LOSS FUNCTIONS AND OPTIMIZATION

- ThS. Đoàn Chánh Thống
- ThS. Nguyễn Cường Phát
- ThS. Nguyễn Hữu Lợi
- ThS. Trương Quốc Dũng
- ThS. Nguyễn Thành Hiệp
- ThS. Võ Duy Nguyên
- ThS. Nguyễn Văn Toàn
- ThS. Lê Ngô Thục Vi
- TS. Nguyễn Duy Khánh
- TS. Nguyễn Tấn Trần Minh Khang

Administrative

- Assignment 1 is released:
- Bài tập 1 đã được thông báo:
- http://cs231n.github.io/assignments2017/assignment1/

- Due Thursday April 20, 11:59pm on Canvas
- Hạn chót vào Thứ Năm, ngày 20 tháng 4, lúc 23:59 tối trên Canvas
- (Extending due date since it was released late).
- (Kéo dài hạn nộp bài vì thông báo muộn).

Administrative

- Check out Project Ideas on Piazza.
- Kiểm tra những ý tưởng dự án trên Piazza.

- Schedule for Office hours is on the course website TA specialties are posted on Piazza.
- Lịch trình giờ hành chính của các trợ giảng (TA) có trên trang web của khóa học.

Administrative

- Details about redeeming Google Cloud Credits should go out today; will be posted on Piazza.
- Thông tin chi tiết về việc đổi Tín dụng Google Cloud sẽ được công bố ngay hôm nay; sẽ được đăng trên Piazza.

- $100 per student to use for homeworks and projects.
- $100 mỗi học sinh để sử dụng cho bài tập về nhà và dự án.

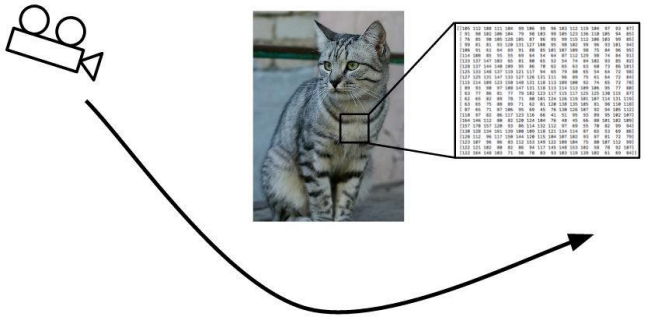
Content

1. Recall from last time
2. Loss function
   + Multiclass SVM loss.
   + Cross Entropy Loss (Softmax)
3. Optimization

RECALL FROM LAST TIME

Challenges

Viewpoint



Illumination

Deformation

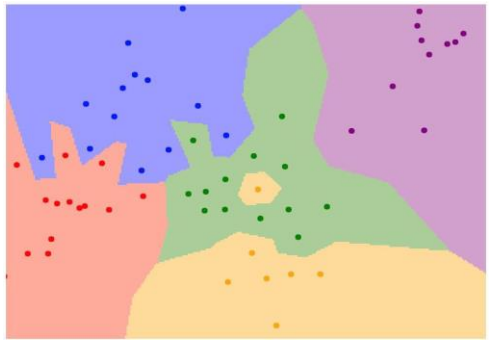
Occlusion

Clutter

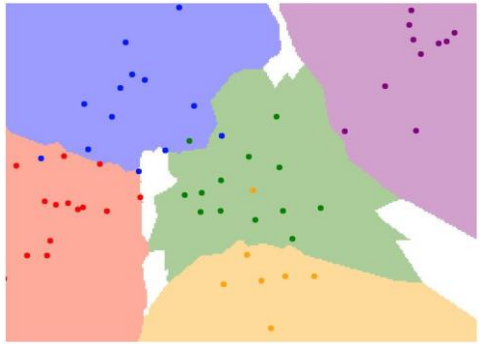
Intraclass Variation

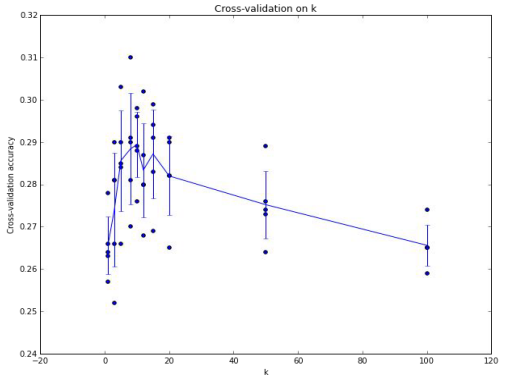
Data – Driven approach, kNN

1-NN classifier



5-NN classifier

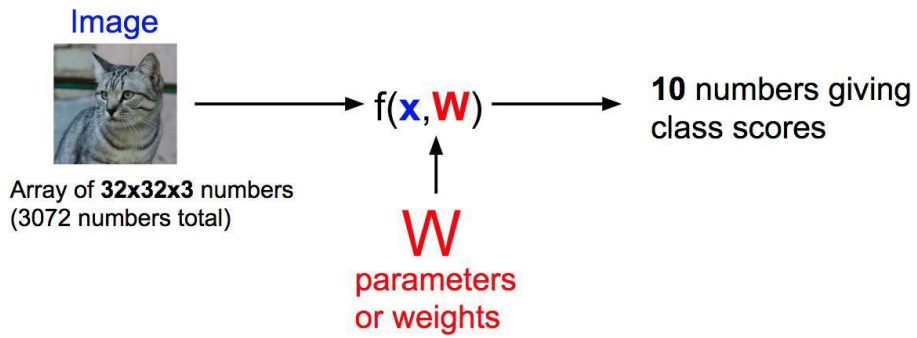




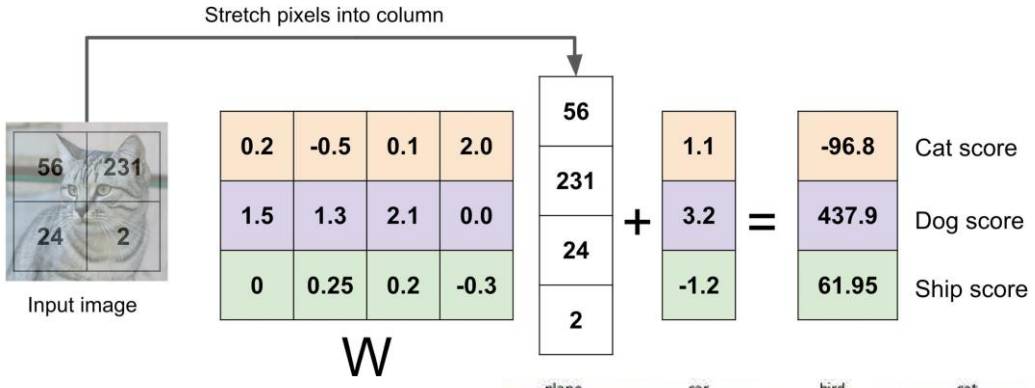
train	test

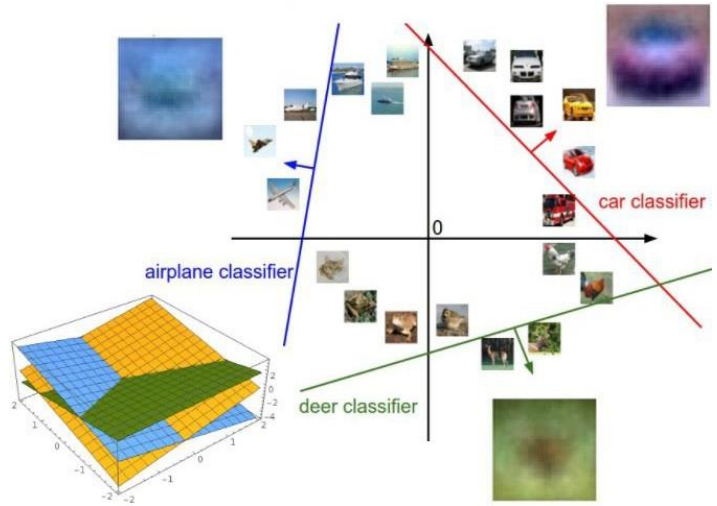
train	validation	test

Linear Classifier



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





Parametric Approach: Linear Classifier

$$\underset{10\times 1}{f(x,W)} = \underset{10\times 3073}{W}\underset{3073\times 1}{x}$$

$$\begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,3070} & w_{0,3071} & b_0 \\ w_{1,0} & w_{1,1} & \cdots & w_{1,3070} & w_{1,3071} & b_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{8,0} & w_{8,1} & \cdots & w_{8,3070} & w_{8,3071} & b_8 \\ w_{9,0} & w_{9,1} & \cdots & w_{9,3070} & w_{9,3071} & b_9 \end{pmatrix} \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \cdots \\ x_i^{(i)} \\ \cdots \\ x_{3069}^{(i)} \\ x_{3070}^{(i)} \\ x_{3071}^{(i)} \\ 1 \end{pmatrix} = \begin{pmatrix} s_0^{(i)} \\ s_1^{(i)} \\ \cdots \\ s_8^{(i)} \\ s_9^{(i)} \end{pmatrix}$$

W is good or bad?



	Cat	Automobile	Frog
Airplane	-3.45	-0.51	3.42
Automobile	-8.87	6.04	4.64
Bird	0.09	5.31	2.65
Cat	2.90	-4.22	5.10
Deer	4.48	-4.19	2.64
Dog	8.02	3.58	5.55
Frog	3.78	4.49	-4.34
Horse	1.06	-4.37	-1.50
Ship	-0.36	-2.09	-4.79
Truck	-0.72	-2.93	6.14

- Example class scores for 3 images for some W:
- How can we tell whether this W is good or bad?

Linear Classifier



	Cat	Automobile	Frog
Airplane	-3.45	-0.51	3.42
Automobile	-8.87	6.04	4.64
Bird	0.09	5.31	2.65
Cat	2.90	-4.22	5.10
Deer	4.48	-4.19	2.64
Dog	8.02	3.58	5.55
Frog	3.78	4.49	-4.34
Horse	1.06	-4.37	-1.50
Ship	-0.36	-2.09	-4.79
Truck	-0.72	-2.93	6.14

TODO:

1. Define a loss function that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function (optimization).

LOSS FUNCTION

Loss function

$$\begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,3070} & w_{0,3071} & b_0 \\ w_{1,0} & w_{1,1} & \cdots & w_{1,3070} & w_{1,3071} & b_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{8,0} & w_{8,1} & \cdots & w_{8,3070} & w_{8,3071} & b_8 \\ w_{9,0} & w_{9,1} & \cdots & w_{9,3070} & w_{9,3071} & b_9 \end{pmatrix} \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \cdots \\ x_i^{(i)} \\ \cdots \\ x_{3069}^{(i)} \\ x_{3070}^{(i)} \\ x_{3071}^{(i)} \\ 1 \end{pmatrix} = \begin{pmatrix} s_0^{(i)} \\ s_1^{(i)} \\ \cdots \\ s_8^{(i)} \\ s_9^{(i)} \end{pmatrix}$$

Loss function

- A loss function tells how good our current classifier (model, W)
- Given a datatrain (dataset) of datapoints:

$$\{(x^{(i)}, y^{(i)})\}_{i=0}^{N-1}$$

- Where:
  - + $x^{(i)}$ ảnh thứ i.
  - + $y^{(i)}$ nhãn thực ảnh thứ i.
  - + N: số điểm dữ liệu trong datatrain.
- Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}$$

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}(\hat{y}^{(i)}, y^{(i)})$$

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}(f(x^{(i)}, W), y^{(i)})$$

Loss function

– Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)} = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right)$$

– Trong đó:

+ $x^{(i)}$: là ảnh thứ i trong datatrain.

+ $L^{(i)}$: là độ mất mát của ảnh $x^{(i)}$ trong datatrain.

+ $y^{(i)}$: nhãn thực của ảnh $x^{(i)}$ trong datatrain.

+ $\hat{y}^{(i)}$: nhãn dự báo của ảnh $x^{(i)}$ trong datatrain với mô hình W.

Loss function

– Loss function (datapoint):

$$L^{(i)} = L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right) = L^{(i)}\left(s^{(i)}, y^{(i)}\right)$$

– Trong đó:

+ $L^{(i)}$: là độ mất mát của ảnh $x^{(i)}$ trong datatrain.

+ $y^{(i)}$: nhãn thực của ảnh $x^{(i)}$ trong datatrain.

+ W: mô hình dự báo.

