

In hard SVM, the optimization problem enforces that all points in training must be classified correctly. However in practical cases, noise prevents/limits the existence of a hyperplane that **perfectly** seperates the data, in which case the hard SVM optimization returns no/poor solutions.

Alternatively, a single outlier determines the boundary in the hard SVM, making it overly sensitive to noise in data

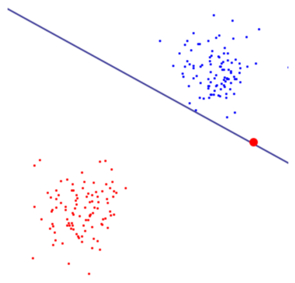


Figure 1: Single red outlier determines boundary - hallmark of overfitting

1.2 Developing a Soft SVM

Therefore, we modify the hard SVM to allow for some training points to be missclassified.

To do this, we introduce *slack variables* $\xi_n \geq 0$, one slack variable for each training data point.

1.2.1 Modifiying the classification constraint

- Classification constraints are now $t_n y_n(x_n) \geq 1 - \xi_n$, in which slack variables constrained to satisfy $\xi_n \geq 0$
  - Recall that $y_n(x_n) = w^T x_n + b$
- Data points for which $\xi_n = 0$ are correctly classified (either on the margin or on the correct side of the margin)
- Points for which $0 < \xi_n \leq 1$ lie inside the margin, but on the correct side of the decision boundary
- Points for which $\xi_n > 1$ lie on the wrong side of the decision boundary and are **misclassified**
- One choice of ξ_n would be the *hinge loss*
  - Hinge loss $= max\{0, 1 - y_n(w^T x_n + b)\}$

1.2.2 Our new optimization goal now

We want to maximize the margin while **penalizing points that lie on wrong side of the decision boundary**.

Therefore, **minimize** $\left\{C \sum_{n=1}^{N} \xi_n + \frac{||w||^2}{2}\right\}$ subject to constraints $t_n y_n(x_n) \geq 1 - \xi_n$; $\xi_n \geq 0$

- **Regularization** parameter C controls the trade-off between maximizing the margin and minimizing the loss

Similar to the hard SVM case, we can solve this by forming converting from primal lagrangian to dual lagragian (See PRML). We will observe that the dual lagrangian of the soft SVM only differs by an upper bound applied to the Lagrange multipliers.

2 Multiclass SVM Loss - CS231N

CS231N terms this 'SVM' loss, although this is not really accurate (we are just looking at the hinge loss term). We will use these two terms interchangeably.

2.1 Recap on loss functions

- Suppose we have dataset of examples $\left\{(x_i, y_i)\right\}_{i=1}^{N}$; x_i is image, y_i is label
- Suppose we have some linear classifier $f(x_i, W)$; W is some weight matrix
- Loss for a **single** example defined as $L_i = (f(x_i, W), y_i)$
- Loss for **entire dataset** defined as average of per-example loss; $L = \frac{1}{N}\sum_i L_i(f(x_i, W), y_i)$

2.2 Visualizing linear classifiers

2.2.1 Algebraic viewpoint

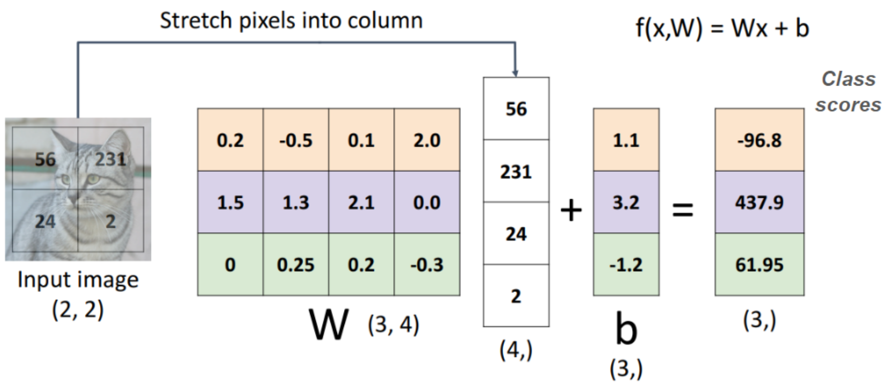


Figure 2: Weight matrix has it's own preference for each class, weighting the input image appropriately

