Loss function and optimization

Loss function tells us how good our current classifier is

Given a data set:

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

where,

x_i is an image

y_i is a label (int)

Loss of the dataset is sum of loss over examples,

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

Multi-class 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: $s = f(x_i, W)$

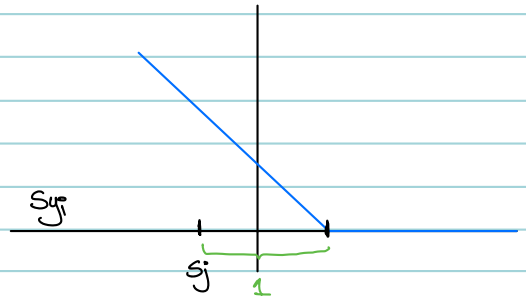
the SVM loss has form:

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

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

SVM cares about only getting the correct score 1 more than the incorrect score.

"Hinge loss"



score incorrect categories (s_j)
score of correct category (s_{y_i})

Qs:

1. What is the min/max possible loss?
   ↳ min is 0 & max is ∞

2. At initialization W is small so all $s \approx 0$. What is the loss?
   ↳ number of classes − 1

   Explanation:

   $s_j \approx 0 \quad s_{y_i} \approx 0$

   $\max(0, s_j - s_{y_i} + 1) = \max(0, 0 - 0 + 1) = 1$ ← Loss for each class

   As this loss sums only incorrect class, so, loss at beginning is number of classes − 1

What happens if the sum was over all classes?
↳ the loss increases by 1

What happens if take mean?
↳ Nothing

What happens if take $\max(0, s_j - s_{y_i} + 1)^2$?
↳ This is a different loss function because loss of each example increases drastically. if L_i is 2 then it would be 4.
For mean, it just scales the overall loss.

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

```
def L_i_vectorized(x, y, W):
  scores = W.dot(x)
  margins = np.maximum(0, scores - scores[y] + 1)
  margins[y] = 0
  loss_i = np.sum(margins)
  return loss_i
```

Gradient:

$$\text{margin} = w_j x - w_i x + \text{delta}$$

For incorrect class:

$$\frac{\partial L}{\partial w_j} = x$$

$w_j = w_j - \alpha x$ ∴ We sub X ∝ decrease it's contribution

For correct class:

$$\frac{\partial L}{\partial w_i} = -x$$

$w_j = w_j - \alpha(-x)$

$w_j = w_j + \alpha x$ ∴ We add X ∝ increase it's contribution

Regularization:

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

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

Softmax Classifier (Multinomial Logistic Regression):

scores = unnormalized log probabilities of the classes.
$s = f(x_i; W)$

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

$$L_i = -\log P(Y = k \mid X = x_i)$$

↳ − log of the true class score

Take the scores run through the softmax. Now, we get the probability scores

Eg: For cat,

cat	3.2		24.5	0.13	
car	5.1	→	164.0	0.87	→ $L_i = -\log(0.13)$
frog	−1.7		0.18	0.00	$= 0.89$

Qs:

1. At initialization W is small so all $s \approx 0$. What is the loss?
↳ $-\log\left(\frac{1}{C}\right)$

Gradient:

grad = a − ∝ * X

if y[j] == y[i]
∝[j] = 1
else:
∝[j] = 0

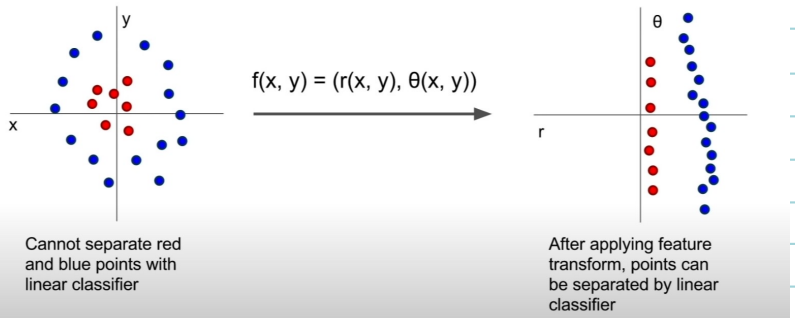
Stochastic Gradient Descent (SGD):

• We sample from dataset to compute an estimate of the full sum & full gradient.

