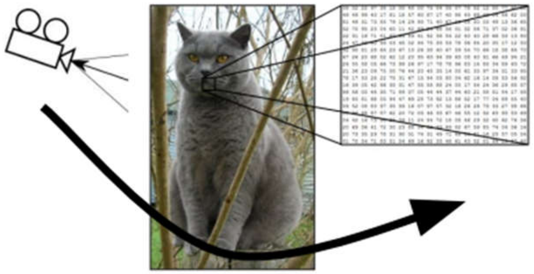
Lecture 3: Linear Classification 2, Optimization

박사과정 김성빈 chengbinjin@inha.edu,
지도교수 김학일 교수 hikim@inha.ac.kr
인하대학교 컴퓨터비전 연구실

Recall From Last Time…

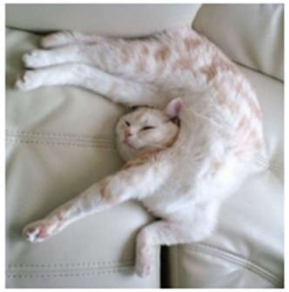
- Challenges in Visual Recognition

Camera pose



Illumination

Deformation



Occlusion

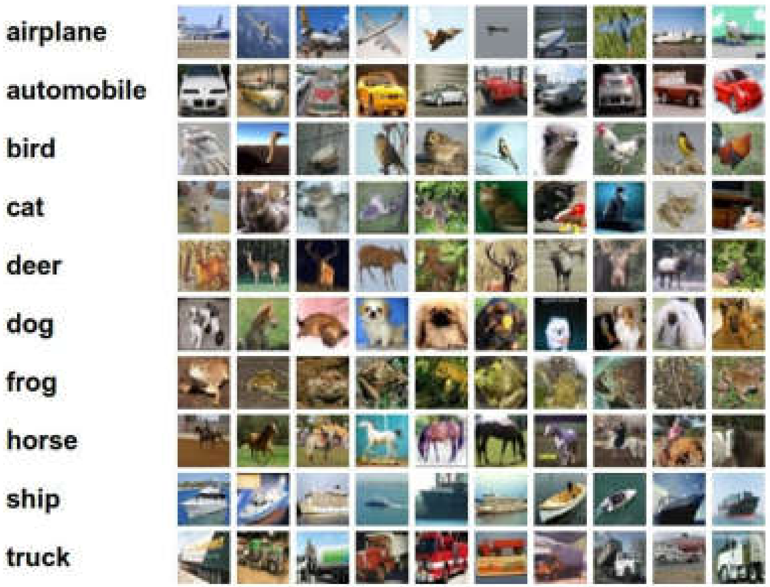
Background clutter

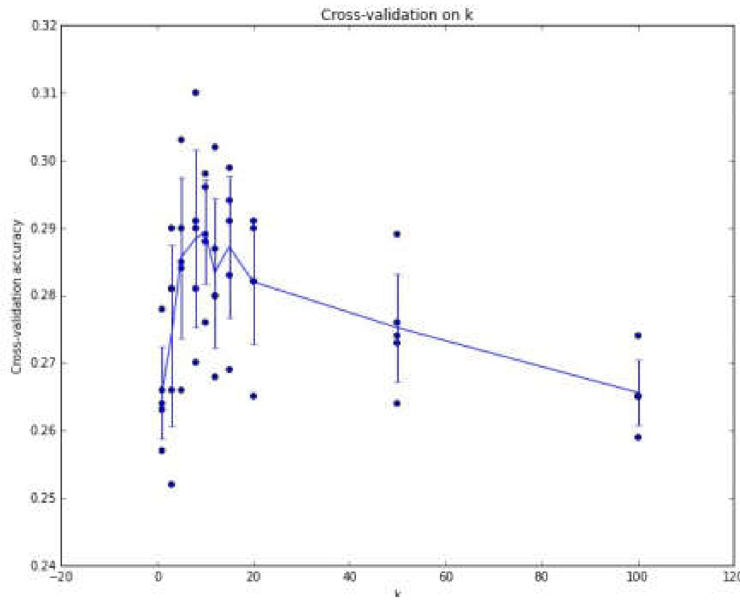
Intraclass variation

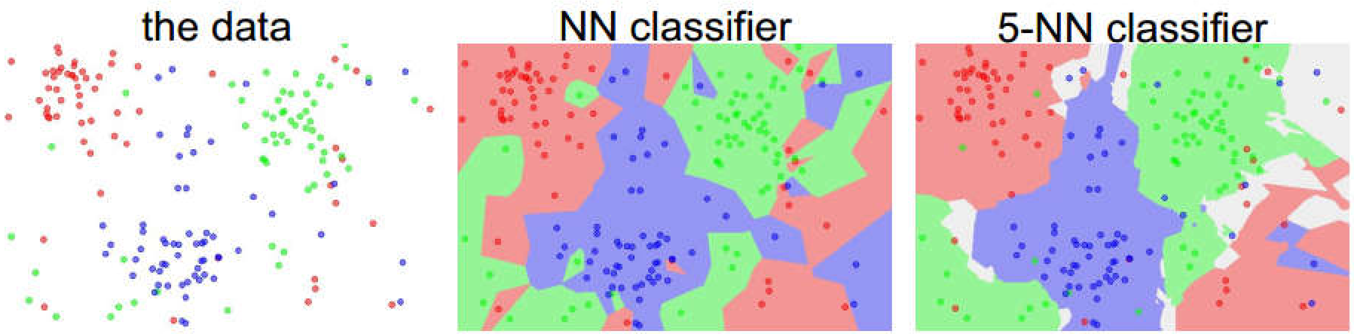
Slide from CS231n

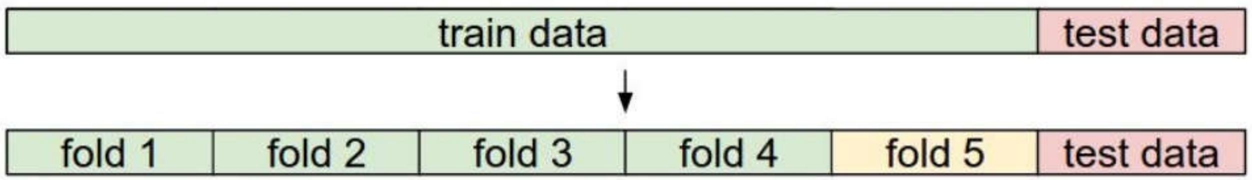
Recall From Last Time…

- Data-driven approach, kNN









Slide from CS231n

Recall From Last Time…

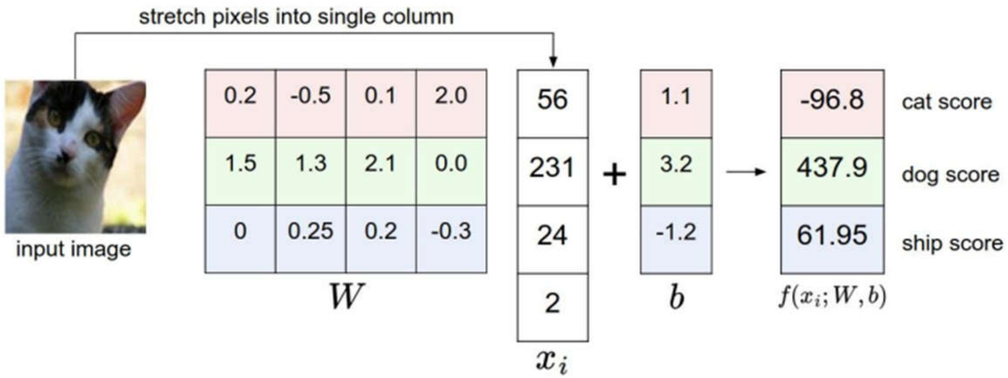
- Linear classifier

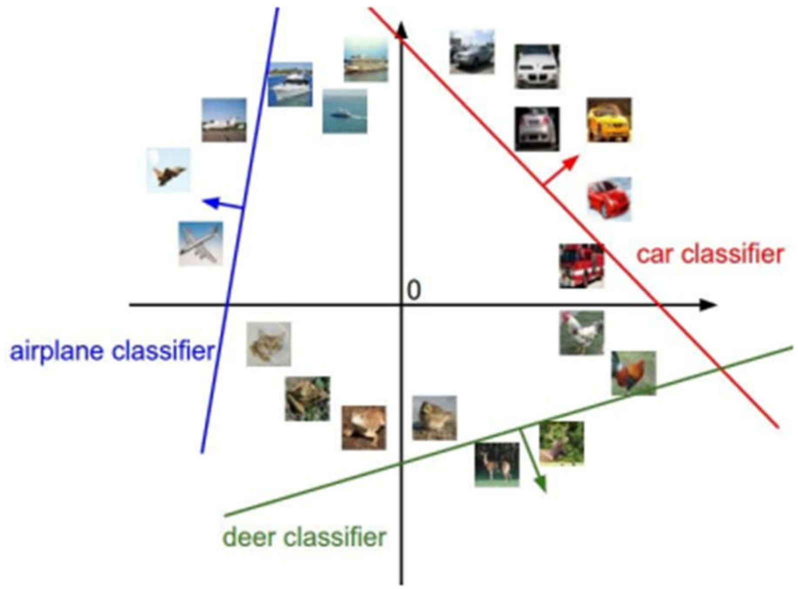
image parameters

$f(\mathbf{x},\mathbf{W})$

[32x32x3]
(3072 numbers total)

10 numbers, indicating class scores





Slide from CS231n

Loss Function & Optimization

- Going forward: Loss function/Optimization



airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
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	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

Random normalized W

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)**

Slide from CS231n

SVM Loss

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, b) = \mathbf{W}x + b$ are:



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

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:



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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:

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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

= 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

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:

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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

= 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

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:

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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

= max(0, 2.2 – (-3.1) + 1) + max(0, 2.5 – (-3.1) + 1)
= max(0, 6.3) + max(0, 6.6)
= 6.3 + 6.6
= 12.9

Slide from CS231n

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:

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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

And the full training loss is the mean over all examples in the training data:

$$\boxed{L = \frac{1}{N} \sum_{i=1}^{N} L_i}$$

L = (2.9 + 0 + 12.9) / 3
= **5.27**

Slide from CS231n

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:



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

Question #1: what's the meaning of the SVM loss?

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

And the full training loss is the mean over all examples in the training data: $\boxed{L = \frac{1}{N} \sum_{i=1}^{N} L_i}$

L = (2.9 + 0 + 12.9) / 3
= **5.27**

Slide from CS231n

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:

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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

Question #2: what if the sum was instead over all classes? (including $j = y_i$)

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:



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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

$$L_i = \boxed{\sum}_{j \neq y_i} \max\left(0, s_j - s_{y_i} + 1\right)$$

Question #3: what if we used a mean instead of a sum here? Does it influence final **W**?

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:

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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