+ $s^{(i)} = f\left(x^{(i)}, W\right)$: là điểm của ảnh $x^{(i)}$ với mô hình W tương ứng với từng lớp nhãn.

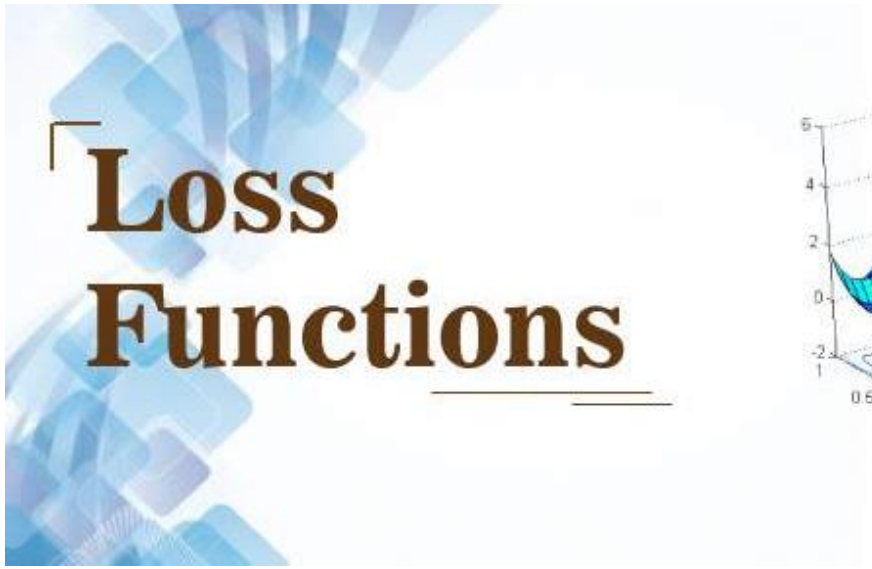
Loss function

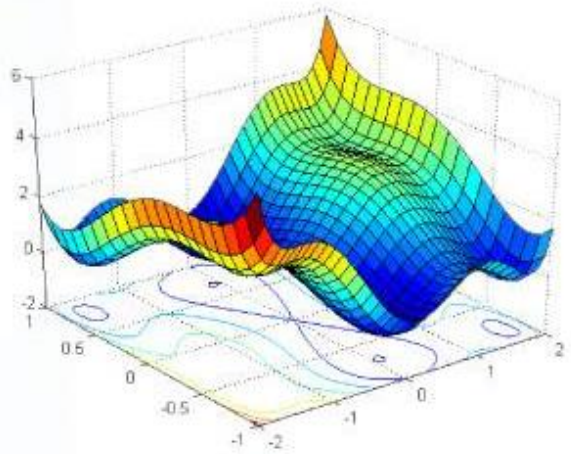
– In this lecture:

+ Multiclass SVM loss – Hinge loss

– In this lecture:

+ Cross Entropy Loss (Softmax)





Loss function

MULTICLASS SVM LOSS

SVM – Loss function

– Multiclass SVM loss has the form:

$$L^{(i)} = \sum_{j \neq y^{(i)}} \begin{cases} 0 & if \quad s_{y^{(i)}}^{(i)} \geq s_j^{(i)} + 1 \\ s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1 & otherwise \end{cases}$$

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

– $s_j^{(i)}$: điểm đánh giá của ảnh $x^{(i)}$ với lớp thứ j.

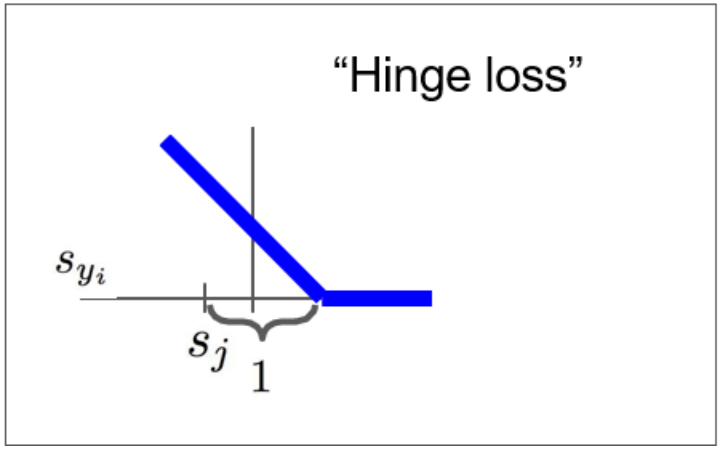
– $s_{y^{(i)}}^{(i)}$: điểm đánh giá với nhãn thực ($y^{(i)}$) của ảnh $x^{(i)}$.

SVM – Loss function

- Multiclass SVM loss has the form:

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

- $s_j^{(i)}$: điểm đánh giá của ảnh $x^{(i)}$ với lớp thứ j.
- $s_{y^{(i)}}^{(i)}$: điểm đánh giá với nhãn thực ($y^{(i)}$) của ảnh $x^{(i)}$.



SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

SVM – Loss function

- Suppose:
  + Datatrain have 3 datapoints:



  + 3 classes (cat, car, frog).
- With some W the scores $f(x, W) = Wx$.

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– With some W the scores $f(x, W) = Wx$.

Cat	3.2
Car	5.1
Frog	-1.7

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– With some W the scores $f(x, W) = Wx$.


Cat	3.2	1.3
Car	5.1	4.9
Frog	-1.7	2.0

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– With some W the scores $f(x, W) = Wx$.



Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $y^{(0)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

- Suppose:
  + Datatrain have 3 datapoints:



  + 3 classes (cat, car, frog).
- Question: $y^{(0)}$?
- Nhãn đúng của ảnh thứ nhất?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $y^{(0)}$?

– Nhãn đúng của ảnh thứ nhất?

– Trả lời: $y^{(0)} = 0$.



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $y^{(1)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $y^{(1)}$?

– Nhãn đúng của ảnh thứ hai?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

- Suppose:
  + Datatrain have 3 datapoints:



  + 3 classes (cat, car, frog).
- Question: $y^{(1)}$?
- Nhãn đúng của ảnh thứ hai?
- Trả lời: $y^{(1)} = 1$.



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $y^{(2)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $y^{(2)}$?

– Nhãn đúng của ảnh thứ ba?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

- Suppose:
  + Datatrain have 3 datapoints:



  + 3 classes (cat, car, frog).
- Question: $y^{(2)}$?
- Nhãn đúng của ảnh thứ ba?
- Trả lời: $y^{(2)} = 2$.



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s_{y^{(0)}}^{(0)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s_{y^{(0)}}^{(0)}$? Điểm ảnh thứ 1 trên nhãn đúng là bao nhiêu?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s_{y^{(0)}}^{(0)}$? Điểm ảnh thứ 1 trên nhãn đúng là bao nhiêu?

– Trả lời: 3.2 !!!!!



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s_{y^{(1)}}^{(1)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s_{y^{(1)}}^{(1)}$? Điểm ảnh thứ 2 trên nhãn đúng là bao nhiêu?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s_{y^{(1)}}^{(1)}$? Điểm ảnh thứ 2 trên nhãn đúng là bao nhiêu?

– Trả lời: 4.9 !!!!!



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s^{(2)}_{y^{(2)}}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s^{(2)}_{y^{(2)}}$? Điểm ảnh thứ 3 trên nhãn đúng là bao nhiêu?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– Question: $s^{(2)}_{y^{(2)}}$? Điểm ảnh thứ 3 trên nhãn đúng là bao nhiêu?

– Trả lời: -3.1 !!!!!



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_0^{(0)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_0^{(0)}$? Điểm ảnh thứ 1 trên nhãn 0 là bao nhiêu?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_0^{(0)}$? Điểm ảnh thứ 1 trên nhãn 0 là bao nhiêu?

– Trả lời: 3.2 !!!!!



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_1^{(0)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_1^{(0)}$? Điểm ảnh thứ 1 trên nhãn 1 là bao nhiêu?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_1^{(0)}$? Điểm ảnh thứ 1 trên nhãn 1 là bao nhiêu?

– Trả lời: 5.1 !!!!!



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_2^{(0)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_2^{(0)}$? Điểm ảnh thứ 1 trên nhãn 2 là bao nhiêu?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– $s_2^{(0)}$? Điểm ảnh thứ 1 trên nhãn 2 là bao nhiêu?

– Trả lời: -1.7 !!!!!



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

- Suppose:
  + Datatrain have 3 datapoints:



  + 3 classes (cat, car, frog).

- $s_0^{(1)}$?
- $s_1^{(1)}$?
- $s_2^{(1)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

- Suppose:
  + Datatrain have 3 datapoints:



  + 3 classes (cat, car, frog).

- $s_0^{(2)}$?
- $s_1^{(2)}$?
- $s_2^{(2)}$?



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1

SVM – Loss function

$$L^{(0)} = \sum_{j \neq y^{(0)}} max\left(0, s_j^{(0)} - s_{y^{(0)}}^{(0)} + 1\right)$$

$$L^{(0)} = \max(0, 5.1 - 3.2 + 1) + \max(0, -1.7 - 3.2 + 1)$$

$$L^{(0)} = \max(0, 2.9) + \max(0, -3.9)$$

$$L^{(0)} = 2.9 + 0$$

$$L^{(0)} = 2.9$$

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1
Loss		2.9		

SVM – Loss function

$$L^{(1)} = \sum_{j \neq y^{(1)}} max\left(0, s_j^{(1)} - s_{y^{(1)}}^{(1)} + 1\right)$$

$$L^{(1)} = \max(0, 1.3 - 4.9 + 1) + \max(0, 2.0 - 4.9 + 1)$$

$$L^{(1)} = \max(0, -2.6) + \max(0, -1.9)$$

$$L^{(1)} = 0 + 0$$

$$L^{(1)} = 0$$

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1
Loss		2.9	0	

SVM – Loss function

$$L^{(2)} = \sum_{j \neq y^{(2)}} max\left(0, s_j^{(2)} - s_{y^{(2)}}^{(2)} + 1\right)$$

$$L^{(2)} = \max(0, 2.2 - (-3.1) + 1) + \max(0, 2.5 - (-3.1) + 1)$$

$$L^{(2)} = \max(0, 6.3) + \max(0, 6.6)$$

$$L^{(2)} = 6.3 + 6.6$$

$$L^{(2)} = 12.9$$

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1
Loss		2.9	0	12.9

SVM – Loss function

– Loss over full dataset is average:

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}$$

$$L = \frac{1}{3}\left(L^{(0)} + L^{(1)} + L^{(2)}\right)$$

$$L = \frac{1}{3}(2.9 + 0 + 12.9) = 5.27$$

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$



Cat	0	3.2	1.3	2.2
Car	1	5.1	4.9	2.5
Frog	2	-1.7	2.0	-3.1
Loss		2.9	0	12.9

SVM – Loss function

- Q1: What happens to loss if car scores change a bit?
- Q2: What is the min/max possible loss?
- Q3: At initialization W is small so all $s \approx 0$. What is the loss?
- Q4: What if the sum was over all classes? (including $j = y^{(i)}$)
- Q5: What if we used Mean instead of sum?
- Q6: What if we used $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)^2$

SVM – Loss function

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

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

SVM – Loss function

- $f(x, W) = Wx$
- $L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}\left(\hat{y}^{(i)}, y^{(i)}\right)$

$$L = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j \neq y^{(i)}} max\left(0, f\left(x^{(i)}; W\right)_j^{(i)} - f\left(x^{(i)}; W\right)_{y_i}^{(i)} + 1\right)$$

- Q: Suppose that we found a W such that $L = 0$. Is this W unique?
- A: No! $2W$ is also has $L = 0$!

