

- Recall

$$f(x; W) = Wx + b$$

weight  
 image vector

- TODO

- define a loss function
- algorithm to do optimization

- dataset

$$\{f(x_i, y_i)\}_{i=1}^N$$

$x_i$  is image  
 $y_i$  is label

- cost function

$$L = \frac{1}{N} \sum_i L_i(f(x_i; W), y_i)$$

- multi-class SVM

Suppose: 3 training examples, 3 classes.  
With some  $W$  the scores  $f(x, W) = Wx$  are:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

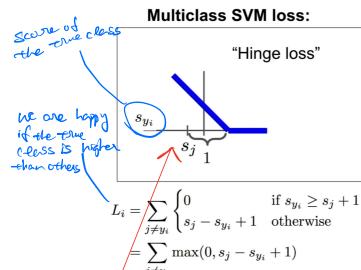
Multiclass SVM loss:

Given an example  $(x_i, y_i)$  where  $x_i$  is the image and where  $y_i$  is the (integer) label,  
and using the shorthand for the scores vector:  $s = f(x_i, W)$

the SVM loss has the form:  

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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



Suppose: 3 training examples, 3 classes.  
With some  $W$  the scores  $f(x, W) = Wx$  are:



cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>
Losses:	<b>2.9</b>		

Multiclass SVM loss:

Given an example  $(x_i, y_i)$  where  $x_i$  is the image and where  $y_i$  is the (integer) label,

and using the shorthand for the scores vector:  $s = f(x_i, W)$

the SVM loss has the form:

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

$$= \max(0, 5.1 - 3.2 + 1) + \max(0, -1.7 - 3.2 + 1)$$

$$= \max(0, 2.9) + \max(0, -3.9)$$

$$= 2.9 + 0$$

$$= 2.9$$

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

$$= \max(0, 1.3 - 4.9 + 1) + \max(0, 2.0 - 4.9 + 1)$$

$$= 0$$

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

$$= \max(0, 2.2 - 2.1 + 1)$$

$$= \max(0, 0.1 + 1)$$

$$= 0.1$$

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

$$= \max(0, 2.5 - 2.1 + 1)$$

$$= \max(0, 0.4 + 1)$$

$$= 0.4$$

$$\frac{1}{3} (2.9 + 0.1 + 0.4) = 1.29$$

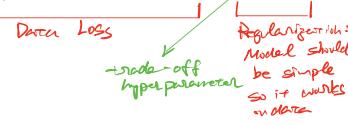
- $W$  not good  
continue to optimize!
- optimal  $W$  unique?
- No,  $2W$  is also optimal

Qs

- \* What's gonna happen if score for car is changed a bit?  
A: loss will not change.  
will still be 0 for car  
and loss still the same.
- \* min/max fix loss?  
A: min = 0  
max =  $\infty$
- \* If we use sum instead of mean?  
A: classifier doesn't change
- \* What if we use  $L_i = \sum_j \max(0, s_j - s_{y_i})^k$ ?  
A: algo different

- Regularization

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(f(x_i; W), y_i) + \lambda R(W)$$



Regularization

$\lambda$  = regularization strength (hyperparameter)

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \lambda R(W)$$

In common use:

L2 regularization  $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization  $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2)  $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Max norm regularization (might see later)

Dropout (will see later)

Fancier: Batch normalization, stochastic depth

## L2 Regularization (Weight Decay)

$$x = [1, 1, 1, 1]$$

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

$$w_1 = [1, 0, 0, 0]$$

$$w_2 = [0.25, 0.25, 0.25, 0.25]$$

(If you are a Bayesian: L2 regularization also corresponds MAP inference using a Gaussian prior on  $W$ )

$$w_1^T x = w_2^T x = 1$$

- softmax classifier (multinomial logistic regression)

## Softmax Classifier (Multinomial Logistic Regression)



scores = unnormalized log probabilities of the classes.

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

softmax

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i | X = x_i)$$

$$L_i = -\log \left( \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right)$$

↓ exp

$$\begin{matrix} 24.5 \\ 164.0 \\ 0.18 \end{matrix} \xrightarrow{\text{normalized}} \begin{matrix} 0.13 \\ 0.87 \\ 0.00 \end{matrix}$$

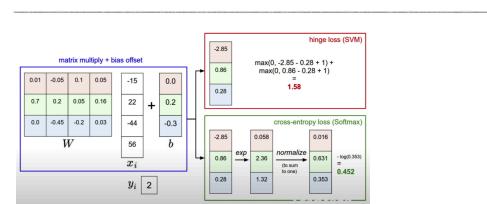
$$L_i = -\log (0.13) = 0.87$$

Q: min/max of "loss"

A:  $\Rightarrow 0 \rightarrow \infty$   
practical, will  $L_i = 0$  be hard to get as we need

$$\frac{e^{s_k}}{\sum_j e^{s_j}} = 1$$

∴ will never get to 0 and near  $s_{y_i} \rightarrow \infty$



## Recap

- We have some dataset of  $(x, y)$
- We have a **score function**:  $s = f(x; W) \stackrel{\text{e.g.}}{=} Wx$
- We have a **loss function**:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \quad \text{Softmax}$$

SVM

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

$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W) \quad \text{Full loss}$$

regularization

- How do we do it? Optimization

- Strategy : Random Search
- Follow the slope : Gradient Descent



```
# Vanilla Gradient Descent

while True:
    weights_grad = evaluate_gradient(loss_fun, data, weights)
    weights += -step_size * weights_grad # perform parameter update
    hyperparameter step
    more-expensive
    of some expression
    (learning rate)
```

### Stochastic Gradient Descent (SGD)

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(x_i, y_i, W) + \lambda R(W) \quad \text{Full sum expensive when N is large!}$$

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^N \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W) \quad \text{Approximate sum using a minibatch of examples}$$

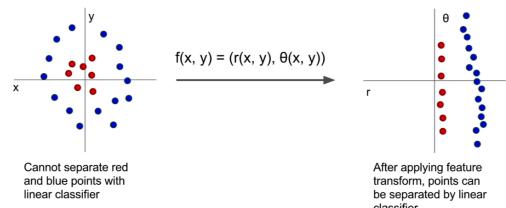
32 / 64 / 128 common

```
# Vanilla Minibatch Gradient Descent
```

```
while True:
    data_batch = sample_training_data(data, 256) # sample 256 examples
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += -step_size * weights_grad # perform parameter update
```

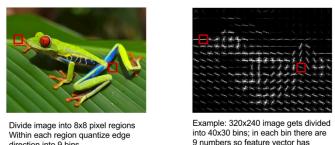
- Image Features (instead of raw data)

### Image Features: Motivation



- image color histogram
- how about MIMO radar?

### Example: Histogram of Oriented Gradients (HoG)



### Example: Bag of Words