2.2.2 Template viewpoint

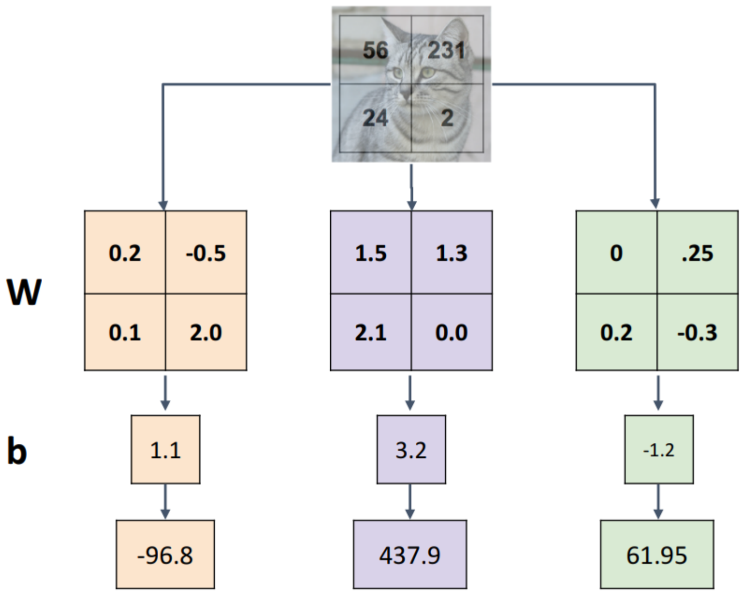


Figure 3: Transform weight matrix into same dimension as input image, then do dot product
Recall that dot product is a measure of similarity

2.2.3 Visual viewpoint

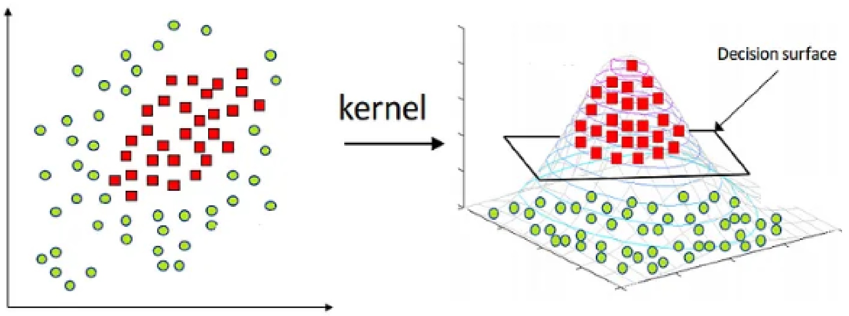


Figure 4: Hyperplane seperating points (after kernel transformation)

2.3 Multiclass SVM Loss

- Suppose for some i_{th} training example, we are given the pixels of image x_i and the label y_i that specifies the index of the correct class
- We have a linear classifier $f(x_i, W)$ that outputs a **vector v** of *class scores* (closeness of x_i to all labels y)
  - eg. j_{th} class score $= v_j = f(x_i, W)_j$
- Define the Multiclass SVM loss for the i_{th} example as $L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta)$; y_i is the 'true' class
  - Since we are working with linear classifiers (ie. $f(x_i, W) = Wx_i$), we can rewrite the loss function as $L_i = \sum_{j \neq y_i} max(0, W_j^T x_i - W_{y_i}^T x_i + \Delta)$
    - W_j is j_{th} row of weight matrix W
    - W_{y_i} is the row of weight matrix W corresponding to the **correct** class label y_i
    - Assuming that vectors x_i are **row** vectors, we transpose the weight matrices into **column** vectors to make mmult dimensions work
  - Given some image x_i, it's correct class score s_{y_i} must be larger than **all** incorrect class scores by at least Δ. Otherwise loss will be incurred.

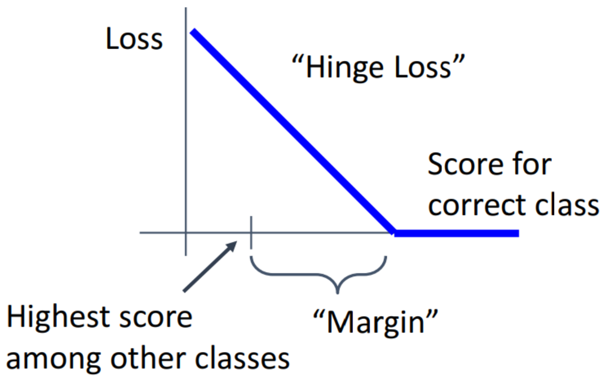
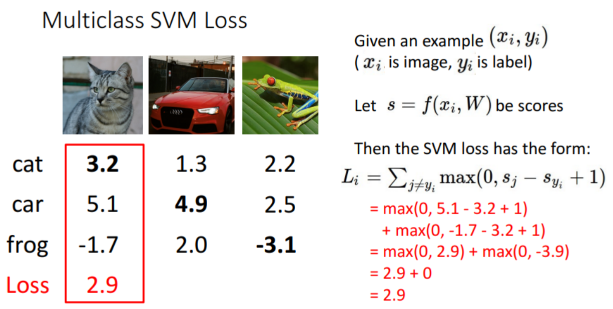