Question #4: what if we used

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

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:

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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

Question #5: what is the min/max possible loss?

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$ are:



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

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: $s = f(x_i, \mathbf{W}, \mathrm{b})$

the SVM loss has the form:

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

Question #6: usually at initialization W are small numbers, so all s~= 0. What is the loss?

SVM Numpy Code

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

import numpy as np

```
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
```

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

= 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

Slide from CS231n

Bug of SVM Loss Function

- There is a bug with the loss:

$$f\left(x, \mathbf{W}, \mathrm{b}\right) = \mathbf{W}x + \mathrm{b}$$

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

Bug of SVM Loss Function

- There is a bug with the loss:

$$f(x, \mathbf{W}, \mathrm{b}) = \mathbf{W}x + \mathrm{b}$$

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

E.Q. Suppose that we found a W such that L = 0. Is this W unique?

SVM Loss

- Suppose: 3 training examples, 3 classes
- with some W scores $f(x, \mathbf{W}) = \mathbf{W}x$ are:

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

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

Before:

= 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

With W twice as large:

= 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

Weight Regularization

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

- Lambda: regularization strength (hyperparameter)

In common use:

- **L2 regularization** $$R(\mathbf{W}) = \sum_k \sum_l \mathbf{W}_{k,l}^2$$
- L1 regularization $$R(\mathbf{W}) = \sum_k \sum_l \left|\mathbf{W}_{k,l}\right|$$ **(Sparse coding)**
- Elastic net (L1 + L2) $$R(\mathbf{W}) = \sum_k \sum_l \beta \mathbf{W}_{k,l}^2 + \left|\mathbf{W}_{k,l}\right|$$
- Dropout (will see later)

- Trade off between **training loss** and **generalization loss**

Slide from CS231n

L2 Regularization: Motivation

$\boldsymbol{x} = [1, 1, 1, 1]$

$\boldsymbol{w}_1 = [1, 0, 0, 0]$

$\boldsymbol{w}_2 = [0.25, 0.25, 0.25, 0.25]$

$$\boldsymbol{w}_1^T \boldsymbol{x} = \boldsymbol{w}_2^T \boldsymbol{x} = 1$$

Question #7: which W L2 regularization like?

Slide from CS231n

Softmax Classifier and Softmax Loss

Softmax Classifier (Multinomial Logistic Regression)

Cat	**3.2**
Car	5.1
Frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

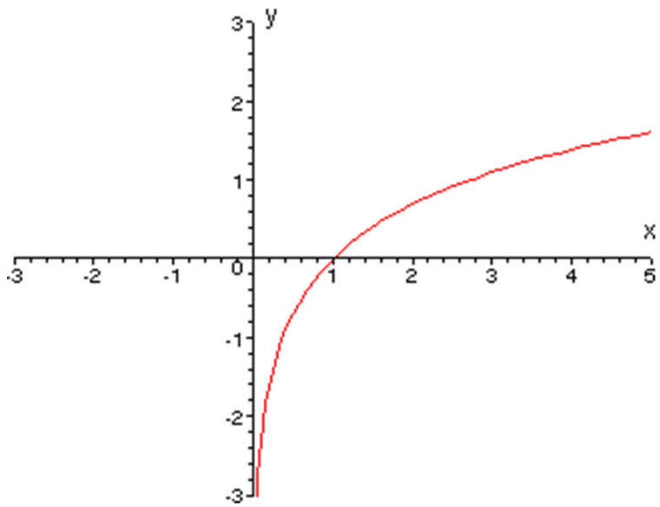
Scores = unnormalized log probabilities of the classes.

$$\boxed{s = f(x_i; W)}$$

Cat	**3.2**
Car	5.1
Frog	-1.7

$$s = \log(z)$$

$$s = \log(e^s)$$



Softmax Classifier (Multinomial Logistic Regression)

Scores = unnormalized log probabilities of the classes.

$$\boxed{P(Y=k|X=x_i)=\frac{e^{s_k}}{\sum_j e^{s_j}}}$$ where $\boxed{s=f(x_i;\mathbf{W},\mathrm{b})}$

Cat	**3.2**
Car	5.1
Frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Cat **3.2**

Car 5.1

Frog -1.7

Scores = unnormalized log probabilities of the classes.

$$\boxed{P(Y=k|X=x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}} \quad \text{where} \quad \boxed{\boldsymbol{s} = f(x_i; \mathbf{W})}$$

Softmax function

Softmax Loss (Multinomial Logistic Regression)

Cat	**3.2**
Car	5.1
Frog	-1.7

Scores = unnormalized log probabilities of the classes.