SVM – Loss function

– Before:

$$L^{(1)} = \sum_{j \neq y^{(1)}} max\left(0, s_j^{(1)} - s_{y^{(1)}}^{(1)} + 1\right)$$

$$L^{(1)} = \max(0, 1.3 - 4.9 + 1) + \max(0, 2.0 - 4.9 + 1)$$

$$L^{(1)} = \max(0, -2.6) + \max(0, -1.9)$$

$$L^{(1)} = 0 + 0 = 0$$

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$



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	

SVM – Loss function

– With W twice as large:

$$L^{(1)} = \sum_{j \neq y^{(1)}} max\left(0, s_j^{(1)} - s_{y^{(1)}}^{(1)} + 1\right)$$

$$L^{(1)} = \max(0, 2.6 - 9.8 + 1) + \max(0, 4.0 - 9.8 + 1)$$

$$L^{(1)} = \max(0, -6.2) + \max(0, -4.8)$$

$$L^{(1)} = 0 + 0 = 0$$

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$



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	

Loss function – Multiclass SVM Loss

REGULARIZATION

Regularization

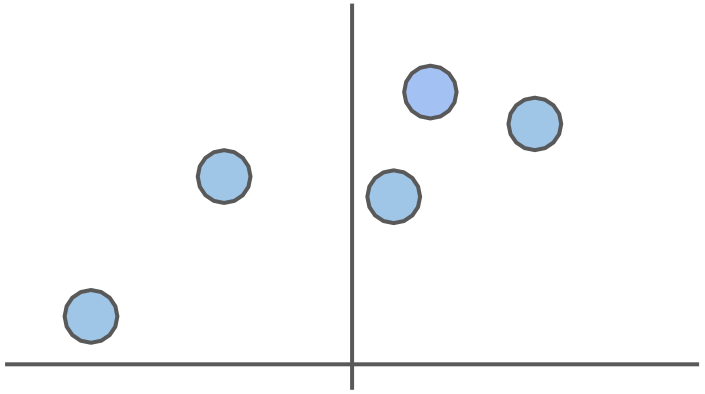
$$L = \frac{1}{N}\sum_{i} L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right)$$

Data loss: Model predictions should match training data

Regularization

$$L = \frac{1}{N}\sum_{i} L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right)$$

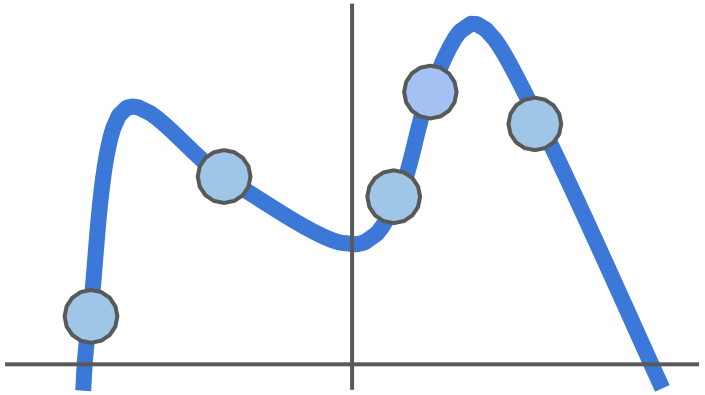
Data loss: Model predictions should match training data



Regularization

$$L = \frac{1}{N}\sum_{i} L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right)$$

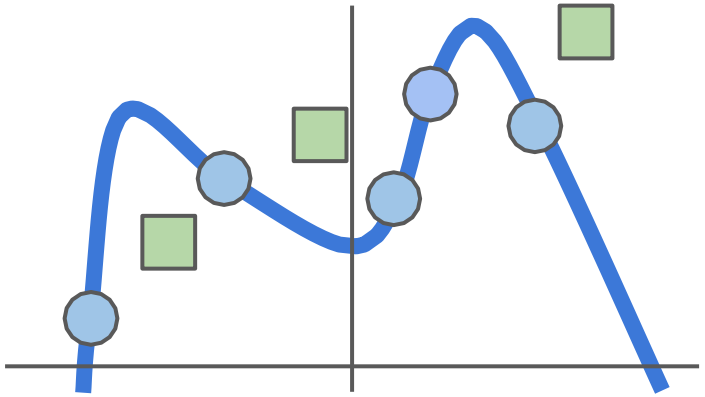
Data loss: Model predictions should match training data



Regularization

$$L = \frac{1}{N}\sum_{i} L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right)$$

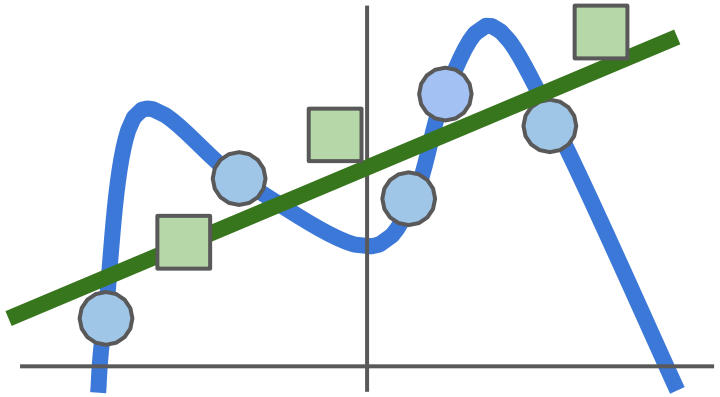
Data loss: Model predictions should match training data



Regularization

$$L = \frac{1}{N}\sum_{i} L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right)$$

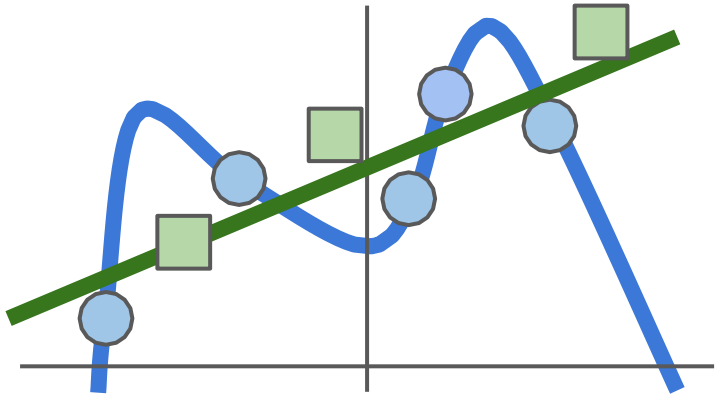
Data loss: Model predictions should match training data



Regularization

$$L = \frac{1}{N}\sum_{i} L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right) + \lambda R(W)$$

Data loss: Model predictions should match training data

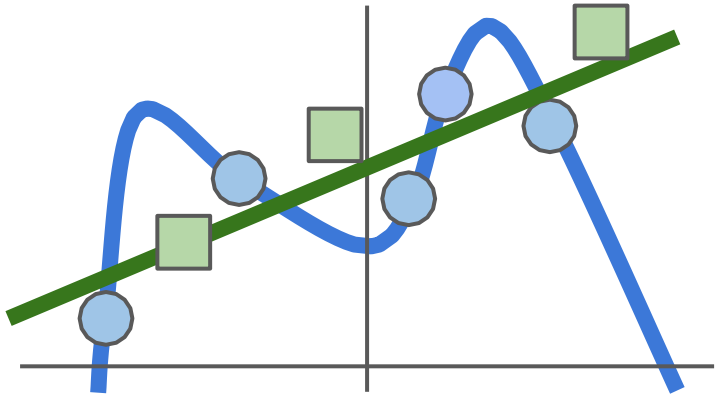


Regularization

$$L = \frac{1}{N}\sum_i L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right) + \lambda R(W)$$

Data loss: Model predictions should match training data

Regularization: Model should be "simple", so it works on test data

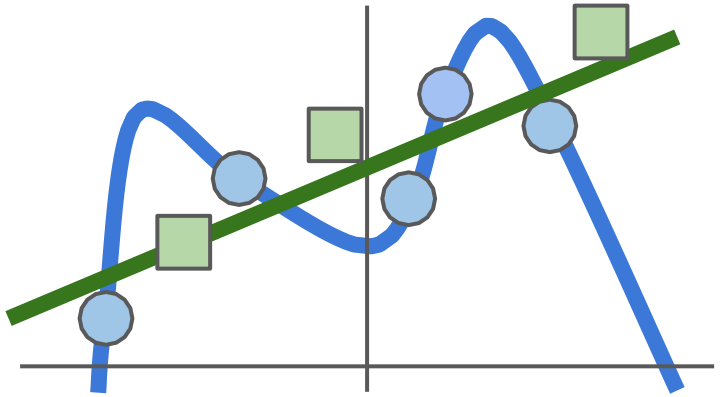


Regularization

$$L = \frac{1}{N}\sum_i L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right) + \lambda R(W)$$

Data loss: Model predictions should match training data

Regularization: Model should be "simple", so it works on test data



Occam's Razor:

"Among competing hypotheses, the simplest is the best"

William of Ockham, 1285 - 1347

Regularization

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

- λ: regularization strength (hyperparameter)
- In common use:
  1. L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$
  2. L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$
  3. Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \left(\beta W_{k,l}^2 + |W_{k,l}|\right)$
  4. …

Regularization

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

- λ: regularization strength (hyperparameter)
- In common use:
  4. Max norm regularization (might see later)
  5. Dropout (will see later)
  6. Fancier: Batch normalization, stochastic depth

L2 Regularization – Weight Decay

- $x = [1,1,1,1]$
- $w_1 = [1,0,0,0]$
- $w_2 = [0.25, 0.25, 0.25, 0.25]$

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

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

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

Loss function

CROSS ENTROPY LOSS (SOFTMAX)

Softmax – Loss function

– The equation for the SoftMax function.

$$P\left(Y = k | X = x^{(i)}\right) = \frac{e^{s_k^{(i)}}}{\sum_j e^{s_j^{(i)}}} \text{ where } s^{(i)} = f\left(x^{(i)}, W\right)$$

– Multiclass Softmax loss has the form – The negative log likelihood of the correct class:

$$L^{(i)} = -\log P\left(Y = y^{(i)} \middle| X = x^{(i)}\right) = -\log\left(\frac{e^{s_{y^{(i)}}^{(i)}}}{\sum_j e^{s_j^{(i)}}}\right)$$

Softmax – Loss function

- The equation for the SoftMax function.
- Phương trình cho hàm Softmax.

$$P\left(Y = k | X = x^{(i)}\right) = \frac{e^{s_k^{(i)}}}{\sum_j e^{s_j^{(i)}}} \text{ where } s^{(i)} = f\left(x^{(i)}, W\right)$$

- Biểu thức $P\left(Y = k | X = x^{(i)}\right)$ đọc là:
  + Xác suất có điều kiện để biến cố Y bằng k khi biến cố X bằng $x^{(i)}$ đã xảy ra.
  + Xác suất để ảnh thứ i có nhãn k.

Softmax – Loss function

– Multiclass Softmax loss has the form – The negative log likelihood of the correct class:

$$L^{(i)} = -\log P\left(Y = y^{(i)} \middle| X = x^{(i)}\right) = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$$

– Biểu thức $P\left(Y = y^{(i)} \middle| X = x^{(i)}\right)$ đọc là:

+ Xác suất có điều kiện để biến cố Y bằng $y^{(i)}$ khi biến cố X bằng $x^{(i)}$ xảy ra.

+ Xác suất để ảnh thứ i có nhãn đúng $y^{(i)}$.

Softmax – Loss function

– Multiclass Softmax loss has the form – The negative log likelihood of the correct class:

$$L^{(i)} = -\log P\left(Y = y^{(i)} \middle| X = x^{(i)}\right) = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$$

– Trong đó:

+ $s_j^{(i)}$: điểm đánh giá của ảnh $x^{(i)}$ với lớp thứ j.

+ $s^{(i)}_{y^{(i)}}$: điểm đánh giá với nhãn thực ($y^{(i)}$) của ảnh $x^{(i)}$.

Softmax – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– With some W the scores $f(x, W) = Wx$.



Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1

Softmax – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– With some W the scores $f(x, W) = Wx$.

				
Cat		3.2	1.3	2.2
Car		5.1	4.9	2.5
Frog		-1.7	2.0	-3.1

Softmax – Loss function

– In summary: $L^{(i)} = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$

unnormalized probabilities

		$e^{s^{(i)}_j}$		$\frac{e^{s^{(i)}_j}}{\sum_j s^{(i)}_j}$		
Cat	3.2	→	24.50	→	0.13	→ $L^{(i)} = -\log(0.13)$
Car	5.1		164.00		0.87	
Frog	-1.7		0.18		0.00	$L_i = 0.89$

Softmax – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– With some W the scores $f(x, W) = Wx$.



Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1

Softmax – Loss function

– In summary: $L^{(i)} = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$

unnormalized probabilities

		$e^{s^{(i)}_j}$		$\frac{e^{s^{(i)}_j}}{\sum_j s^{(i)}_j}$		
Cat	1.3	→	3.67	→	0.02	→ $L_i = -\log(0.92)$
Car	4.9		134.29		0.92	
Frog	2.0		7.39		0.06	

$L_i = 0.03$

Softmax – Loss function

– Suppose:

+ Datatrain have 3 datapoints:



+ 3 classes (cat, car, frog).