Figure 5: Hinge Loss



Multiclass SVM Loss

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9		

Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

= max(0, 5.1 - 3.2 + 1)
+ max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Figure 6: Example calculation

2.4 Gradient of Hinge Loss - Single datapoint

We know that the multiclass SVM loss is $L_i = \sum_{j \neq y_i} max(0, W_j^T x_i - W_{y_i}^T x_i + \Delta)$ (1)

We can split this up as a piecewise function $L_i = \begin{cases} \sum_{j \neq y_i} (W_j^T x_i - W_{y_i}^T x_i + \Delta) & \text{if } W_{y_i}^T x_i < W_j^T x_i + \Delta \\ 0 & \text{otherwise} \end{cases}$ (2.1)

Intuitively, this is saying that we incur non-zero error whenever the correct class score $W_{y_i}^T x_i = s_{y_i}$ is not larger than all other incorrect class scores by Δ (2.2)

We are doing a scalar-by-matrix derivative, arranged in denominator layout

$$\nabla_{\boldsymbol{W}} L_i = \frac{\partial L_i}{\partial \boldsymbol{W}} = \begin{bmatrix} \frac{\partial L_i}{W_{11}} & \frac{\partial L_i}{W_{12}} & \cdots & \frac{\partial L_i}{W_{1p}} \\ \frac{\partial L_i}{W_{21}} & \frac{\partial L_i}{W_{22}} & \cdots & \frac{\partial L_i}{W_{2p}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial L_i}{W_{j1}} & \frac{\partial L_i}{W_{j2}} & \cdots & \frac{\partial L_i}{W_{jp}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial L_i}{W_{n1}} & \frac{\partial L_i}{W_{n2}} & \cdots & \frac{\partial L_i}{W_{np}} \end{bmatrix} \tag{3.1}$$

Let us proceed to derive the hinge loss' gradient **for the case where error is incurred**

$$\nabla_{w_{y_i}} L_i = -\sum_{j \neq y_i} x_i \tag{3.2}$$

This is saying that we scale x_i by the number of classes that didn't meet the margin Δ (3.2.1)
Note that this is the gradient only with respect to the row of $\boldsymbol{W}$ that corresponds to the correct class
Recall that W_{y_i} corresponds to row of weight matrix $\boldsymbol{W}$
ie. We are doing a scalar-by-vector derivative that provides us with a row of $\frac{\partial L_i}{\boldsymbol{W}}$, as seen below (3.2.2)

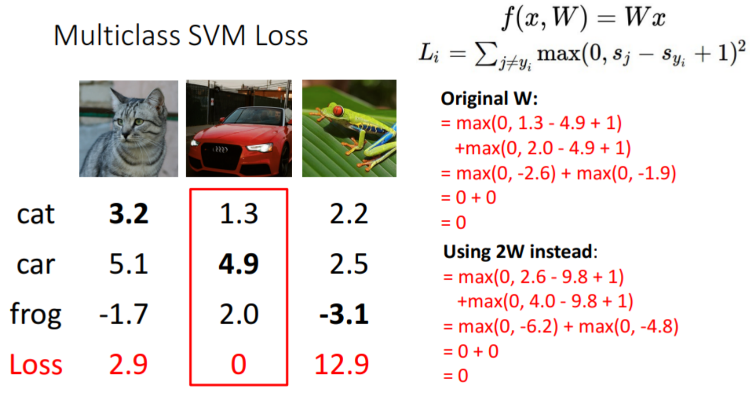
$$\frac{\partial L_i}{\partial \boldsymbol{W}_{y_i}} = \begin{bmatrix} \frac{\partial L_i}{W_{y_i 1}} & \frac{\partial L_i}{W_{y_i 2}} & \cdots & \frac{\partial L_i}{W_{y_i p}} \end{bmatrix}$$

Hence for the other rows of $\nabla_{\boldsymbol{W}} L_i$, we have that $\nabla_{w_j} L_i = x_i$ (3.3)

2.5 Regularization

2.5.1 Intuition

Set of $\boldsymbol{W}$ that correctly classifies (ie. $L_i = 0$ for all i) is **not unique**.
Intuitively speaking this is true. If some $\boldsymbol{W}$ satisfies that $L_i = 0\ \forall i$, then $\lambda \boldsymbol{W}; \lambda > 1$ also satifies that.