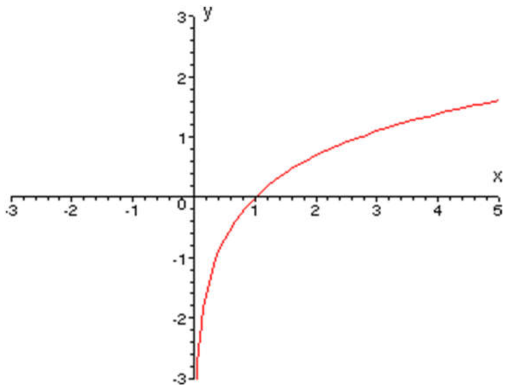
$$\boxed{P(Y=k|X=x_i)=\frac{e^{s_k}}{\sum_j e^{s_j}}}$$ where $$\boxed{s=f(x_i;\mathbf{W})}$$

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

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

$$s=\log(z)$$

$$s=\log(e^s)$$



Softmax Loss (Multinomial Logistic Regression)

Cat	**3.2**
Car	5.1
Frog	-1.7

Scores = unnormalized log probabilities of the classes.

$$\boxed{P(Y=k|X=x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ where $$\boxed{\mathbf{s} = f(x_i;\mathbf{W})}$$

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

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

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

Example of the Softmax

Cat	**3.2**
Car	5.1
Frog	-1.7

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \qquad \mathbf{s} = f(x_i; \mathbf{W})$$

unnormalized log probabilities

Example of the Softmax

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

	unnormalized log probabilities	exp	unnormalized probabilities	normalize	probabilities
Cat	**3.2**		**24.5**		**0.13**
Car	5.1	→	164.0	→	0.87
Frog	-1.7		0.18		0.00

Example of the Softmax

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

unnormalized probabilities

	unnormalized log probabilities	$\xrightarrow{\text{exp}}$	unnormalized probabilities	$\xrightarrow{\text{normalize}}$	probabilities
Cat	**3.2**		**24.5**		**0.13**
Car	5.1		164.0		0.87
Frog	-1.7		0.18		0.00

$\longrightarrow L_i = -\log(0.13) = \mathbf{0.89}$

unnormalized log probabilities

probabilities

Example of the Softmax

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

Question #8: what is the min/max possible loss L_i? When?

unnormalized probabilities

		$\xrightarrow{\text{exp}}$		$\xrightarrow{\text{normalize}}$		
Cat	**3.2**		**24.5**		**0.13**	$\rightarrow L_i = -\log(0.13) = \mathbf{0.89}$
Car	5.1		164.0		0.87	
Frog	-1.7		0.18		0.00	

unnormalized log probabilities

probabilities

Example of the Softmax

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

Question #9: usually at initialization W are small numbers, so all s~=0. What is the loss?

unnormalized probabilities

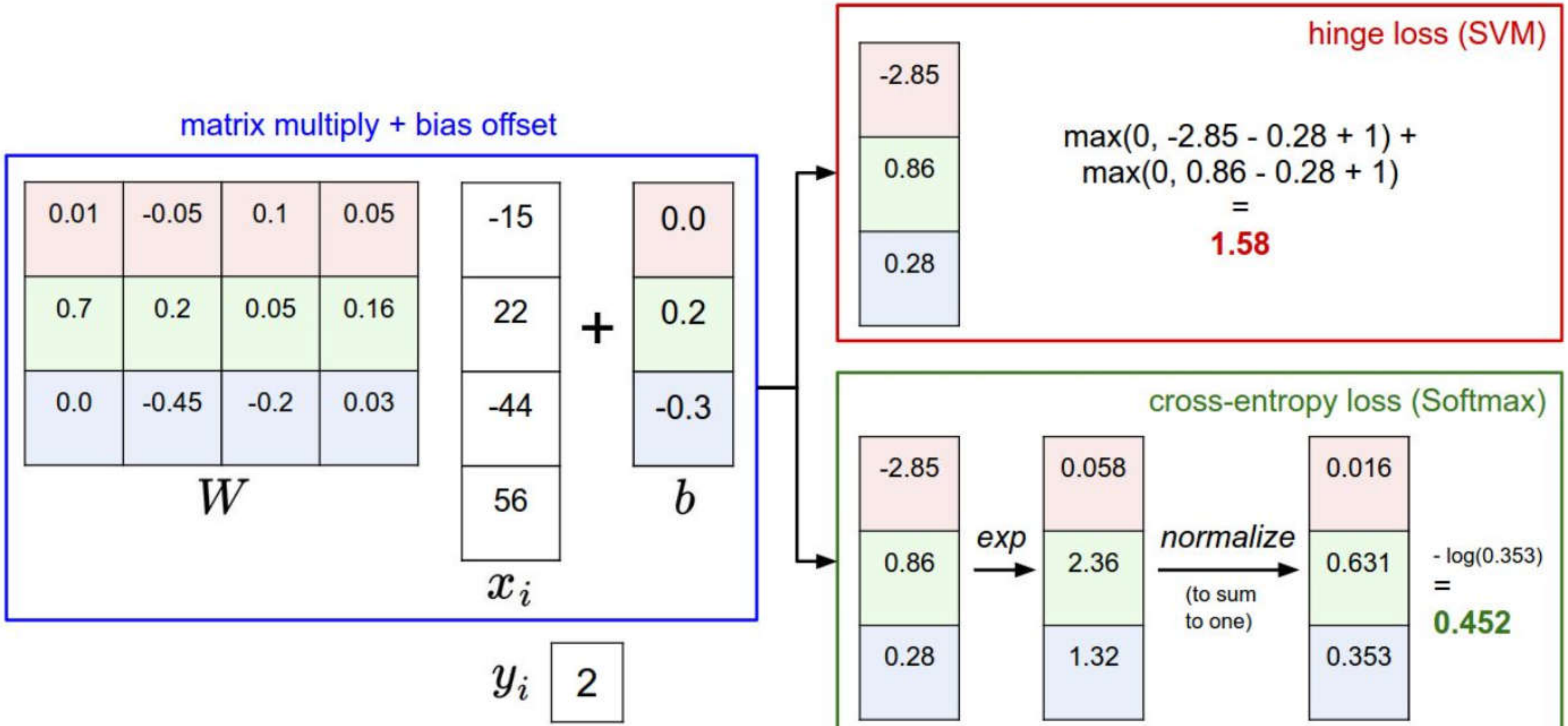
Cat	**3.2**	**24.5**	**0.13**	$L_i = -\log(0.13) = \mathbf{0.89}$
Car	5.1	164.0	0.87	
Frog	-1.7	0.18	0.00	

exp → normalize →

unnormalized log probabilities

probabilities

Flowchart of the SVM and Softmax Loss



Slide from CS231n

Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \qquad L_i = \sum_{i \neq y_i} \max\left(0, s_j - s_{y_i} + 1\right)$$

Assume scores:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

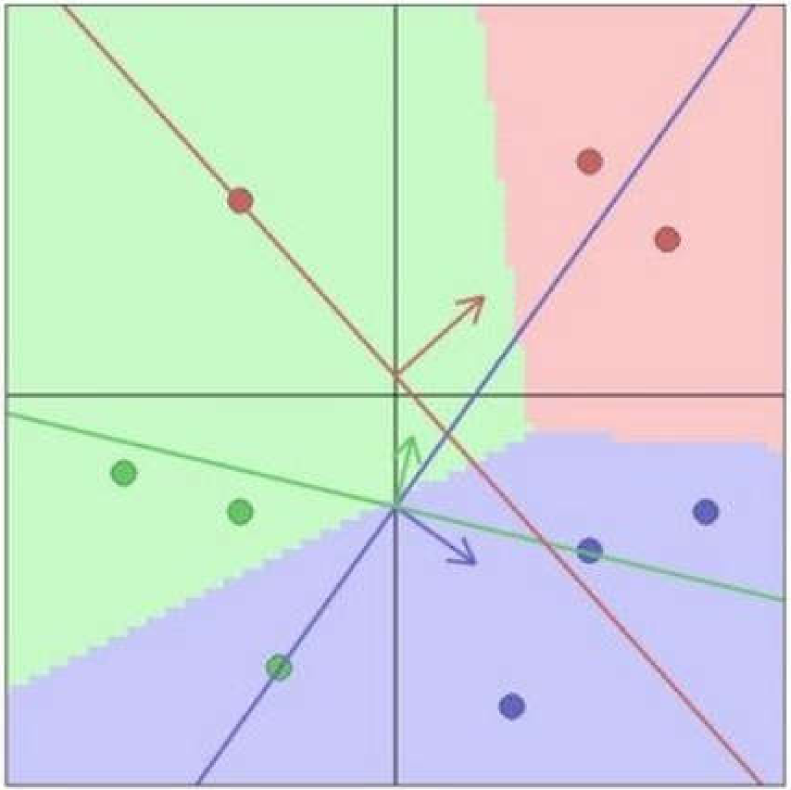
and $\boxed{y_i = 0}$

Question #10: Suppose I take a data point and I jiggle a bit (changing its scores slightly). What happens to the loss in both cases?