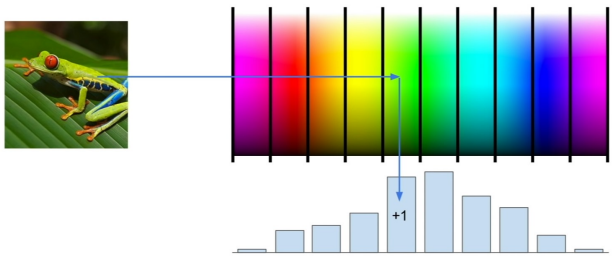
```
while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

(256: power of 2)

Image features:

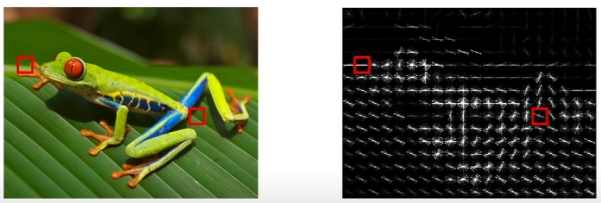


Example: Color Histogram



"What of colors exist in the image?"

Example: Histogram of Oriented Gradients (HoG)

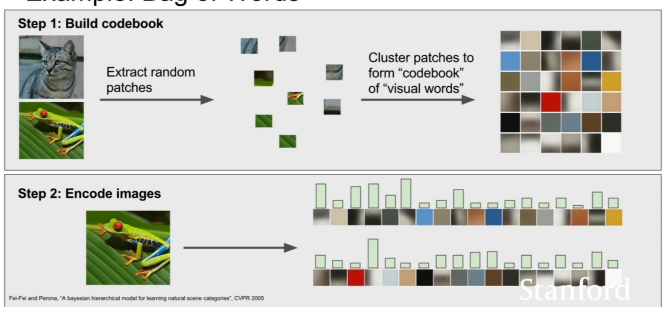


Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins

Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

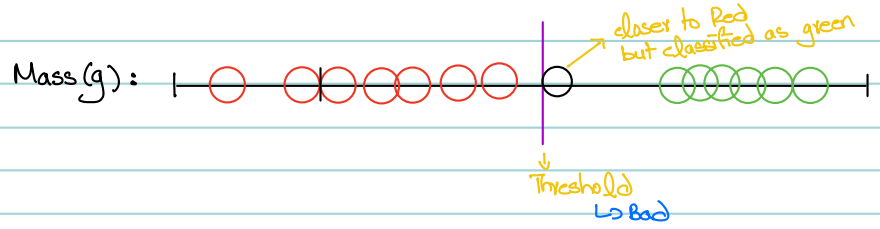
"What type of edges exist in the image?"

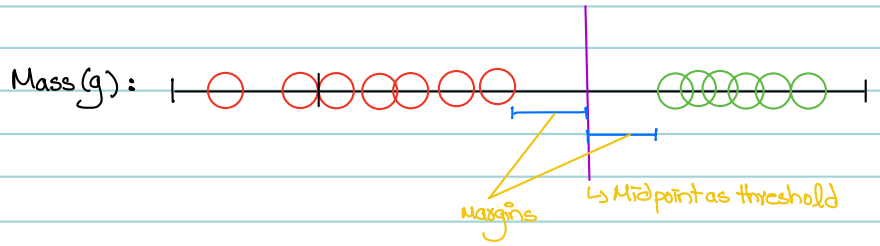
Example: Bag of Words



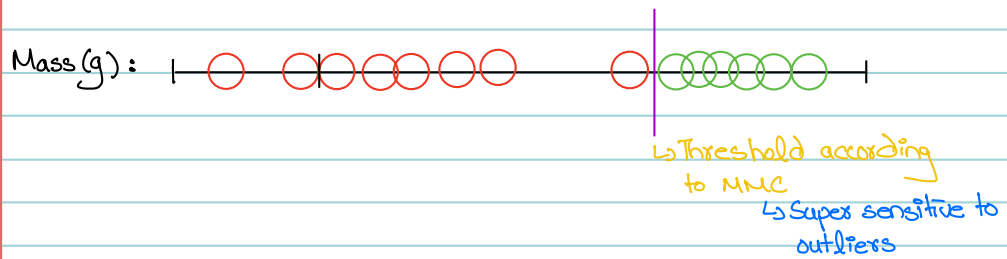
colors in the image
edges in different directions.

Fundamentals of SVM (From statQuest):

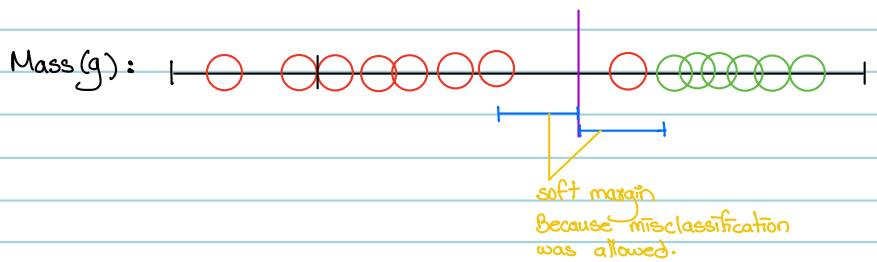




- When we use a threshold that gives us the largest margin to make classification is called maximum margin classifier.



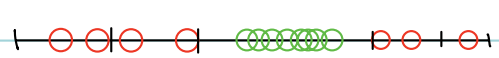
- To make it not sensitive, we must allow mis-classification



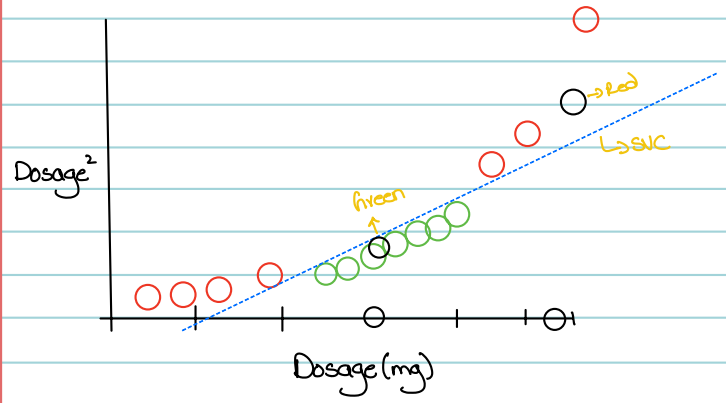
- How do we choose soft margin?
  - ↳ We CV to determine how many misclassifications and observations to allow inside of the soft margin to get the best classification.

- When we use soft margin to determine the location of a threshold, then it is called soft margin classifier aka Support Vector Machines
- Observations on the edge & within the soft margin are called support vectors.

Tons of overlap:

Dosage (mg): 

- SVM don't perform well on this type of data. So, we use SVM



1) Start in lower dimension
2) Move to higher dimension
3) Find a SVC

- Why Dosage² on y-axis?
  - ↳ SVM uses kernel functions to find SVC in higher dimension

- In above example, we use polynomial kernel, it finds relation between each pair of observations & these relationships is used to find SVC at every d.
- Good value of d can be found by CV.

- Another common kernel is Radial Basis Function (RBF) kernel. It finds SVC in infinite dimensions