Multiclass SVM Loss

$$f(x, W) = Wx$$
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Loss	2.9	0	12.9

Original W:
= max(0, 1.3 - 4.9 + 1)
+max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Using 2W instead:
= max(0, 2.6 - 9.8 + 1)
+max(0, 4.0 - 9.8 + 1)
= max(0, -6.2) + max(0, -4.8)
= 0 + 0
= 0

Figure 7: Non-uniqueness of weights

2.6 Regularization term

We want to encode some preference for a certain set of weights $\boldsymbol{W}$ over others to remove this ambiguity.
We can do so by extending the loss function with a regularization penalty $R(W)$.
Common $R(W)$ is the *Squared L2 norm*, which discourages large weights through a quadratic penalty.

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

Figure 8: Notice that $R(W)$ is function of weights only, not data
Intuitively speaking, smaller weights are preferred since they generalize better (no input dimension can have very large influence on scores all by itself)

2.7 Gradient of Hinge Loss - All datapoints

Let's attempt to calculate gradient of hinge loss for **all** datapoints in a non-naive way

$$\text{Define } margin_{ij} = W_j^T x_i - W_{y_i}^T x_i + \Delta \tag{1.1}$$

$$\text{Rewrite } L_i = \sum_{j \neq y_i} max(0, W_j^T x_i - W_{y_i}^T x_i + \Delta) = \sum_{j \neq y_i} max(0, margin_{ij}) \tag{1.2}$$

Let's rederive the partial derivatives, but not breaking it up into piecewise function as we did previously (1.3)

$$\begin{aligned} \nabla W_{y_i} L_i &= \nabla W_{y_i} \Big\{ \sum_{j \neq y_i} max(0, W_j^T x_i - W_{y_i}^T x_i + \Delta) \Big\} \\ &= -\sum_{j \neq y_i} \mathbb{1}_{>0}(W_j^T x_i - W_{y_i}^T x_i + \Delta) x_i \\ &= -\sum_{j \neq y_i} \mathbb{1}_{>0}(margin_{ij}) x_i \end{aligned} \tag{1.3.1}$$

$$\begin{aligned} \nabla W_j L_i &= \nabla W_j \Big\{ \sum_{j \neq y_i} max(0, W_j^T x_i - W_{y_i}^T x_i + \Delta) \Big\} \\ &= \mathbb{1}_{>0}(W_j^T x_i - W_{y_i}^T x_i + \Delta) x_i \\ &= \mathbb{1}_{>0}(margin_{ij}) x_i \end{aligned} \tag{1.3.2}$$

Let's derive equations of the partial derivatives, for all datapoints now (1.4)

$$\begin{aligned} \nabla W_y L &= \nabla W_y \Big\{ \frac{1}{N} \sum_i^N L_i \Big\}; \; \boldsymbol{W}_y \text{ is rows of } \boldsymbol{W}\text{, y represents all correct class labels} \\ &= \frac{1}{N} \sum_i^N \Big\{ \nabla W_{y_i} L_i \Big\} \\ &= \frac{1}{N} \sum_i^N \Big\{ \sum_{j \neq y_i} -\mathbb{1}_{>0}(margin_{ij}) x_i \Big\} \end{aligned} \tag{1.4.1}$$

$$\begin{aligned} \nabla W_j L &= \nabla W_j \Big\{ \frac{1}{N} \sum_i^N L_i \Big\}; \text{ j is our loop iterator} \\ &= \frac{1}{N} \sum_i^N \Big\{ \nabla W_j L_i \Big\} \\ &= \frac{1}{N} \sum_i^N \Big\{ \mathbb{1}_{>0}(margin_{ij}) x_i \Big\} \end{aligned} \tag{1.4.2}$$

2.7.1 Vectorized Dot Product approach

Let's build some intuition using a toy example first. Note that matrix values are not accurate, they are just for illustration