– With some W the scores $f(x, W) = Wx$.



Cat	3.2	1.3	2.2
Car	5.1	4.9	2.5
Frog	-1.7	2.0	-3.1

Softmax – Loss function

– In summary: $L^{(i)} = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$

unnormalized probabilities

		$e^{s^{(i)}_j}$		$\frac{e^{s^{(i)}_j}}{\sum_j s^{(i)}_j}$		
Cat	2.2	→	9.02	→	0.42	→ $L_i = -\log(0.01)$
Car	2.5		12.18		0.57	$L_i = 2.00$
Frog	-3.1		0.04		0.01	

Softmax – Loss function

– In summary:

$$L^{(i)} = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$$

– Q1: What is the min/max possible loss softmax $L^{(i)}$?

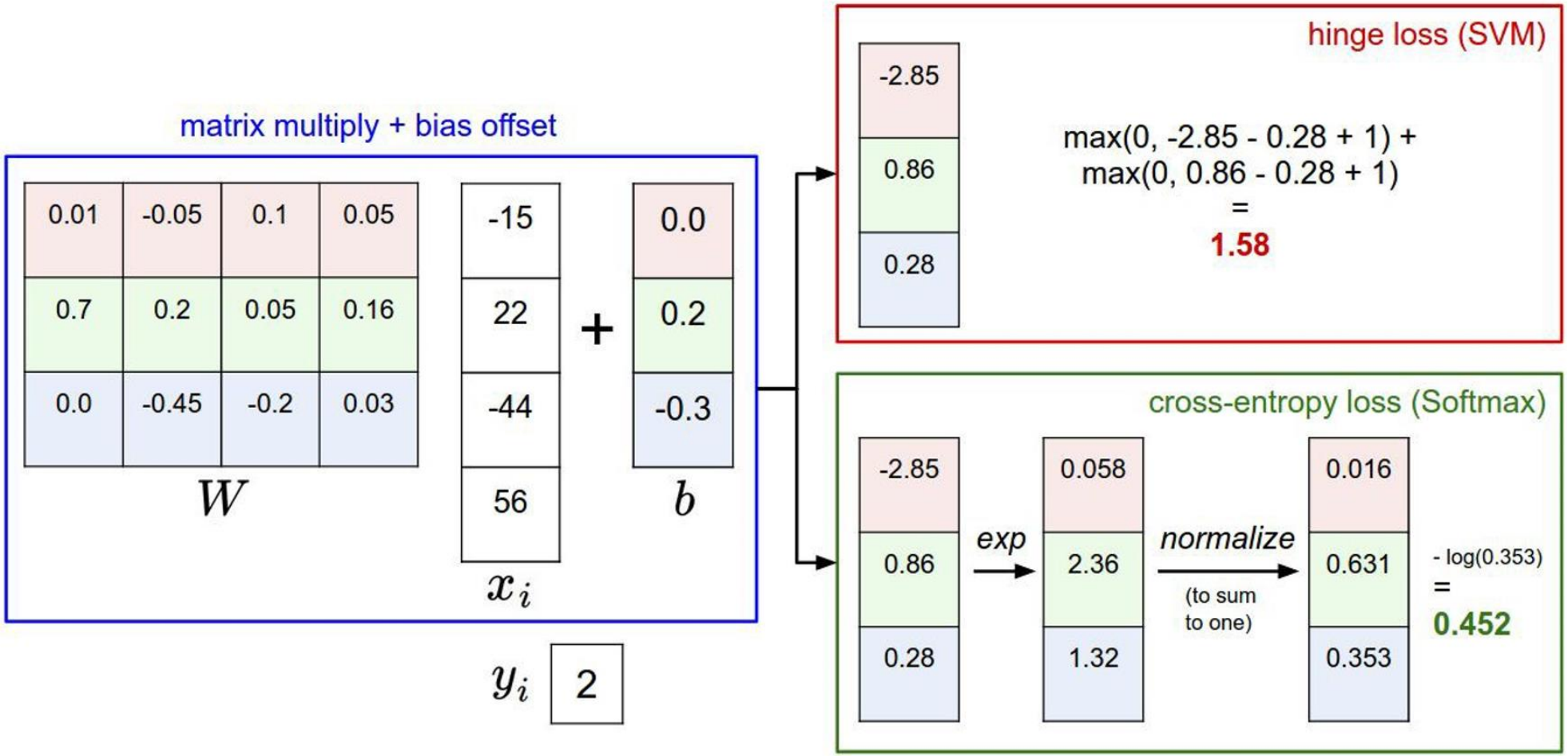
Softmax – Loss function

- In summary:

$$L^{(i)} = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$$

- Q2: Usually at initialization W is small so all $s \approx 0$. What is the loss softmax?

Softmax – Loss function



Loss function

SOFTMAX VS SVM

Softmax vs SVM

- Softmax

$$L^{(i)} = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$$

- Assume scores:

  $[10, -2, 3]$

  $[10, 9, 9]$

  $[10, 100, -100]$

  and $y_i = 0$

- SVM

$$L^{(i)} = \sum_{j \neq y_i} max\left(0, s^{(i)}_j - s^{(i)}_{y^{(i)}} + 1\right)$$

- Q: Suppose I take a datapoint and I jiggle a bit (changing its score slightly). What happens to the loss in both cases?

Loss function

RECAP

Recap Loss function

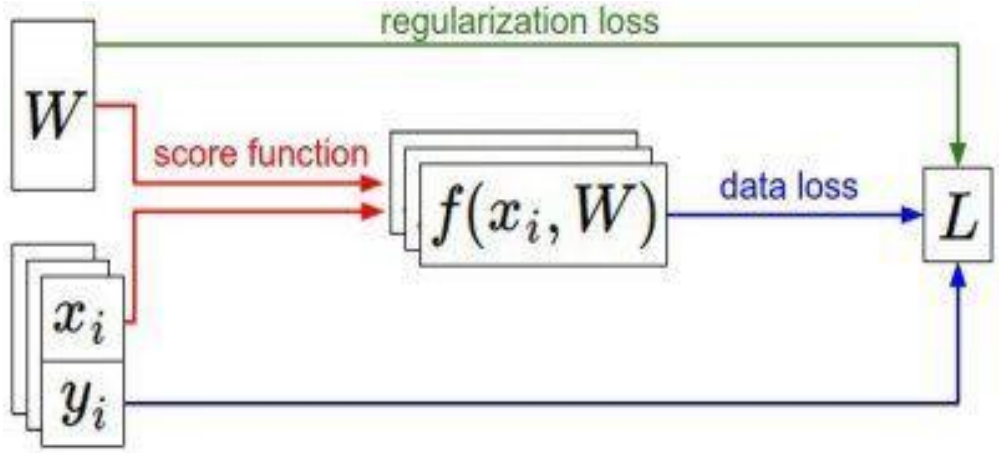
- We have some dataset of $\{(x^{(i)}, y^{(i)})\}_{i=0}^{N-1}$
- We have a score function: $s = f(x, W) = Wx$
- We have a loss function:

How do we find the best W?

+ Softmax: $L^{(i)} = -\log\left(\frac{e^{s^{(i)}_{y^{(i)}}}}{\sum_j e^{s^{(i)}_j}}\right)$

+ SVM: $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$

+ Full loss: $L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)} + R(W)$



OPTIMIZATION

Optimization

Optimization

Optimization

STRATEGY 01 – RANDOM SEARCH

Random search

Thuật toán random search

1. Định nghĩa không gian tham số: Định nghĩa phạm vi của các tham số cần tối ưu hóa.
2. Random sampling: Lấy mẫu ngẫu nhiên từ không gian tham số để tạo ra một tập hợp các giá trị tham số.
3. Đánh giá: Đánh giá hiệu suất của mỗi tập hợp tham số thông qua hàm mục tiêu hoặc mô hình.
4. Lặp lại: Lặp lại bước 2 và bước 3 trên cho đến khi đạt được số lượng lần lặp yêu cầu hoặc một điều kiện dừng nhất định (ví dụ: đạt được hiệu suất mong muốn).

Thuật toán random search

$$\begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,3070} & w_{0,3071} & b_0 \\ w_{1,0} & w_{1,1} & \cdots & w_{1,3070} & w_{1,3071} & b_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{8,0} & w_{8,1} & \cdots & w_{8,3070} & w_{8,3071} & b_8 \\ w_{9,0} & w_{9,1} & \cdots & w_{9,3070} & w_{9,3071} & b_9 \end{pmatrix} \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \cdots \\ x_i^{(i)} \\ \cdots \\ x_{3069}^{(i)} \\ x_{3070}^{(i)} \\ x_{3071}^{(i)} \\ 1 \end{pmatrix} = \begin{pmatrix} s_0^{(i)} \\ s_1^{(i)} \\ \cdots \\ s_8^{(i)} \\ s_9^{(i)} \end{pmatrix}$$

Thuật toán random search

– The loss is just a function of W:

+ $s^{(i)} = f\left(x^{(i)}; W\right) = Wx^{(i)}$

+ $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j - s_{y^{(i)}} + 1\right)$

+ $L = \frac{1}{N} \sum_{i=0}^{N-1} L^{(i)}$

Random search

```
1. bestloss = float("inf")
2. for num in xrange(1000):
3.   W = np.random.randn(10, 3073) * 0.0001
4.   loss = L(X_train, Y_train, W)
5.   if loss < bestloss:
6.     bestloss = loss
7.     bestW = W
8.    print 'in attemp %d the loss was %f, best %f' %
     (num, loss, bestloss)
```

Random search

– Let see how well this works in the test set…

```
1. scores = bestW.dot(Xte_cols)
2. Yte_predict = np.argmax(scores, axis = 0)
3. np.mean(Yte_predict == Yte)
```

➢ 15.5% accuracy! not bad! (SOTA is ~95%)

Optimization

STRATEGY 02 – RANDOM LOCAL SEARCH

Random Local Search

- Random Local Search (Tìm kiếm cục bộ ngẫu nhiên) là một thuật toán tối ưu hóa thuộc nhóm các phương pháp tìm kiếm cục bộ.
- Thuật toán Random Local Search hoạt động dựa trên nguyên lý thực hiện các bước đi ngẫu nhiên trong không gian tìm kiếm để tìm ra nghiệm tối ưu hoặc gần tối ưu của một hàm mục tiêu.

Random Local Search

- Các bước cơ bản của Random Local Search như sau:
  - + Khởi tạo ngẫu nhiên một nghiệm khả thi trong không gian tìm kiếm.
  - + Đánh giá nghiệm hiện tại bằng cách tính giá trị của hàm mục tiêu tại nghiệm đó.
  - + Sinh ra nghiệm mới bằng cách thực hiện một bước đi ngẫu nhiên từ nghiệm hiện tại.
  - + ...

Random Local Search

– Các bước cơ bản của Random Local Search như sau:

+ …

+ Đánh giá nghiệm mới:

- Nếu nghiệm mới tốt hơn nghiệm hiện tại, cập nhật nghiệm hiện tại bằng nghiệm mới.
- Nếu nghiệm mới không tốt hơn nghiệm hiện tại, có thể bỏ qua nghiệm mới hoặc tiếp tục với một số xác suất nhất định để tránh rơi vào cực tiểu cục bộ.

+ …

Random Local Search

– Các bước cơ bản của Random Local Search như sau:

+ …

+ Lặp lại quá trình trên cho đến khi đạt được điều kiện dừng (ví dụ: đạt được số lần lặp tối đa, hoặc khi giá trị hàm mục tiêu đạt yêu cầu).

Random Local Search

```
11.W = np.random.randn(10, 3073) * 0.001
12.# generate random starting W
13.bestloss = L(Xtr_cols, Ytr, W)
14.for i in range(1000):
15.    ...
```

Random Local Search

```
11.for i in range(1000):
12.     step_size = 0.0001
13.     Wtry = W + np.random.randn(10, 3073) * step_size
14.     loss = L(Xtr_cols, Ytr, Wtry)
15.     if loss < bestloss:
16.         W = Wtry
17.         bestloss = loss
18.     print 'iter %d loss is %f' % (i, bestloss)
```

Random Local Search

- Using the same number of loss function evaluations as before (1000), this approach achieves test set classification accuracy of 21.4%.
- This is better, but still wasteful and computationally expensive.