Note: Margin 1 in SVM is not a hyperparameter, our W can be controlled in Regularization Term, so we don't need worry about 1.

Slide from CS231n

Interactive Web Demo



W[0,0]	W[0,1]	b[0]
2.06 -0.12	1.48 0.11	-0.42 0.00
W[1,0]	**W[1,1]**	**b[1]**
0.44 0.19	-1.82 -0.37	0.52 0.11
W[2,0]	**W[2,1]**	**b[2]**
2.27 0.27	-2.04 0.14	-0.10 -0.21

Step size: 0.10000

Single parameter update

Start repeated update

Stop repeated update

Randomize parameters

x[0]	x[1]	y	s[0]	s[1]	s[2]	L
0.50	0.40	0	1.20	0.01	0.22	0.02
0.80	0.30	0	1.67	0.33	1.10	0.44
0.30	0.80	0	1.38	-0.80	-1.05	0.00
-0.40	0.30	1	-0.80	-0.20	-1.62	0.39
-0.30	0.70	1	-0.01	-0.88	-2.21	1.87
-0.70	0.20	1	-1.57	-0.15	-2.10	0.00
0.70	-0.40	2	0.43	1.55	2.31	0.25
0.50	-0.60	2	-0.28	1.83	2.26	0.57
-0.40	-0.50	2	-1.98	1.26	0.01	2.24
						mean:
						0.64

Total data loss: 0.64
Regularization loss: 1.92
Total loss: 2.57

L2 Regularization strength: 0.10000

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

Optimization

Optimization

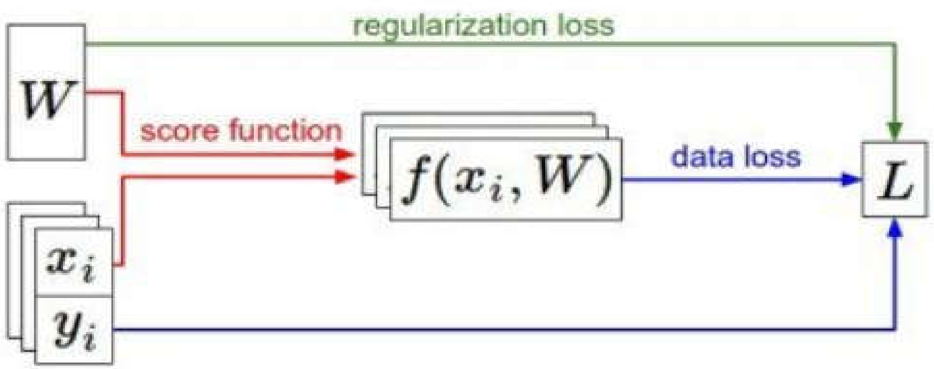
Recap

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

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

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

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



Slide from CS231n

Strategy #1: Random Search

- A first very bad idea solution: **Random search**

```
# assume X_train is the data where each column is an example (e.g. 3073 x 50,000)
# assume Y_train are the labels (e.g. 1D array of 50,000)
# assume the function L evaluates the loss function

bestloss = float("inf") # Python assigns the highest possible float value
for num in xrange(1000):
  W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
  loss = L(X_train, Y_train, W) # get the loss over the entire training set
  if loss < bestloss: # keep track of the best solution
    bestloss = loss
    bestW = W
  print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)

# prints:
# in attempt 0 the loss was 9.401632, best 9.401632
# in attempt 1 the loss was 8.959668, best 8.959668
# in attempt 2 the loss was 9.044034, best 8.959668
# in attempt 3 the loss was 9.278948, best 8.959668
# in attempt 4 the loss was 8.857370, best 8.857370
# in attempt 5 the loss was 8.943151, best 8.857370
# in attempt 6 the loss was 8.605604, best 8.605604
# ... (trunctated: continues for 1000 lines)
```

- Amazing process of optimization… - -)

Strategy #1: Random Search

- Lest see how well this works on the **test set**…

```
# Assume X_test is [3073 x 10000], Y_test [10000 x 1]
scores = Wbest.dot(Xte_cols) # 10 x 10000, the class scores for all test examples
# find the index with max score in each column (the predicted class)
Yte_predict = np.argmax(scores, axis = 0)
# and calculate accuracy (fraction of predictions that are correct)
np.mean(Yte_predict == Yte)
# returns 0.1555
```

10.0% random prediction

15.5% accuracy! Not bad!

(SOTA is **~95%**)

Loss Landscape

- W is a **2 dimensional vector** and loss is the **height**

Slide from CS231n

Loss Landscape

- W is a **2 dimensional vector** and loss is the **height**

Slide from CS231n

Strategy #2: Follow the Slope

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

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

- In multiple dimensions, the **gradient** is the vector of (**partial derivatives**).

Strategy #2: Follow the Slope

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

gradient dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,…]

Strategy #2: Follow the Slope

current W:	W + h (first dim):	gradient dW:
[0.34,	[0.34 + **0.0001**,	[?,
-1.11,	-1.11,	?,
0.78,	0.78,	?,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	?,
-3.1,	-3.1,	?,
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	loss 1.25322	

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

Slide from CS231n

Strategy #2: Follow the Slope

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25322

gradient dW:

[**-2.5**,
?,
?,
?,
?,…]

(1.25322 - 1.25347)/0.0001
= -2.5

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

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

Strategy #2: Follow the Slope

current W:	**W + h** (second dim)**:**	**gradient dW:**
[0.34,	[0.34,	[-2.5,
-1.11,	-1.11 **+ 0.0001**,	?,
0.78,	0.78,	?,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	?,
-3.1,	-3.1,	?,
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25353**	

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

Slide from CS231n

Strategy #2: Follow the Slope

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (second dim):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25353

gradient dW:

[-2.5,
0.6,
?,
?,

(1.25353 - 1.25347)/0.0001
= 0.6

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

?,…]

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

Slide from CS231n

Strategy #2: Follow the Slope

current W:	W + h (third dim):	gradient dW:
[0.34,	[0.34,	[-2.5,
-1.11,	-1.11,	0.6,
0.78,	0.78 + **0.0001**,	?,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	?,
-3.1,	-3.1,	?,
-1.5,	-1.5,	?,
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25347**	

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

Slide from CS231n

Strategy #2: Follow the Slope

current W:	W + h (third dim):	gradient dW:
[0.34,	[0.34,	[-2.5,
-1.11,	-1.11,	0.6,
0.78,	0.78 + **0.0001**,	**0**,
0.12,	0.12,	?,
0.55,	0.55,	?,
2.81,	2.81,	
-3.1,	-3.1,	
-1.5,	-1.5,	
0.33,…]	0.33,…]	?,…]
loss 1.25347	**loss 1.25347**	

(1.25347 - 1.25347)/0.0001
= 0

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

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

Numerical Gradient

- Evaluation the gradient numerically

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

- ConvNet has more than **100 millions parameters**…

Cons:

- Approximate (finite difference approximations)
- Very slow to evaluate

```
def eval_numerical_gradient(f, x):
  """
  a naive implementation of numerical gradient of f at x
  - f should be a function that takes a single argument
  - x is the point (numpy array) to evaluate the gradient at
  """

  fx = f(x) # evaluate function value at original point
  grad = np.zeros(x.shape)
  h = 0.00001

  # iterate over all indexes in x
  it = np.nditer(x, flags=['multi_index'], op_flags=['readwrite'])
  while not it.finished:

    # evaluate function at x+h
    ix = it.multi_index
    old_value = x[ix]
    x[ix] = old_value + h # increment by h
    fxh = f(x) # evalute f(x + h)
    x[ix] = old_value # restore to previous value (very important!)

    # compute the partial derivative
    grad[ix] = (fxh - fx) / h # the slope
    it.iternext() # step to next dimension

  return grad
```

Slide from CS231n

Calculus

- This is **silly**. The loss is just a function of W

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \sum_k \mathrm{W}_k^2$$

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

$$s = f\left(x; \mathrm{W}\right) = \mathrm{W}x$$

Calculus

Inventors of the Calculus

Want $\nabla_{\mathrm{W}} L$ …

Calculus

- This is silly. The loss is just a function of W

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

dW = ...
(some function
data and W)

gradient dW:

[-2.5,
0.6,
0,
0.2,
0.7,
-0.5,
1.1,
1.3,
-2.1,…]

- ➢ Evaluate entire vector **at once**
- ➢ Run this in **practice**

Slide from CS231n

In Summary:

- **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**.