- It behaves like Weighted Nearest Neighbor model, it checks new observation's neighbor's label & gives it that label.
- The trick, calculating the higher dimensional relationships without actually transforming the data to higher dimension, is called The kernel trick.

Polynomial kernel:

$$(a \times b + r)^d$$

- d → degree of polynomial
- a, b → diff observations
- r → coefficient

For $d=2$, $r=\frac{1}{2}$;

$$\left(a \times b + \frac{1}{2}\right)^2 = a^2b^2 + ab + \frac{1}{4}$$

$$= ab + a^2b^2 + \frac{1}{4}$$

$$= \left(a, a^2, \frac{1}{2}\right) \cdot \left(b, b^2, \frac{1}{2}\right)$$ { high dimensional relationship

Radial Basis function kernel:

$e^{-\gamma(a-b)^2}$

scales sq distance

- More close observations have high influence

$= e^{-\frac{1}{2}(a-b)^2}$

$= e^{-\frac{1}{2}(a^2+b^2-2ab)}$

$= e^{-\frac{1}{2}(a^2+b^2)} e^{ab}$

$e^{ab} = 1 + \frac{1}{1!}ab + \frac{1}{2!}(ab)^2 + \frac{1}{3!}(ab)^3 + \cdots + \frac{1}{\infty!}(ab)^\infty$

A polynomial kernel with $r=0$ & $d=\infty$;

$= a^0b^0 + a^1b^1 + \cdots + a^\infty b^\infty = (1, a, a^2, \cdots, a^\infty) \cdot (1, b, b^1, \cdots, b^\infty)$

$e^{ab} = \left(1, \sqrt{\frac{1}{1!}}a, \sqrt{\frac{1}{2!}}a^2, \cdots, \sqrt{\frac{1}{\infty!}}a^\infty\right) \cdot \left(1, \sqrt{\frac{1}{1!}}b, \sqrt{\frac{1}{2!}}b^2, \cdots, \sqrt{\frac{1}{\infty!}}b^\infty\right)$

$e^{-\frac{1}{2}(a-b)^2} = e^{-\frac{1}{2}(a^2+b^2)} \left[\left(1, \sqrt{\frac{1}{1!}}a, \sqrt{\frac{1}{2!}}a^2, \cdots, \sqrt{\frac{1}{\infty!}}a^\infty\right) \cdot \left(1, \sqrt{\frac{1}{1!}}b, \sqrt{\frac{1}{2!}}b^2, \cdots, \sqrt{\frac{1}{\infty!}}b^\infty\right)\right]$

$= \left(s, s\sqrt{\frac{1}{1!}}a, s\sqrt{\frac{1}{2!}}a^2, \cdots, s\sqrt{\frac{1}{\infty!}}a^\infty\right) \cdot \left(s, s\sqrt{\frac{1}{1!}}b, s\sqrt{\frac{1}{2!}}b^2, \cdots, s\sqrt{\frac{1}{\infty!}}b^\infty\right) \quad \because s = e^{-\frac{1}{2}(a^2+b^2)}$