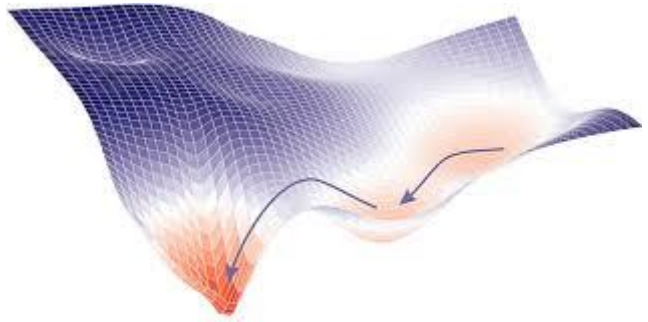
Optimization

STRATEGY 03 – FOLLOW THE SLOPE

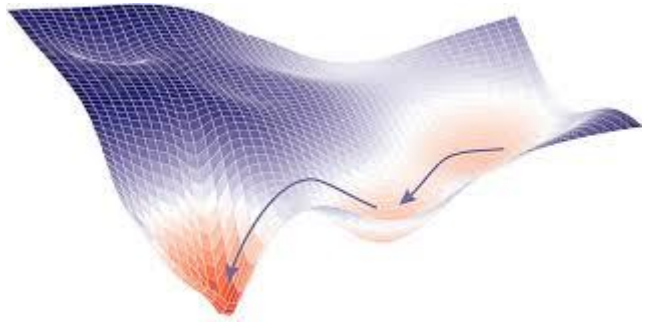
Follow the slope

Follow the slope

- "Follow the slope" trong ngữ cảnh machine learning và tối ưu hóa thường liên quan đến kỹ thuật tối ưu hóa gradient descent.
- "Follow the slope" là một phương pháp rất phổ biến để tối ưu hóa các hàm mục tiêu phức tạp, chẳng hạn như trong quá trình huấn luyện mô hình machine learning.

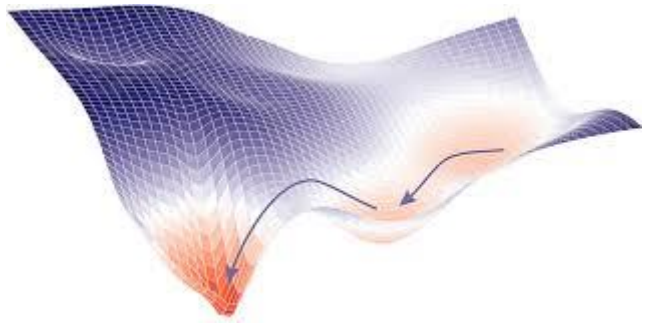


Follow the slope



Follow the slope

- Gradient Descent (Tối ưu hóa theo độ dốc): Gradient descent là một thuật toán tối ưu hóa lặp đi lặp lại để tìm giá trị tối thiểu (hoặc tối đa) của một hàm mục tiêu.
- Nguyên lý cơ bản của gradient descent là "theo dõi độ dốc" của hàm mục tiêu để điều chỉnh các tham số của mô hình nhằm giảm giá trị của hàm mục tiêu.

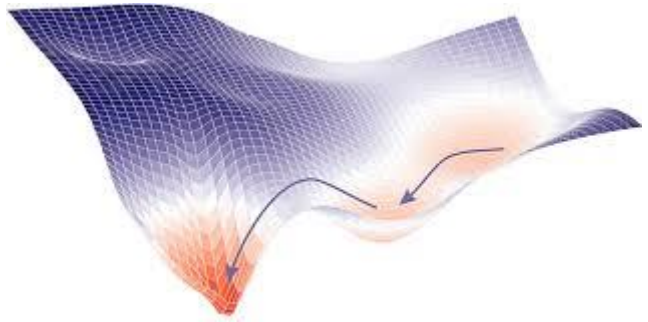


Follow the slope

– Thuật toán Gradient descent:

1. Khởi tạo tham số: Bắt đầu với các giá trị ngẫu nhiên cho các tham số cần tối ưu hóa.
2. Tính gradient: Tính đạo hàm của hàm mục tiêu đối với từng tham số. Đạo hàm cho biết hướng và độ lớn của sự thay đổi của hàm mục tiêu đối với thay đổi nhỏ trong tham số.

+ ...



Follow the slope

– Thuật toán Gradient descent:

+ …

3. Cập nhật tham số:
   - Điều chỉnh các tham số theo hướng ngược lại của gradient.
   - Hệ số học (learning rate) xác định bước điều chỉnh tham số.
4. Lặp lại: Tiếp tục quá trình tính gradient và cập nhật tham số cho đến khi hội tụ (khi sự thay đổi của hàm mục tiêu trở nên rất nhỏ) hoặc đạt số lần lặp tối đa.

Follow the slope

– Công thức cập nhật tham số:

$$\theta_{new} = \theta_{old} - \eta \nabla_\theta J(\theta)$$

+ θ là các tham số của mô hình.

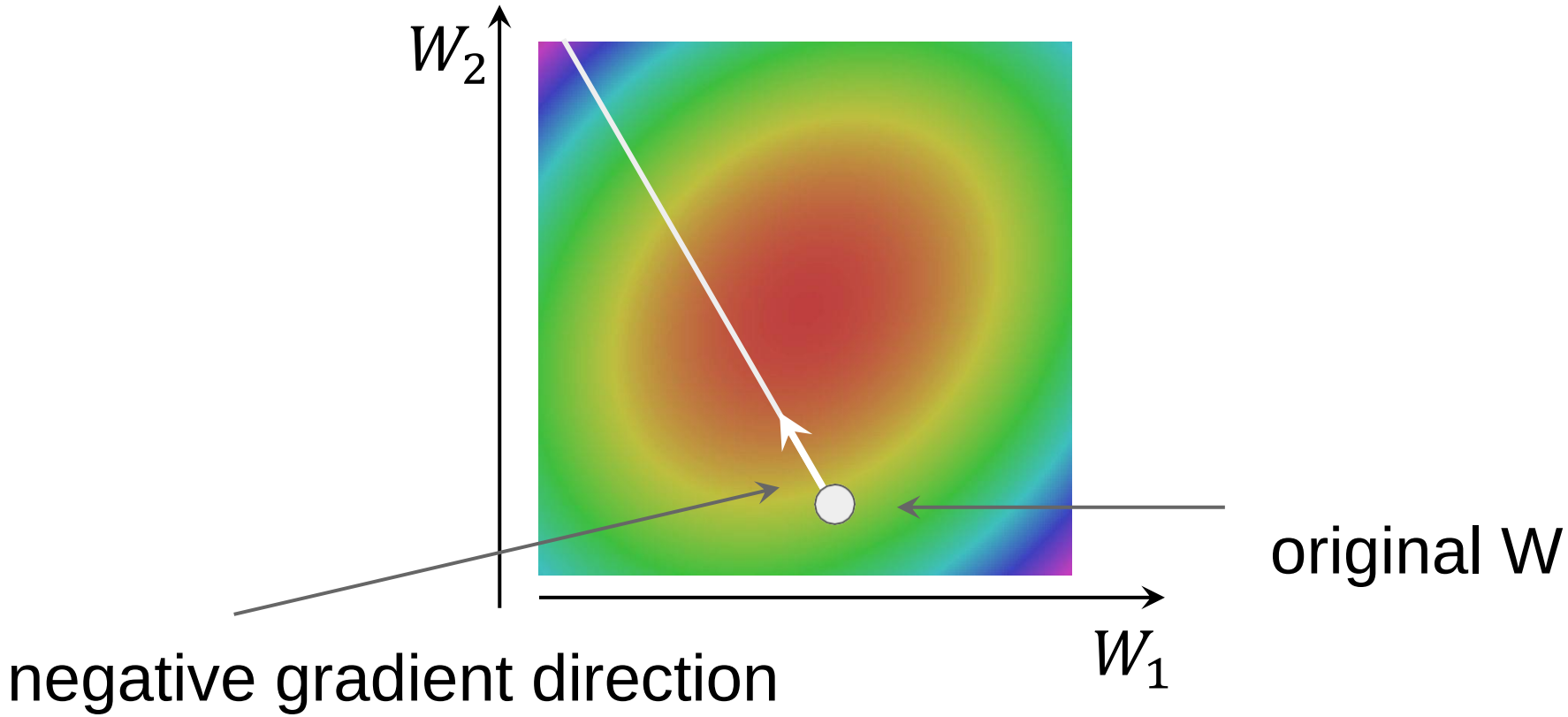
+ η là hệ số học (learning rate).

+ $\nabla_\theta J(\theta)$ là gradient của hàm mục tiêu J đối với các tham số θ.

Follow the slope

```
1. # Vanilla Gradient Descent
2. while True:
3.     weights_grad = d_gradient(loss_fun, data, weights)
4.     weights += - step_size * weights_grad
5.     # perform parameter update
```

Follow the slope



Follow the slope

- Ưu điểm của thuật toán Gradient descent:
  + Hiệu quả: Gradient descent rất hiệu quả cho việc tối ưu hóa các hàm mục tiêu phức tạp.
  + Đơn giản: Thuật toán đơn giản và dễ hiểu.

Follow the slope

– Nhược điểm của thuật toán Gradient descent:

+ Phụ thuộc vào hệ số học:

- Nếu hệ số học quá nhỏ, thuật toán sẽ hội tụ chậm.
- Nếu quá lớn, thuật toán có thể không hội tụ.

+ Tìm cực tiểu cục bộ: Thuật toán có thể dừng lại tại các cực tiểu cục bộ thay vì cực tiểu toàn cục.

Follow the slope

- Các biến thể của Gradient Descent:
  - + Batch Gradient Descent: Sử dụng toàn bộ tập dữ liệu để tính gradient.
  - + Stochastic Gradient Descent (SGD): Sử dụng một mẫu dữ liệu ngẫu nhiên để tính gradient tại mỗi bước cập nhật.
  - + Mini-batch Gradient Descent: Sử dụng một tập con ngẫu nhiên của dữ liệu để tính gradient tại mỗi bước cập nhật.

Follow the slope

Types of Gradient Descent Algorithms

Adam

RMSProp

Adagrad

Nesterov Accelerated Gradient Descent

Batch Gradient Descent

Stochastic Gradient Descent

Mini-Batch Gradient Descent

Momentum Gradient Descent

Batch Gradient Descent

$$- L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}\left(x^{(i)}, y^{(i)}, W\right) + \lambda R(W)$$

$$- \nabla_W L(W) = \frac{1}{N}\sum_{i=0}^{N-1} \nabla_W L^{(i)}\left(x^{(i)}, y^{(i)}, W\right) + \lambda \nabla_W R(W)$$

```
1. while True:
2.     weights_grad = d_gradient (loss_fun,data,weights)
3.     weights += - step_size * weights_grad
```

Mini-batch Gradient Descent

- $L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}\left(x^{(i)}, y^{(i)}, W\right) + \lambda R(W)$
- $\nabla_W L(W) = \frac{1}{N}\sum_{i=0}^{N-1} \nabla_W L^{(i)}\left(x^{(i)}, y^{(i)}, W\right) + \lambda \nabla_W R(W)$

- Full sum expensive when N is large!
- Approximate sum using a minibatch of examples $32/64/128/256/512$ common.