Gradient Descent

```
# Vanilla Gradient Descent

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

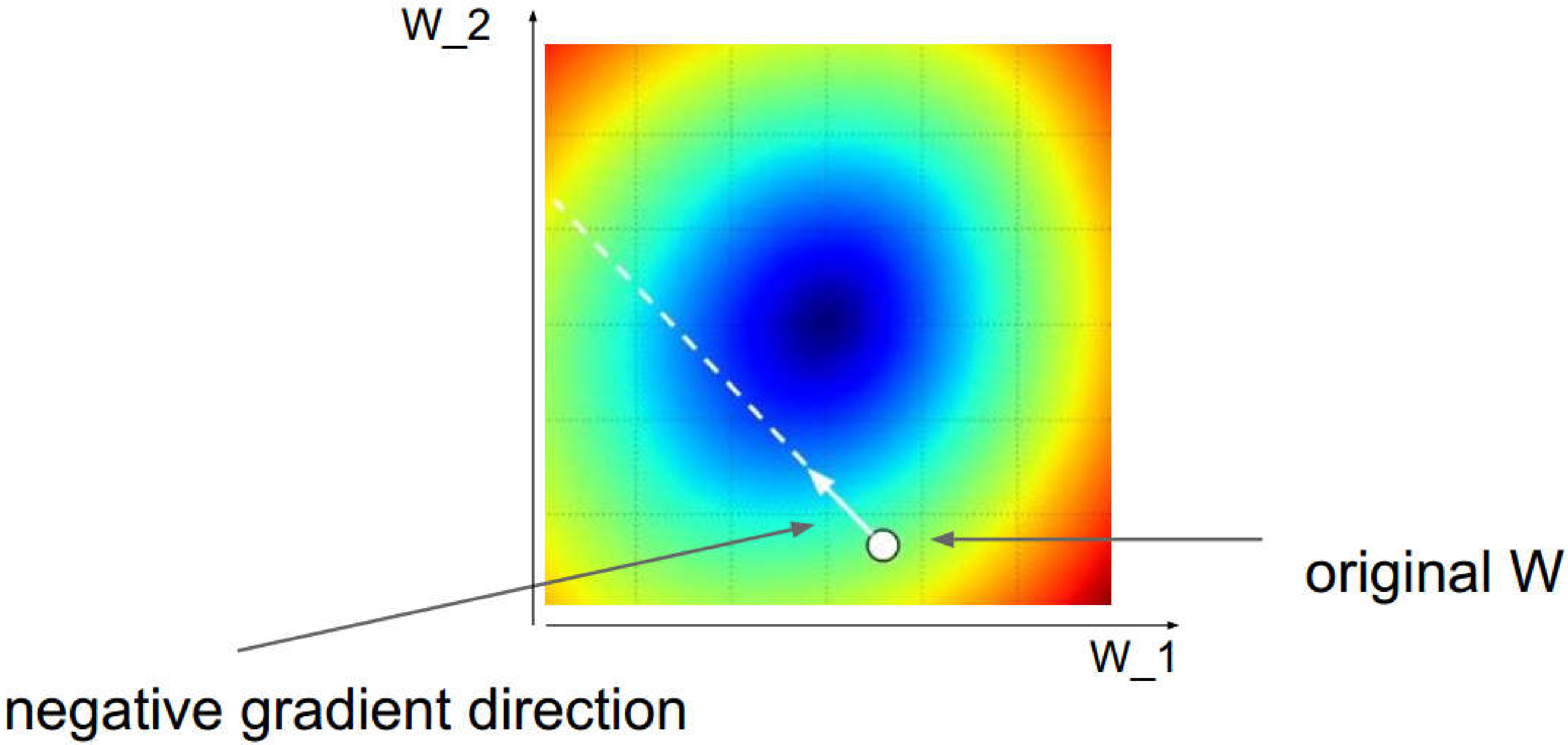
SGD: Stochastic Gradient Descent

NOTE

- **Step_size** is a **hyperparameter**, it's same with **learning rate**
- **The negative** before step_size is that gradient tells us the direction of the greatest increase of loss, and we want to minimize it where the negative is coming
- **Gradient** is the slope for every single direction

Slide from CS231n

Gradient Descent



Mini-Batch Gradient Descent

- Only use a **small portion of the training set** to compute the gradient.

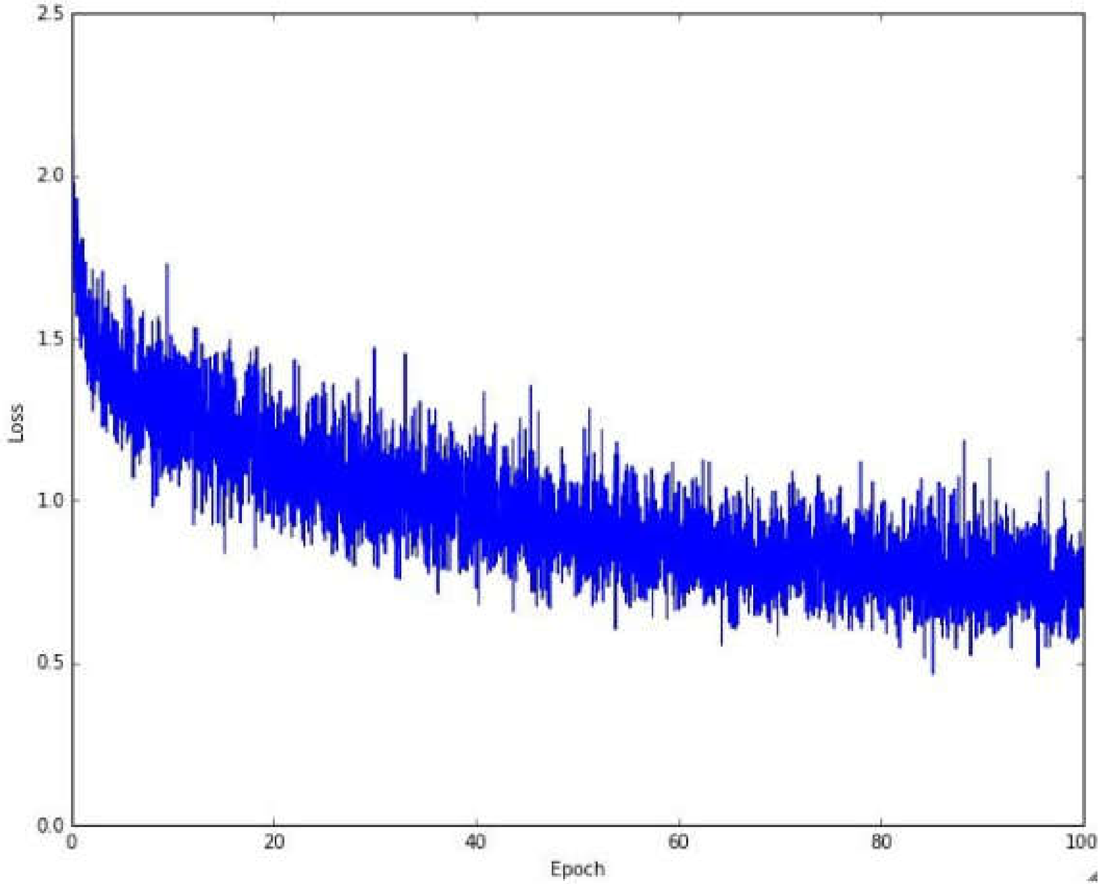
```
# Vanilla Minibatch Gradient Descent