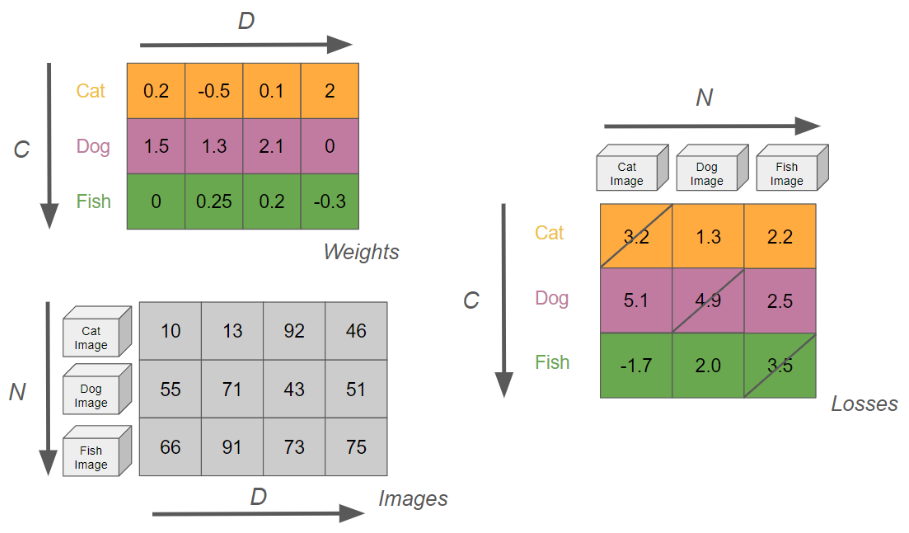


Figure 9: C (number of classes) = 3
D (data dimension) = 4
N (number of samples) = 3

- Let us consider how $\frac{\partial L_i}{\partial \boldsymbol{W}}$ is calculated, taking cat image as an example
  - $$\nabla_{W_{y_i}} L_i = -\sum_{j\neq y_i} \mathbb{1}_{>0}(margin_{ij})x_i$$
  - $$\nabla_{w_j} L_i = \mathbb{1}_{>0}(margin_{ij})x_i$$

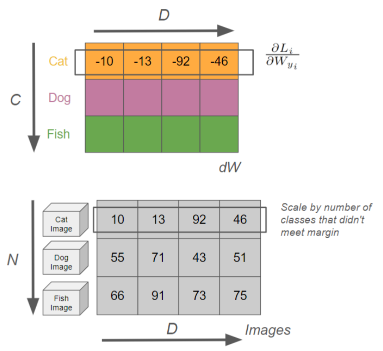


Figure 10: Dog class margin did not meet Δ constraint

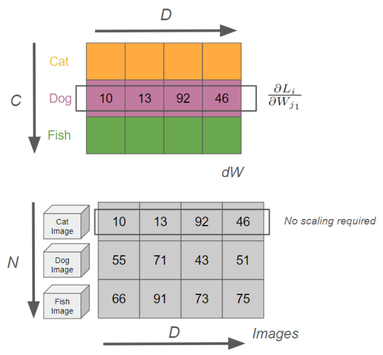


Figure 11: No scaling required (as per derivation calculation)

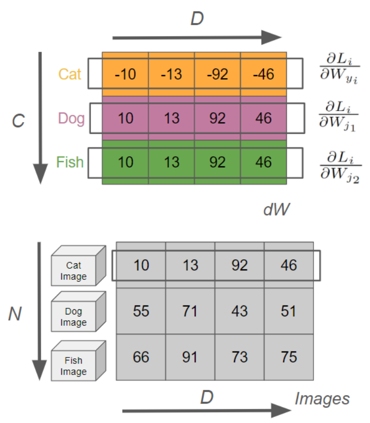


Figure 12: Final dW matrix for L_i corresponding to cat image

- Note the key discovery here . . .

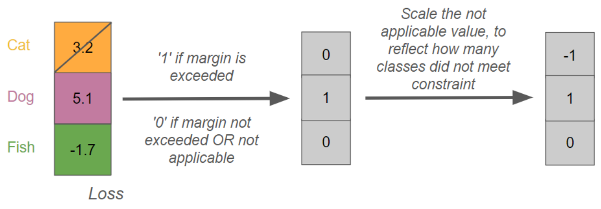

Cat
3.2
Dog
5.1
Fish
-1.7
Loss
'1' if margin is exceeded
'0' if margin not exceeded OR not applicable
0
1
0
Scale the not applicable value, to reflect how many classes did not meet constraint
-1
1
0

Figure 13: Convert *Loss* to a 'flag' matrix

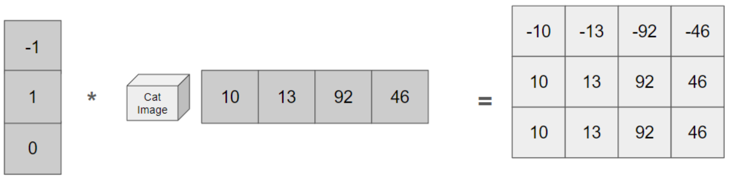


Figure 14: Strang's Row-Matrix multiplication idea