Mini-batch Gradient Descent

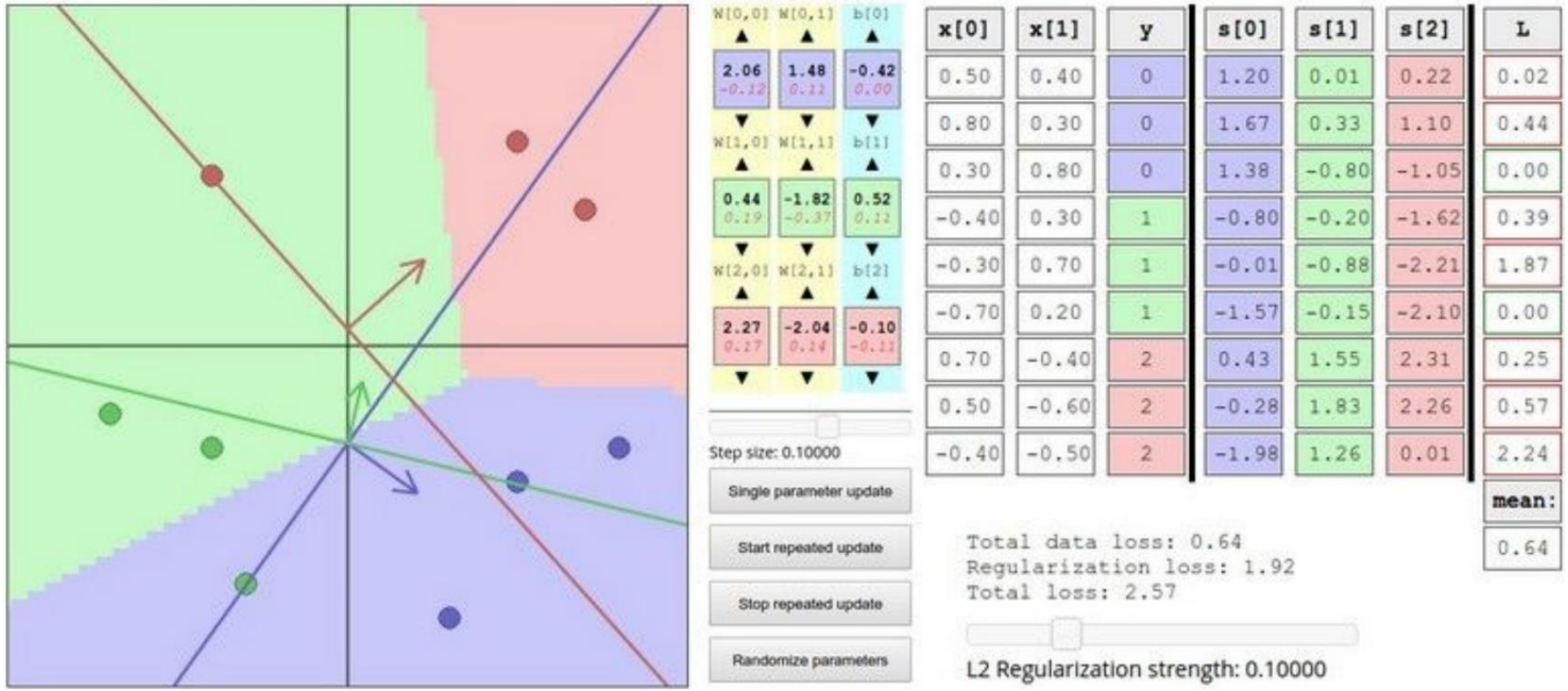
$$- L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}\left(x^{(i)}, y^{(i)}, W\right) + \lambda R(W)$$

$$- \nabla_W L(W) = \frac{1}{N}\sum_{i=0}^{N-1} \nabla_W L^{(i)}\left(x^{(i)}, y^{(i)}, W\right) + \lambda \nabla_W R(W)$$

```
1. while True:
2.     data_batch = sample_training_data(data, 256)
3.     weights_grad = d_gradient (loss_fun, data_batch,
4.                                weights)
5.     weights += - step_size * weights_grad
```

Follow the slope demo

– Interactive Web Demo time....



– http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/

Optimization

HOW TO COMPUTE GRADIENT

How to compute gradient

- The loss is just a function of W:
  - $L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}$
  - $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j - s_{y^{(i)}} + 1\right)$
  - $s^{(i)} = f\left(x^{(i)}; W\right) = W x^{(i)}$
- Want $\nabla_W L$

How to compute gradient

$$s^{(i)} = \begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,3070} & w_{0,3071} & b_0 \\ w_{1,0} & w_{1,1} & \cdots & w_{1,3070} & w_{1,3071} & b_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{8,0} & w_{8,1} & \cdots & w_{8,3070} & w_{8,3071} & b_8 \\ w_{9,0} & w_{9,1} & \cdots & w_{9,3070} & w_{9,3071} & b_9 \end{pmatrix} \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \cdots \\ x_i^{(i)} \\ \cdots \\ x_{3069}^{(i)} \\ x_{3070}^{(i)} \\ x_{3071}^{(i)} \\ 1 \end{pmatrix} = \begin{pmatrix} s_0^{(i)} \\ s_1^{(i)} \\ \cdots \\ s_8^{(i)} \\ s_9^{(i)} \end{pmatrix}$$

How to compute gradient

$$\nabla_W L = \frac{dL}{dW} = \begin{pmatrix} \frac{dL}{dw_{0,0}} & \frac{dL}{dw_{0,1}} & \cdots & \frac{dL}{dw_{0,3070}} & \frac{dL}{dw_{0,3071}} & \frac{dL}{db_0} \\ \frac{dL}{dw_{1,0}} & \frac{dL}{dw_{1,1}} & \cdots & \frac{dL}{dw_{1,3070}} & \frac{dL}{dw_{1,3071}} & \frac{dL}{db_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{dL}{dw_{8,0}} & \frac{dL}{dw_{8,1}} & \cdots & \frac{dL}{dw_{8,3070}} & \frac{dL}{dw_{8,3071}} & \frac{dL}{db_8} \\ \frac{dL}{dw_{9,0}} & \frac{dL}{dw_{9,1}} & \cdots & \frac{dL}{dw_{9,3070}} & \frac{dL}{dw_{9,3071}} & \frac{dL}{db_9} \end{pmatrix}$$

Numerical gradient

- In 1-dimension, the derivative of a function:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- In multiple dimensions, the gradient is the vector of (partial derivatives) along each dimension.
- The slope in any direction is the dot product of the direction with the gradient.
- The direction of steepest descent is the negative gradient.

Numerical gradient

- A loss function tells how good our current classifier (model, W)
- Given a datatrain (dataset) of datapoints:

$$\{(x^{(i)}, y^{(i)})\}_{i=0}^{N-1}$$

- Where:
  - + $x^{(i)}$ ảnh thứ i.
  - + $y^{(i)}$ nhãn thực ảnh thứ i.
  - + N: số điểm dữ liệu trong datatrain.
- Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}$$

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}(\hat{y}^{(i)}, y^{(i)})$$

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}(f(x^{(i)}, W), y^{(i)})$$

Numerical gradient

– Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)} = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right)$$

– Trong đó:

+ $x^{(i)}$: là ảnh thứ i trong datatrain.

+ $L^{(i)}$: là độ mất mát của ảnh $x^{(i)}$ trong datatrain.

+ $y^{(i)}$: nhãn thực của ảnh $x^{(i)}$ trong datatrain.

+ $\hat{y}^{(i)}$: nhãn dự báo của ảnh $x^{(i)}$ trong datatrain với mô hình W.

Numerical gradient

– Loss function (datapoint):

$$L^{(i)} = L^{(i)}\left(f\left(x^{(i)}, W\right), y^{(i)}\right) = L^{(i)}\left(s^{(i)}, y^{(i)}\right)$$

– Trong đó:

+ $L^{(i)}$: là độ mất mát của ảnh $x^{(i)}$ trong datatrain.

+ $y^{(i)}$: nhãn thực của ảnh $x^{(i)}$ trong datatrain.

+ W: mô hình dự báo.

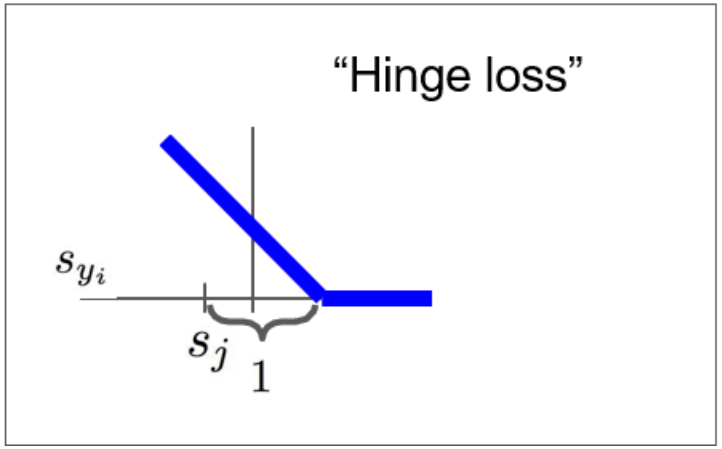
+ $s^{(i)} = f\left(x^{(i)}, W\right)$: là điểm của ảnh $x^{(i)}$ với mô hình W tương ứng với từng lớp nhãn.

Numerical gradient

- Multiclass SVM loss has the form:

$$L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$$

- $s_j^{(i)}$: điểm đánh giá của ảnh $x^{(i)}$ với lớp thứ j.
- $s_{y^{(i)}}^{(i)}$: điểm đánh giá với nhãn thực ($y^{(i)}$) của ảnh $x^{(i)}$.



Numerical gradient

- The loss is just a function of W:
  - $L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)}$
  - $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$
  - $s^{(i)} = f\left(x^{(i)}; W\right) = Wx^{(i)}$
- Want $\nabla_W L$

Numerical gradient

$$\nabla_W L = \frac{dL}{dW} = \begin{pmatrix} \frac{dL}{dw_{0,0}} & \frac{dL}{dw_{0,1}} & \cdots & \frac{dL}{dw_{0,3070}} & \frac{dL}{dw_{0,3071}} & \frac{dL}{db_0} \\ \frac{dL}{dw_{1,0}} & \frac{dL}{dw_{1,1}} & \cdots & \frac{dL}{dw_{1,3070}} & \frac{dL}{dw_{1,3071}} & \frac{dL}{db_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{dL}{dw_{8,0}} & \frac{dL}{dw_{8,1}} & \cdots & \frac{dL}{dw_{8,3070}} & \frac{dL}{dw_{8,3071}} & \frac{dL}{db_8} \\ \frac{dL}{dw_{9,0}} & \frac{dL}{dw_{9,1}} & \cdots & \frac{dL}{dw_{9,3070}} & \frac{dL}{dw_{9,3071}} & \frac{dL}{db_9} \end{pmatrix}$$

Numerical gradient

- Want $\nabla_{w_{0,0}} L$
- The loss is just a function of W:
  - $s^{(i)} = f\left(x^{(i)}; W\right) = Wx^{(i)}$
  - $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$
  - $L = \frac{1}{N} \sum_{i=0}^{N-1} L^{(i)}$

Numerical gradient

- Want $\nabla_{w_{0,0}} L$
- Gradient

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Apply

$$\frac{dL(w_{0,0})}{dw_{0,0}} = \lim_{\epsilon \to 0} \frac{L(w_{0,0} + \epsilon) - L(w_{0,0})}{\epsilon}$$

$$\frac{dL(w_{0,0})}{dw_{0,0}} = \lim_{\epsilon \to 0} \frac{L(\mathrm{W}_{w_{0,0}+\epsilon}) - L(\mathrm{W}_{w_{0,0}})}{\epsilon}$$

Numerical gradient

$$W_{w_{0,0}} = \begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,3070} & w_{0,3071} & b_0 \\ w_{1,0} & w_{1,1} & \cdots & w_{1,3070} & w_{1,3071} & b_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{8,0} & w_{8,1} & \cdots & w_{8,3070} & w_{8,3071} & b_8 \\ w_{9,0} & w_{9,1} & \cdots & w_{9,3070} & w_{9,3071} & b_9 \end{pmatrix}$$

$$W_{w_{0,0}+\epsilon} = \begin{pmatrix} w_{0,0} + \epsilon & w_{0,1} & \cdots & w_{0,3070} & w_{0,3071} & b_0 \\ w_{1,0} & w_{1,1} & \cdots & w_{1,3070} & w_{1,3071} & b_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{8,0} & w_{8,1} & \cdots & w_{8,3070} & w_{8,3071} & b_8 \\ w_{9,0} & w_{9,1} & \cdots & w_{9,3070} & w_{9,3071} & b_9 \end{pmatrix}$$

Numerical gradient

– Compute $L(W_{w_{0,0}})$

+ $s^{(i)} = f(x^{(i)}; W_{w_{0,0}}) = W_{w_{0,0}} \cdot x^{(i)}$

+ $L^{(i)}(W_{w_{0,0}}) = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$

+ $L(W_{w_{0,0}}) = \frac{1}{N} \sum_{i=0}^{N-1} L^{(i)}(W_{w_{0,0}})$

$$\frac{dL(w_{0,0})}{dw_{0,0}} = \lim_{\epsilon \to 0} \frac{L(W_{w_{0,0}+\epsilon}) - L(W_{w_{0,0}})}{\epsilon}$$

Numerical gradient

– Compute $L(W_{w_{0,0}+\epsilon})$

+ $s^{(i)} = f(x^{(i)}; W_{w_{0,0}+\epsilon}) = W_{w_{0,0}+\epsilon} \cdot x^{(i)}$