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
```

- Common mini-batch sizes are **32/64/128** examples (<u>not hyperparameter</u>)

e.g. Krizhevsky ILSVRC ConvNet used 256 examples

Slide from CS231n

Optimization Progress

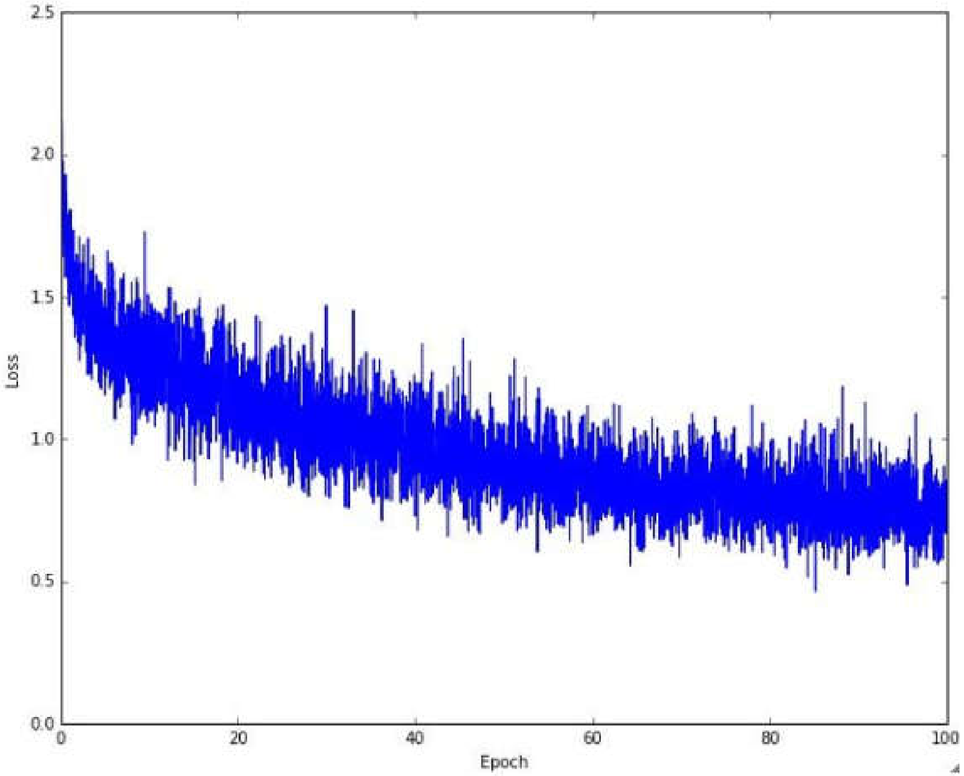


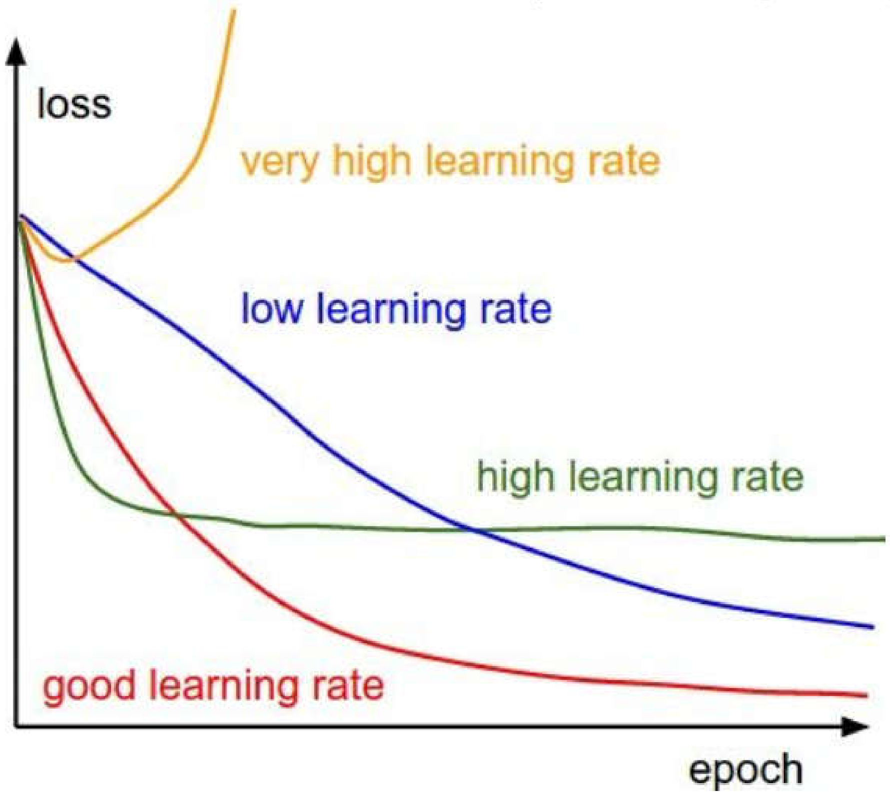
- Example of optimization progress while training a neural network.

(<u>Loss over mini-batches goes down over time</u>)

Optimization Progress

- The effects of **step size** (or "**learning rate**")





- Set a high learning rate
- Decay it over time

Slide from CS231n

Mini-Batch Gradient Descent

- Only use a **small portion of the training set** to compute the gradient.

```
# 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
```

SGD

We will look at more fancy update formulas

(**momentum**, **Adagrad**, **RMSProp**, **Adam**, …)

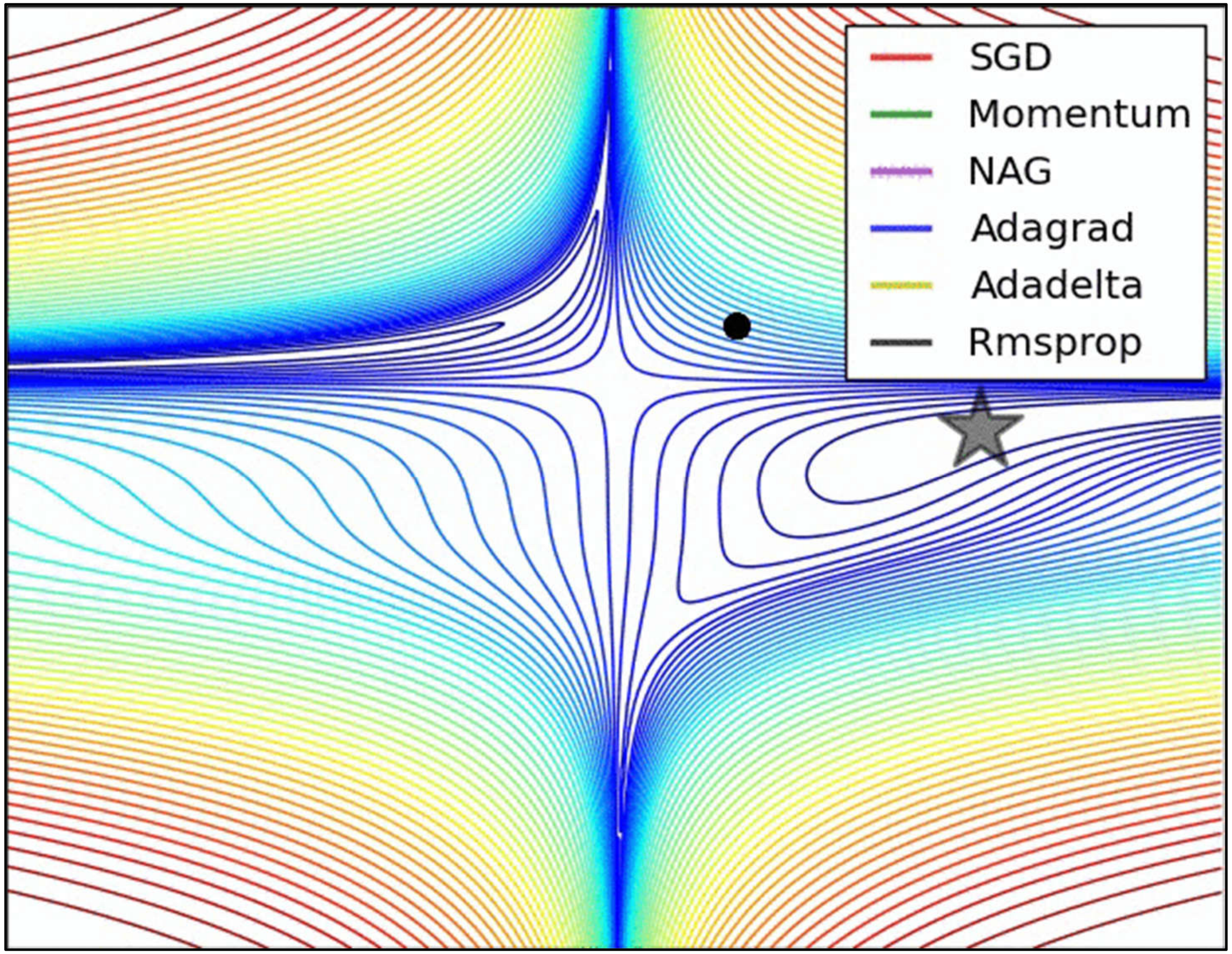
- Common mini-batch sizes are 32/64/128 examples (not hyperparameter)

e.g. Krizhevsky ILSVRC ConvNet used 256 examples

Slide from CS231n

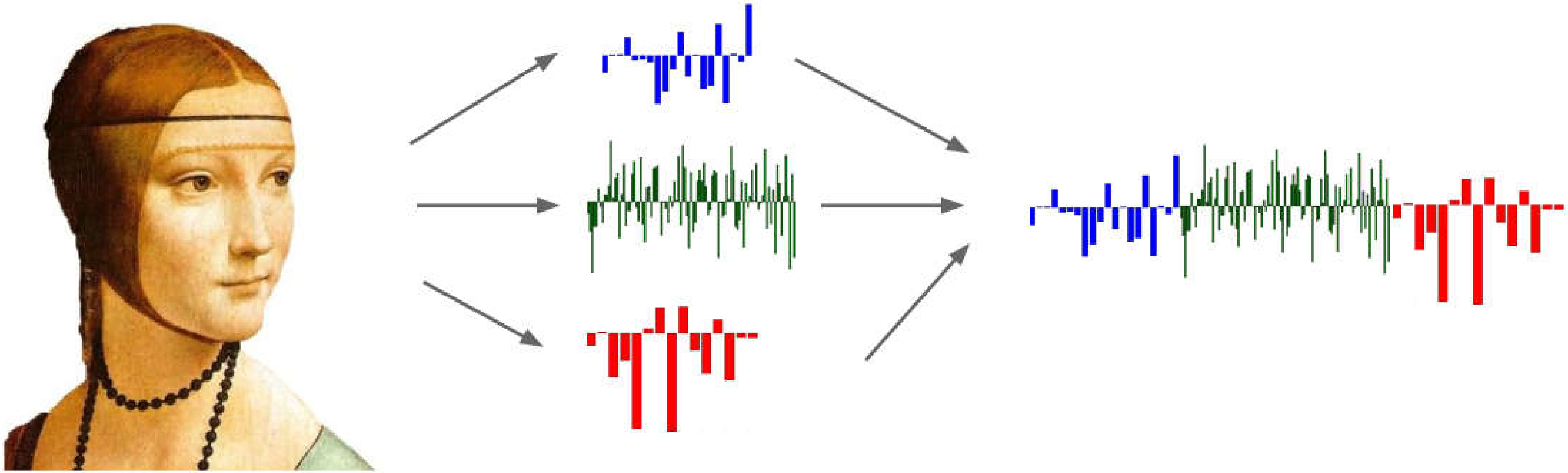
Different Update Form

- The effects of different update form formulas



(image credits to Alec Radford)

Aside: Image Features



Slide from CS231n

Assignment #1

March 28: Deadline of Assignment #1 (17:59:59) ~ 14 days left

Q1: k-Nearest Neighbor classifier (20 points)

The IPython Notebook **knn.ipynb** will walk you through implementing the kNN classifier.

Q2: Training a Support Vector Machine (25 points)

The IPython Notebook **svm.ipynb** will walk you through implementing the SVM classifier.

Q3: Implement a Softmax classifier (20 points)

The IPython Notebook **softmax.ipynb** will walk you through implementing the Softmax classifier.

Q4: Two-Layer Neural Network (25 points)

The IPython Notebook **two_layer_net.ipynb** will walk you through the implementation of a two-layer neural network classifier.

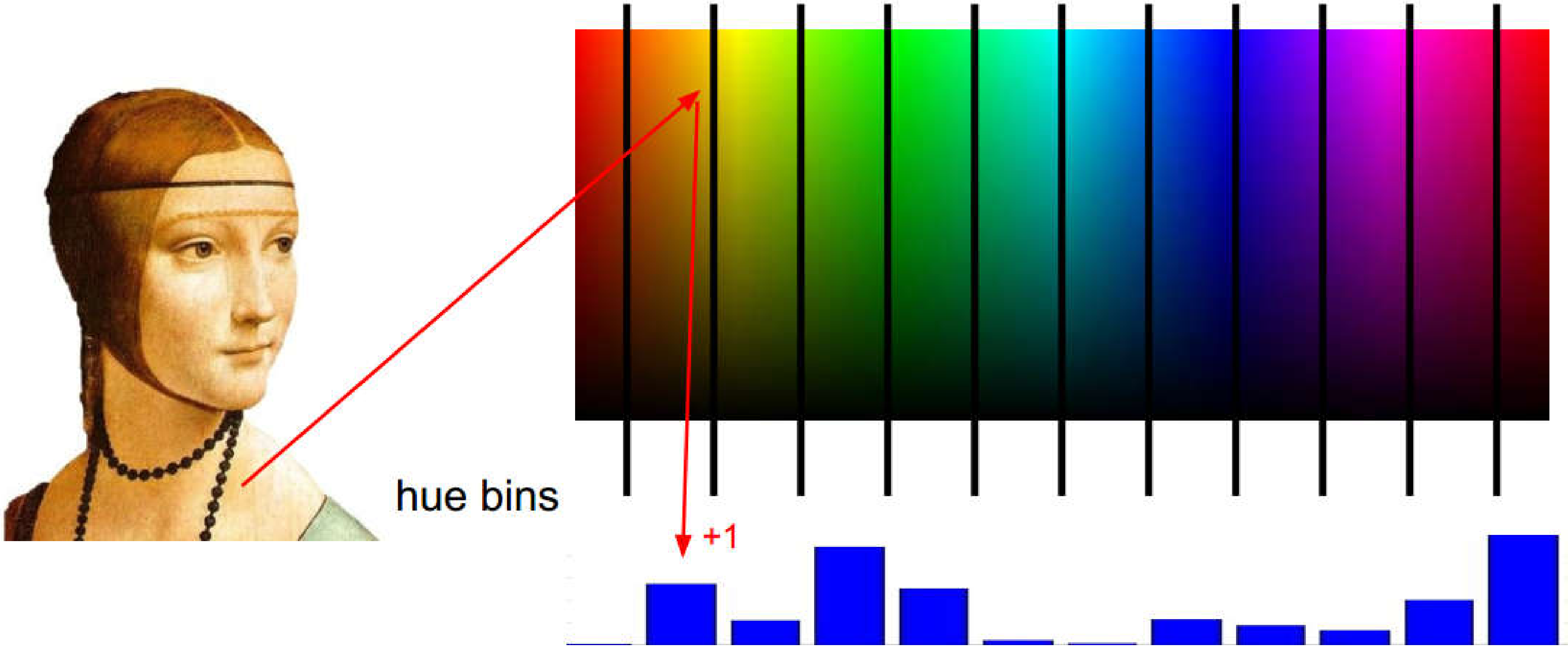
Q5: Higher Level Representations: Image Features (10 points)

The IPython Notebook **features.ipynb** will walk you through this exercise, in which you will examine the improvements gained by using higher-level representations as opposed to using raw pixel values.