+ $L^{(i)}(W_{w_{0,0}+\epsilon}) = \sum_{j \neq y^{(i)}} max\left(0, s_j^{(i)} - s_{y^{(i)}}^{(i)} + 1\right)$

+ $L(W_{w_{0,0}+\epsilon}) = \frac{1}{N} \sum_{i=0}^{N-1} L^{(i)}(W_{w_{0,0}+\epsilon})$

$$\frac{dL(w_{0,0})}{dw_{0,0}} = \lim_{\epsilon \to 0} \frac{L(W_{w_{0,0}+\epsilon}) - L(W_{w_{0,0}})}{\epsilon}$$

Numerical gradient

$$\nabla_W L = \frac{dL}{dW} = \begin{pmatrix} \frac{dL}{dw_{0,0}} & \frac{dL}{dw_{0,1}} & \cdots & \frac{dL}{dw_{0,3070}} & \frac{dL}{dw_{0,3071}} & \frac{dL}{db_0} \\ \frac{dL}{dw_{1,0}} & \frac{dL}{dw_{1,1}} & \cdots & \frac{dL}{dw_{1,3070}} & \frac{dL}{dw_{1,3071}} & \frac{dL}{db_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{dL}{dw_{8,0}} & \frac{dL}{dw_{8,1}} & \cdots & \frac{dL}{dw_{8,3070}} & \frac{dL}{dw_{8,3071}} & \frac{dL}{db_8} \\ \frac{dL}{dw_{9,0}} & \frac{dL}{dw_{9,1}} & \cdots & \frac{dL}{dw_{9,3070}} & \frac{dL}{dw_{9,3071}} & \frac{dL}{db_9} \end{pmatrix}$$

Numerical gradient

- The loss is just a function of W:
  - $L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)} + \sum_k W_k^2$
  - $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j - s_{y^{(i)}} + 1\right)$
  - $s^{(i)} = f\left(x^{(i)}; W\right) = W x^{(i)}$
- Want $\nabla_W L$

This is silly

Numerical gradient

- current W:
  - + [0.34,
  - + −1.11,
  - + 0.78,
  - + 0.12,
  - + 0.55,
  - + 0.33, ...]
- **loss 1.25347**

- gradient dW:
  - + [?,
  - + ?,
  - + ?,
  - + ?,
  - + ?,
  - + ?, ...]

Numerical gradient

- current W:
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- $W + h$ (second dim):
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- gradient dW:
  - $+ [?,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?, \ldots]$
- **loss 1.25347**

Numerical gradient

- current W:
  - + $[0.34,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - + $[0.34 + 0.0001,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33, \ldots]$
- **loss 1.25322**

- gradient dW:
  - + $[?,$
  - + $?,$
  - + $?,$
  - + $?,$
  - + $?,$
  - + $?, \ldots]$

Numerical gradient

- current W:
  - + $[0.34,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - + $[0.34 + 0.0001,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33, \ldots]$
- **loss 1.25322**

- gradient dW:
  - + $[-2.5,$
  - + ?,
  - + ?

$$\frac{(1.25322 - 1.25347)}{0.0001} = -2.5$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Numerical gradient

- current W:
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - $+ [0.34,$
  - $+ -1.11 + 0.0001,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- **loss 1.25353**

- gradient dW:
  - $+ [-2.5,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?, \ldots]$

Numerical gradient

- current W:
  + $[0.34,$
  + $-1.11,$
  + $0.78,$
  + $0.12,$
  + $0.55,$
  + $0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  + $[0.34,$
  + $-1.11 + 0.0001,$
  + $0.78,$
  + $0.12,$
  + $0.55,$
  + $0.33, \ldots]$
- **loss 1.25353**

- gradient dW:
  + $[-2.5,$
  + $0.6,$
  + $?,$

$$\frac{(1.25353 - 1.25347)}{0.0001} = 0.6$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Numerical gradient

- current W:
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78 + 0.0001,$
  - $+ 0.12,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- **loss 1.25347**

- gradient dW:
  - $+ [-2.5,$
  - $+ 0.6,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?, \ldots]$

Numerical gradient

- current W:
  - + [0.34,
  - + −1.11,
  - + 0.78,
  - + 0.12,
  - + 0.55,
  - + 0.33, ...]
- **loss 1.25347**

- $W + h$ (second dim):
  - + [0.34,
  - + −1.11,
  - + 0.78 + 0.0001,
  - + 0.12,
  - + 0.55,
  - + 0.33, ...]
- **loss 1.25347**

- gradient dW:
  - + [−2.5,
  - + 0.6,
  - + 0.0
  - + ?,

$$\frac{(1.25347 - 1.25347)}{0.0001} = 0.0$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Numerical gradient

- current W:
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12 + 0.0001,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- **loss 1.25247**

- gradient dW:
  - $+ [-2.5,$
  - $+ 0.6,$
  - $+ 0.0,$
  - $+ ?,$
  - $+ ?,$
  - $+ ?, \ldots]$

Numerical gradient

- current W:
  - + $[0.34,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - + $[0.34,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12 + 0.0001,$
  - + $0.55,$
  - + $0.33, \ldots]$
- **loss 1.25247**

- gradient dW:
  - + $[-2.5,$
  - + $0.6,$
  - + $0,$
  - + $-10.0,$
  - + ?,

$$\frac{(1.25247 - 1.25347)}{0.0001} = -10.0$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Numerical gradient

- current W:
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ 0.55,$
  - $+ 0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ 0.55 + 0.0001,$
  - $+ 0.33, \ldots]$
- **loss 1.25387**

- gradient dW:
  - $+ [-2.5,$
  - $+ 0.6,$
  - $+ 0,$
  - $+ -10.0,$
  - $+ ?,$
  - $+ ?, \ldots]$

Numerical gradient

- current W:
  - + [0.34,
  - + −1.11,
  - + 0.78,
  - + 0.12,
  - + 0.55,
  - + 0.33, ...]
- **loss 1.25347**

- $W + h$ (second dim):
  - + [0.34,
  - + −1.11,
  - + 0.78,
  - + 0.12,
  - + 0.55 + 0.0001,
  - + 0.33, ...]
- **loss 1.25387**

- gradient dW:
  - + [−2.5,
  - + 0.6,
  - + 0,
  - + −10.0,
  - + 4.0,
  - + 2, ...]

$$\frac{(1.25387 - 1.25347)}{0.0001} = 4.0$$

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Numerical gradient

- current W:
  - + $[0.34,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - + $[0.34,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33 + 0.0001, \ldots]$
- **loss 1.25346**

- gradient dW:
  - + $[-2.5,$
  - + $0.6,$
  - + $0,$
  - + $-10.0,$
  - + $4.0,$
  - + $?, \ldots]$

Numerical gradient

- current W:
  - + $[0.34,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33, \ldots]$
- **loss 1.25347**

- $W + h$ (second dim):
  - + $[0.34,$
  - + $-1.11,$
  - + $0.78,$
  - + $0.12,$
  - + $0.55,$
  - + $0.33 + 0.0001, \ldots]$
- **loss 1.25346**

- gradient dW:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$\frac{(1.25346 - 1.25347)}{0.0001} = -0.1$$

  - + $4.0,$
  - + $-0.1, \ldots]$

Numerical gradient

- current W:
  + $[0.34,$
  + $-1.11,$
  + $0.78,$
  + $0.12,$
  + $0.55,$
  + $0.33, \ldots]$
- $W + h$ (second dim):
  + $[0.34,$
  + $-1.11,$
  + $0.78,$
  + $0.12,$
  + $0.55,$
  + $0.33, \ldots]$
- gradient dW:
  + $[-2.5,$
  + $0.6,$
  + $0,$
  + $-10.0,$
  + $4.0,$
  + $-0.1, \ldots]$
- **loss 1.25347**

Numerical gradient

- current W:
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ \mathbf{0.55},$
  - $+ 0.33, \ldots]$
- $W + h$ (second dim):
  - $+ [0.34,$
  - $+ -1.11,$
  - $+ 0.78,$
  - $+ 0.12,$
  - $+ \mathbf{0.55},$
  - $+ 0.33, \ldots]$
- gradient dW:
  - $+ [-2.5,$
  - $+ 0.6,$
  - $+ 0,$
  - $+ -10.0,$
  - $+ \mathbf{4.0},$
  - $+ -0.1, \ldots]$
- **loss 1.25347**

Numerical gradient

- The loss is just a function of W:
  - $L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)} + \sum_k W_k^2$
  - $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j - s_{y^{(i)}} + 1\right)$
  - $s^{(i)} = f\left(x^{(i)}; W\right) = W x^{(i)}$
- Want $\nabla_W L$

This is silly

In summary – How to compute gradient

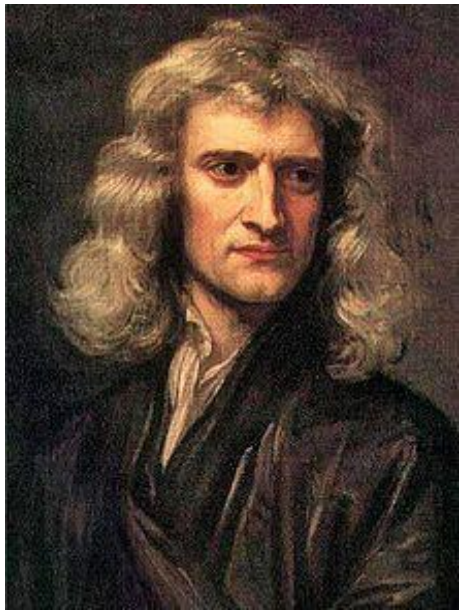
- Numerical gradient: approximate, slow, easy to write.
- Analytic gradient: exact, fast, error – prone.

➢ In practice:

+ Always use analytic gradient but check implementation with numerical gradient.
+ This is called a gradient check.

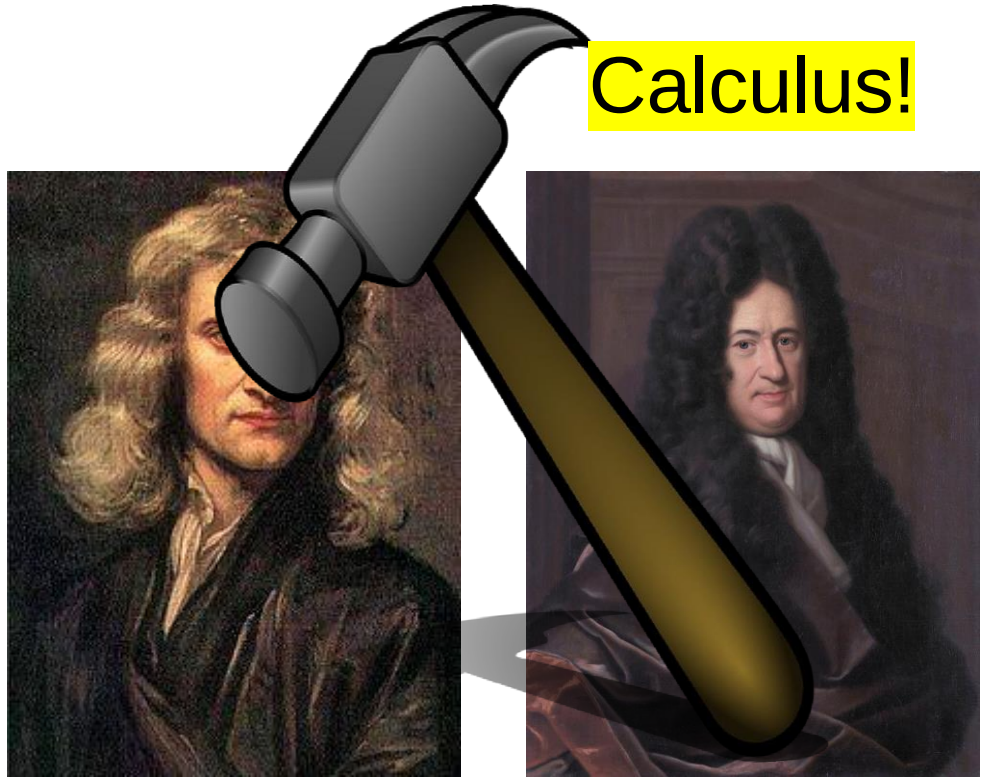
How to compute gradient

- The loss is just a function of W:
  - $L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)} + \sum_k W_k^2$
  - $L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j - s_{y^{(i)}} + 1\right)$
  - $s^{(i)} = f\left(x^{(i)}; W\right) = Wx^{(i)}$
- Want $\nabla_W L$



How to compute gradient

- The loss is just a function of W:
  - $+ L = \frac{1}{N}\sum_{i=0}^{N-1} L^{(i)} + \sum_k W_k^2$
  - $+ L^{(i)} = \sum_{j \neq y^{(i)}} max\left(0, s_j - s_{y^{(i)}} + 1\right)$
  - $+ s^{(i)} = f\left(x^{(i)}; W\right) = W x^{(i)}$
- Want $\nabla_W L$



➢ Use calculus to compute an analytic gradient.