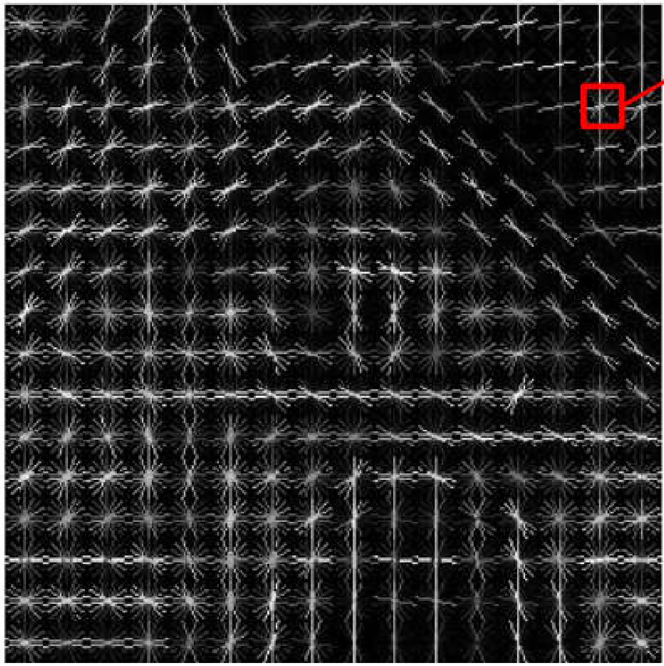
Q6: Cool Bonus: Do something extra! (+10 points)

Implement, investigate or analyze something extra surrounding the topics in this assignment, and using the code you developed. For example, is there some other interesting question we could have asked? Is there any insightful visualization you can plot? Or anything fun to look at? Or maybe you can experiment with a spin on the loss function? If you try out something cool we'll give you up to 10 extra points and may feature your results in the lecture.

Example: Color (Hue) Histogram



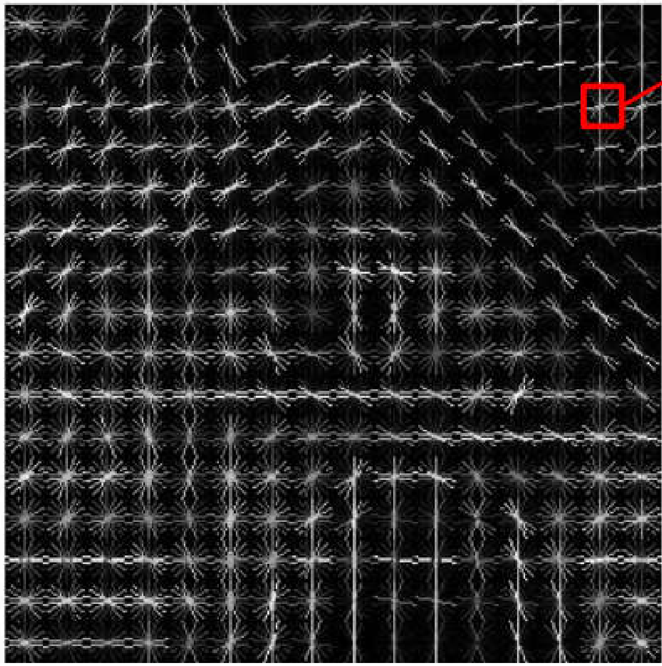
Example: HOG Features



8x8 pixel region, quantize the edge orientation into 9 bins

(image from vlfeat.org)

Example: HOG/SIFT Features



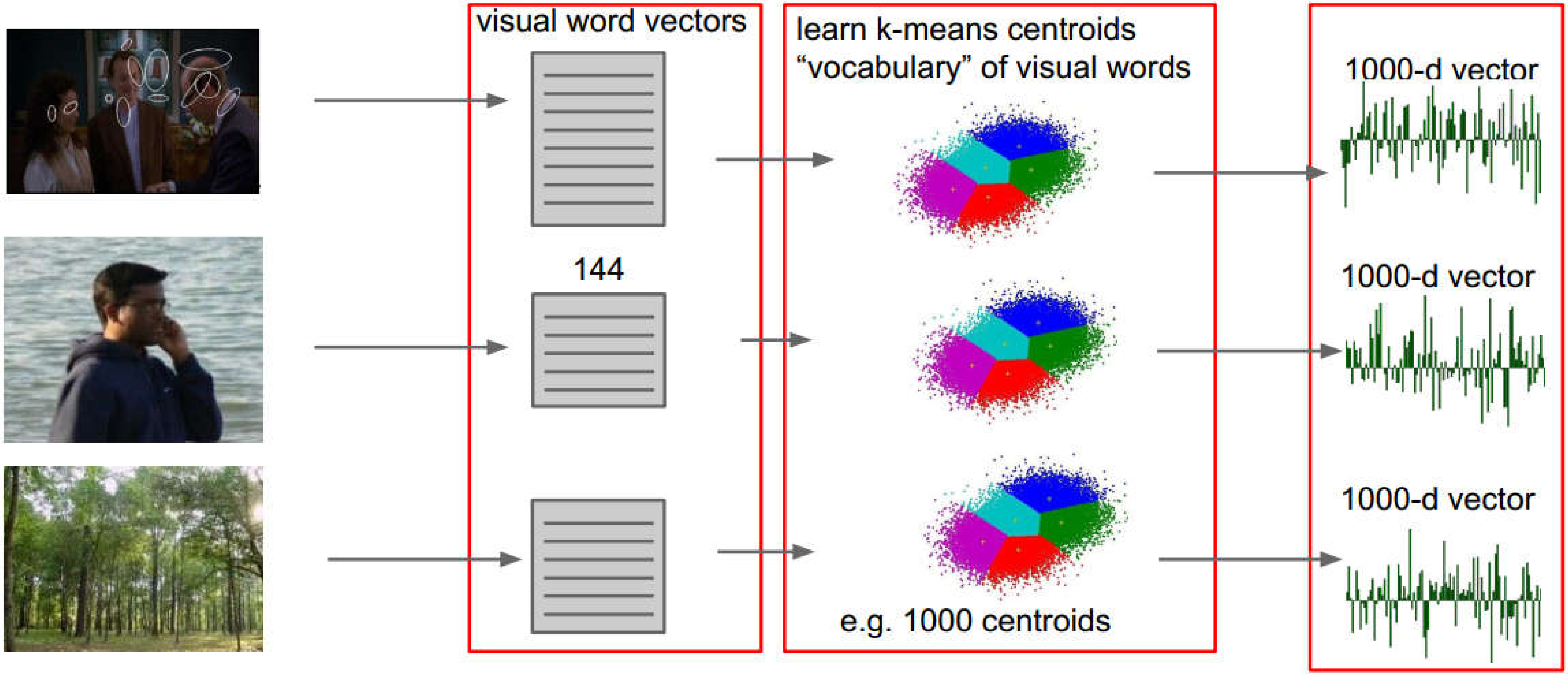
8x8 pixel region, quantize the edge orientation into 9 bins

Many more: GIST, LBP, Texton, SSIM, …

(image from vlfeat.org)

Example: Bag of Words

Histogram of visual words



Slide from CS231n

Historical Pipeline

Vector describing
various image statistics

[32x32x3]

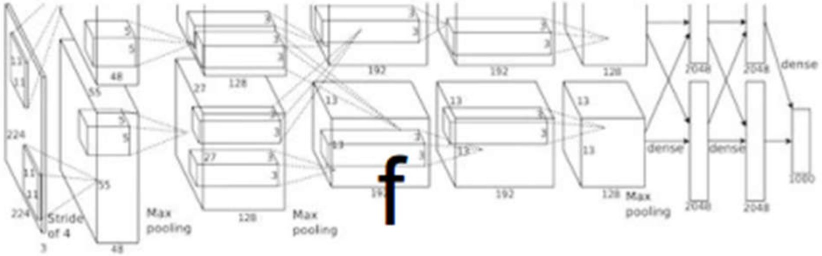
Feature
Extraction

f

training

10 numbers, indicating
class scores

[32x32x3]



f

training

10 numbers, indicating
class scores

Slide from CS231n

Next Class

Becoming a backpropagation ninja

and

Neural networks (part 1)

Q & A