Next time

- Introduction to neural networks
- Backpropagation

Chúc các bạn học tốt

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN TP.HCM

Nhóm UIT-Together

TS. Nguyễn Tấn Trần Minh Khang

IMAGE FEATURES

Aside: Image Features

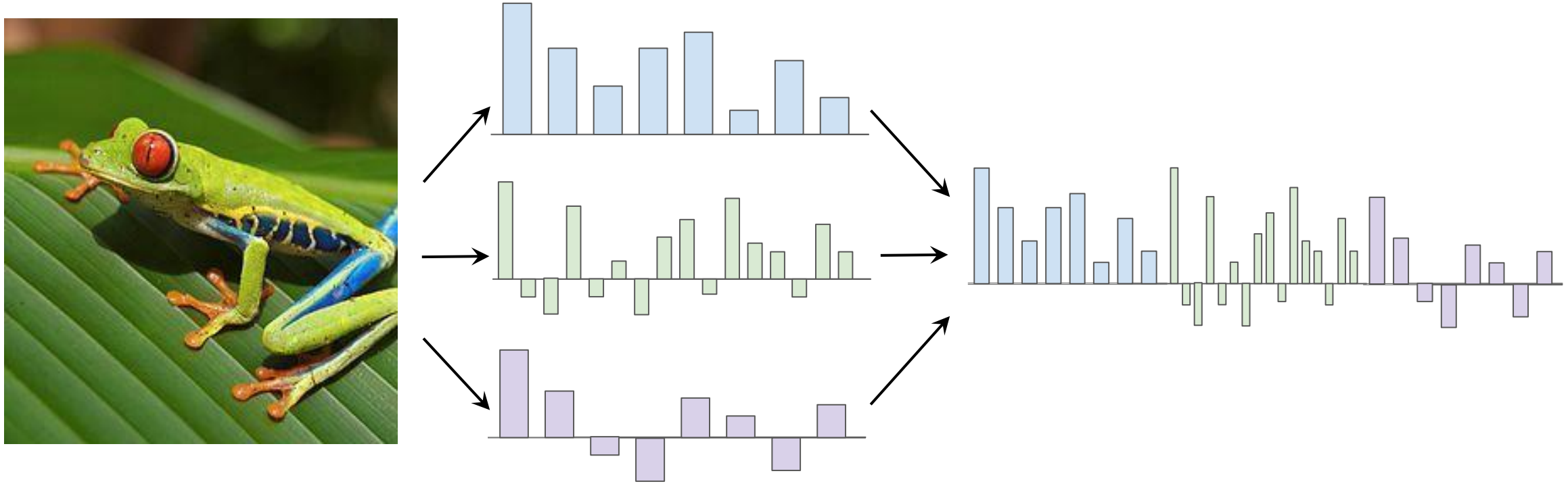
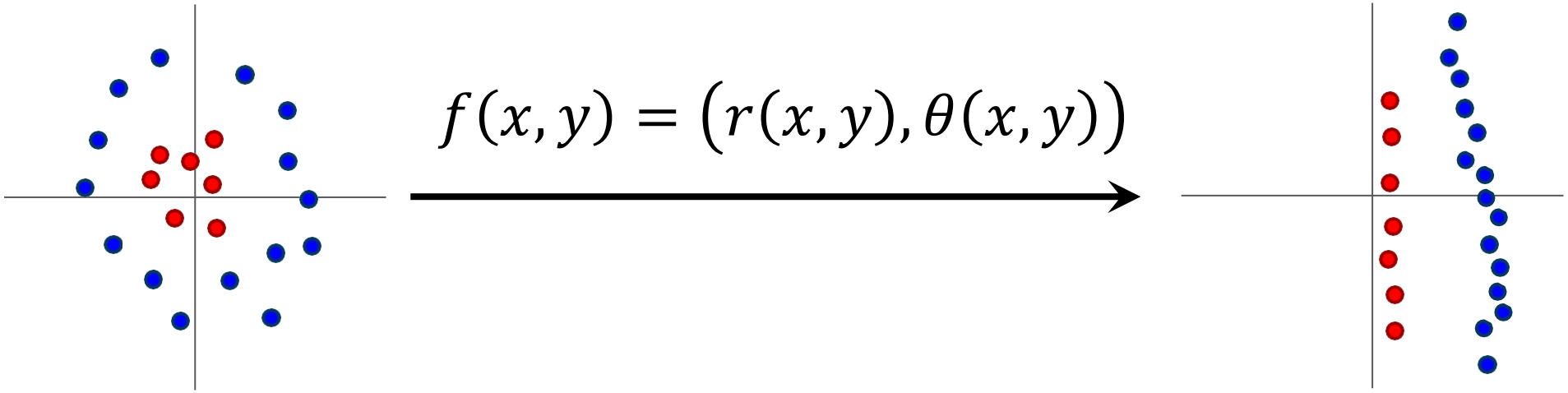
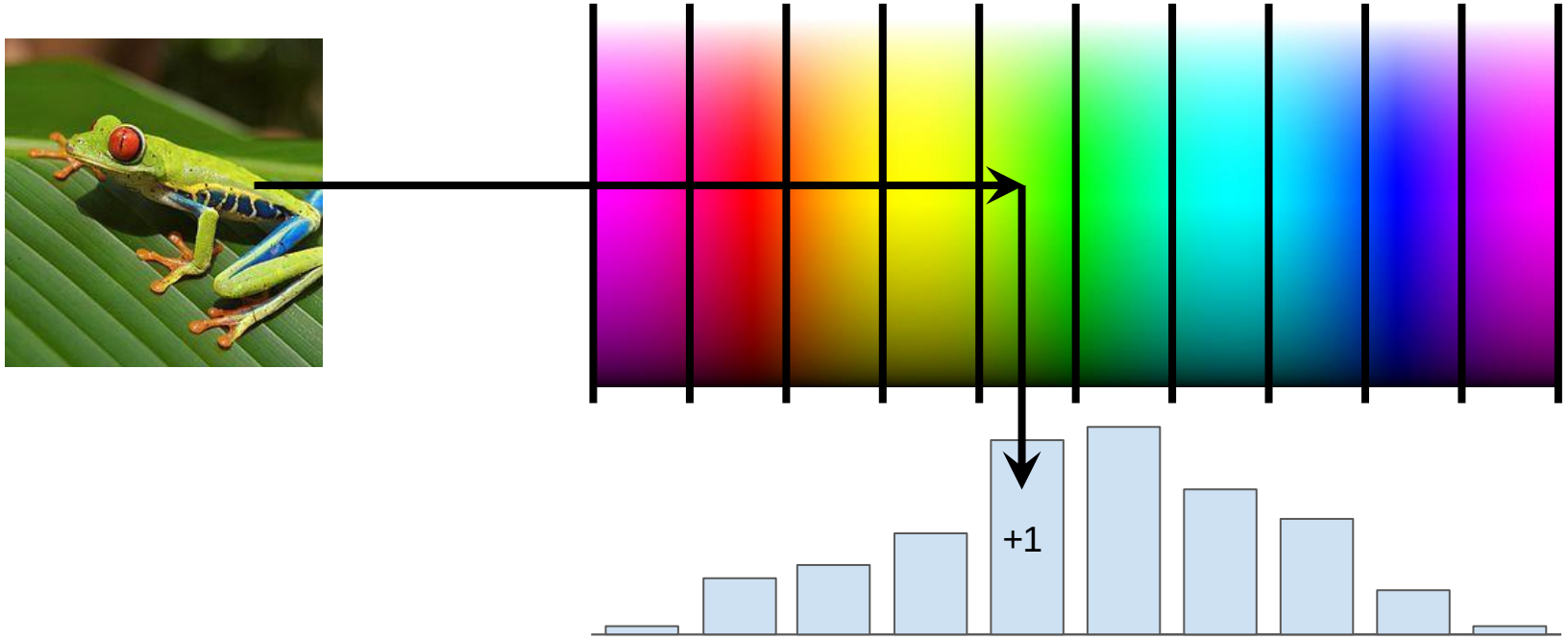


Image Features: Motivation

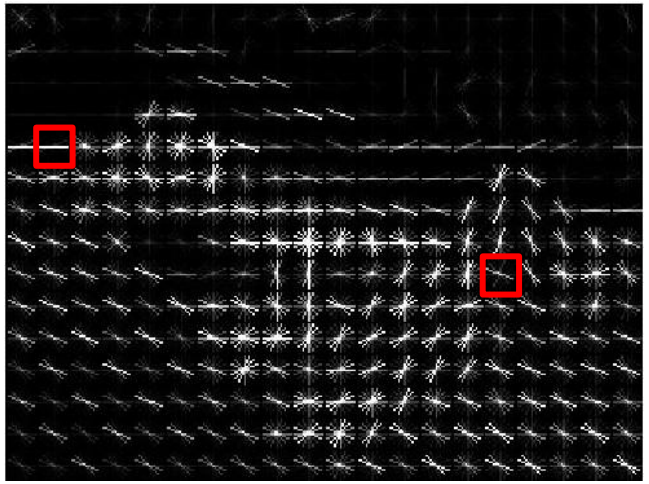


- Cannot separate red and blue points with linear classifier.
- After applying feature transform, points can be separated by linear classifier.

Example: Color Histogram



Example: HoG



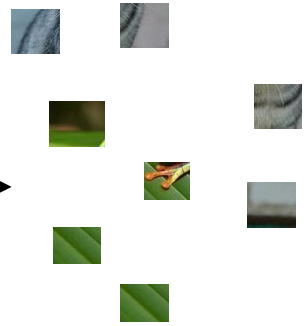
Divide image into 8×8 pixel regions Within each region quantize edge direction into 9 bins.

Example: 320×240 image gets divided into 40×30 bins; in each bin have 9 numbers so feature vector has $30 \times 40 \times 9 = 10{,}800$ numbers.

Example: Bag of Words

– Step 1: Build codebook

Extract random patches



Cluster patches to form “codebook” of “visual words”

– Step 2: Encode images

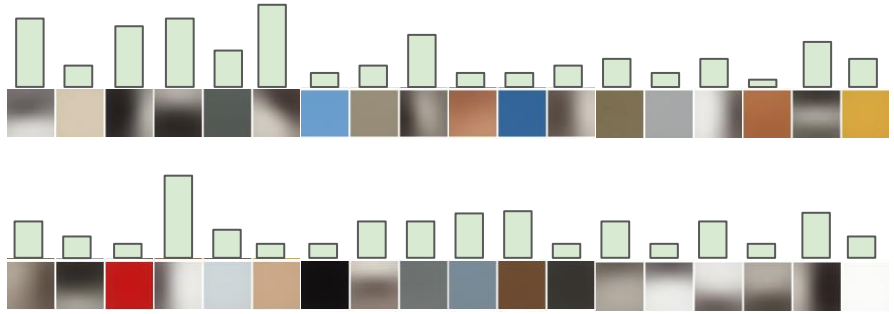
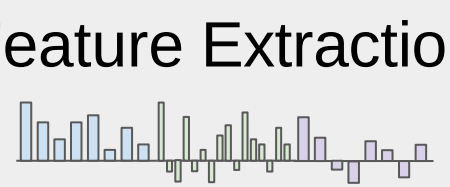


Image features vs ConvNets

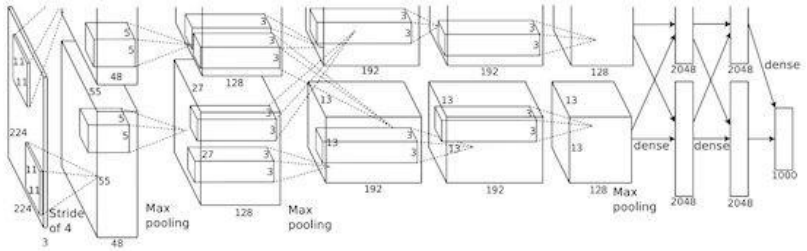
Feature Extraction



f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

10 numbers giving scores for classes

training