이화여자대학교
EWHA WOMANS UNIVERSITY

LECTURE 3
Loss Functions and Optimization

2070070 정보경

EWHA,
THE FUTURE
WE CREATE

CONTENTS

1. Loss Function
2. Regularization
3. Optimization
4. Image Feature

1. Loss Function

✓ 정의

Classifier의 training set에 대한 성능을 평가

<Training set>

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

자동차는 잘 분류/ 고양이, 개구리에는 성능 ▼

→ **Loss function**으로 분류기 성능을 종합하여 나타냄

$\{(x_i, y_i)\}_i^N$ Dataset의 Loss function (x는 image, y는 label)

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

각 Data Loss function 값의 평균치

1. Loss Function

✓ Multiclass SVM loss

Loss function의 한 종류

옳은 Class와 아닌 Class의 **score를 비교**하여 Loss 값을 구하는 방식

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

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

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

y_i의 score값이 j의 score 값보다 상당히 크다
= 잘 분류되었다 (한 가지 Class로 특정된 것)
= Loss=0

y_i의 score값이 j의 score 보다 작거나 조금 크다
= 잘 분류되지 않은 편이다
= Loss= (j의 score)- (y_i의 score)+1

*'상당히', +1: Safety Margin. 'Score 차이가 크다 ' 의 기준 값

1. Loss Function

✓ Multiclass SVM loss

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

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

Loss over full dataset is average:

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

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

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

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

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

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

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

= 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

1. Loss Function

✓ Multiclass SVM loss

Q1. “잘 분류된” 항목의 score가 약간 변화할 때의 SVM LOSS값 변화는?

없다! 두 항목의 크기를 비교하는 function이므로 약간의 score 변화는 크기비교에 결과에 영향을 주지 않는다.

Q2. Min, Max 값

Min 항목이 잘 분류되었을 때 0

Max: infinity

Q3. W의 초기값이 매우 작아서 모든 score 도 0에 가까울 시 LOSS 값은?

Safety margin=1이므로 식에 의해 SVM LOSS = Class -1

1. Loss Function

✓ Multiclass SVM loss

Q4. Sum에서 옳은 Class의 score를 제외하지 않을 때의 계산값은?

Safety margin이 더해지므로 LOSS+1

Q5. Sum 대신 mean을 활용하였을 때 결과값 차이

없다! 단순 rescale이므로.

Q6. 해당 식을 사용할 시 LOSS 값은? $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$

기존 결과와 다르다! Error 정도가 제곱되어 더해지므로 더욱 극단적인 결과를 확인할 수 있다.

Error를 고려하는 수준에 따라 LOSS function을 다양하게 선택할 수 있다.

1. Loss Function

✓ Multiclass SVM loss

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

```python
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
```

1. Loss Function

이화여자대학교
EWHA WOMANS UNIVERSITY

✓ Softmax Classifier

Score 값 자체를 고려하는 Loss function

Score: Class들의 비정규화된 log 확률

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

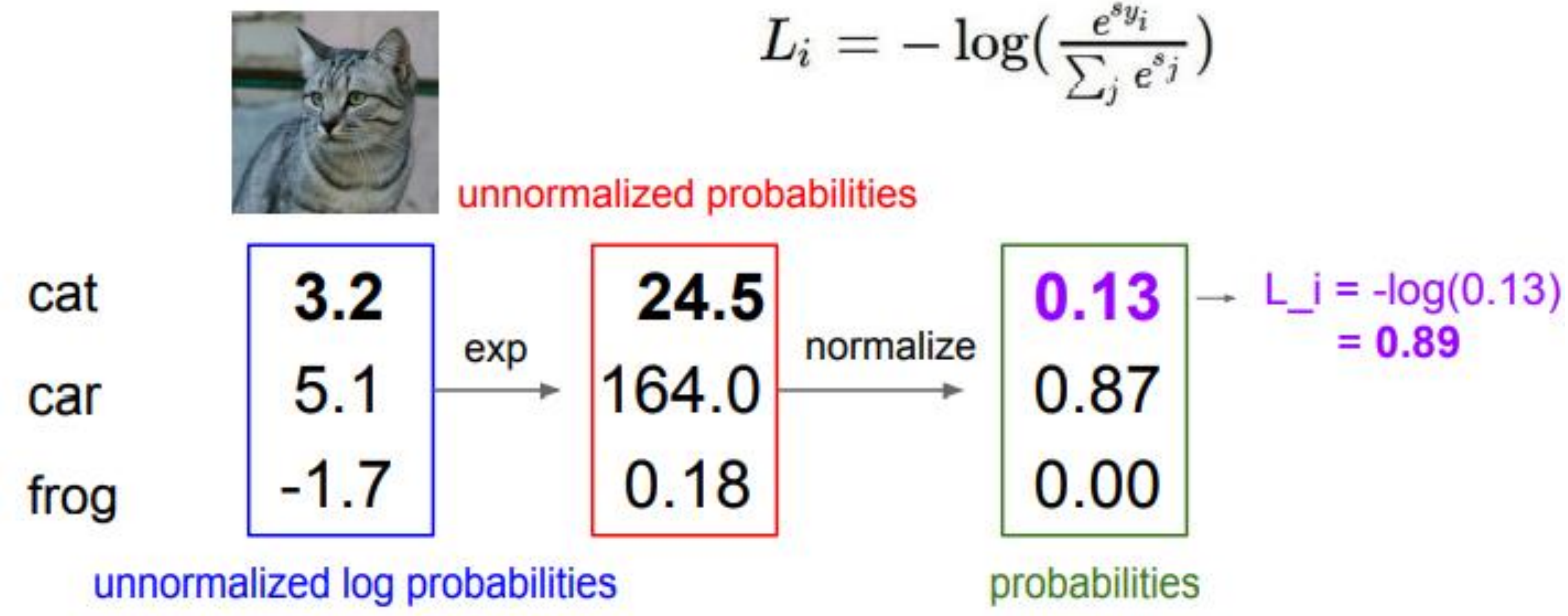
$$L_i = -\log P(Y=y_i|X=x_i)$$

확률값이 높을수록 확률의 -Log 값 (=Loss 값)이 최소화된다.

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

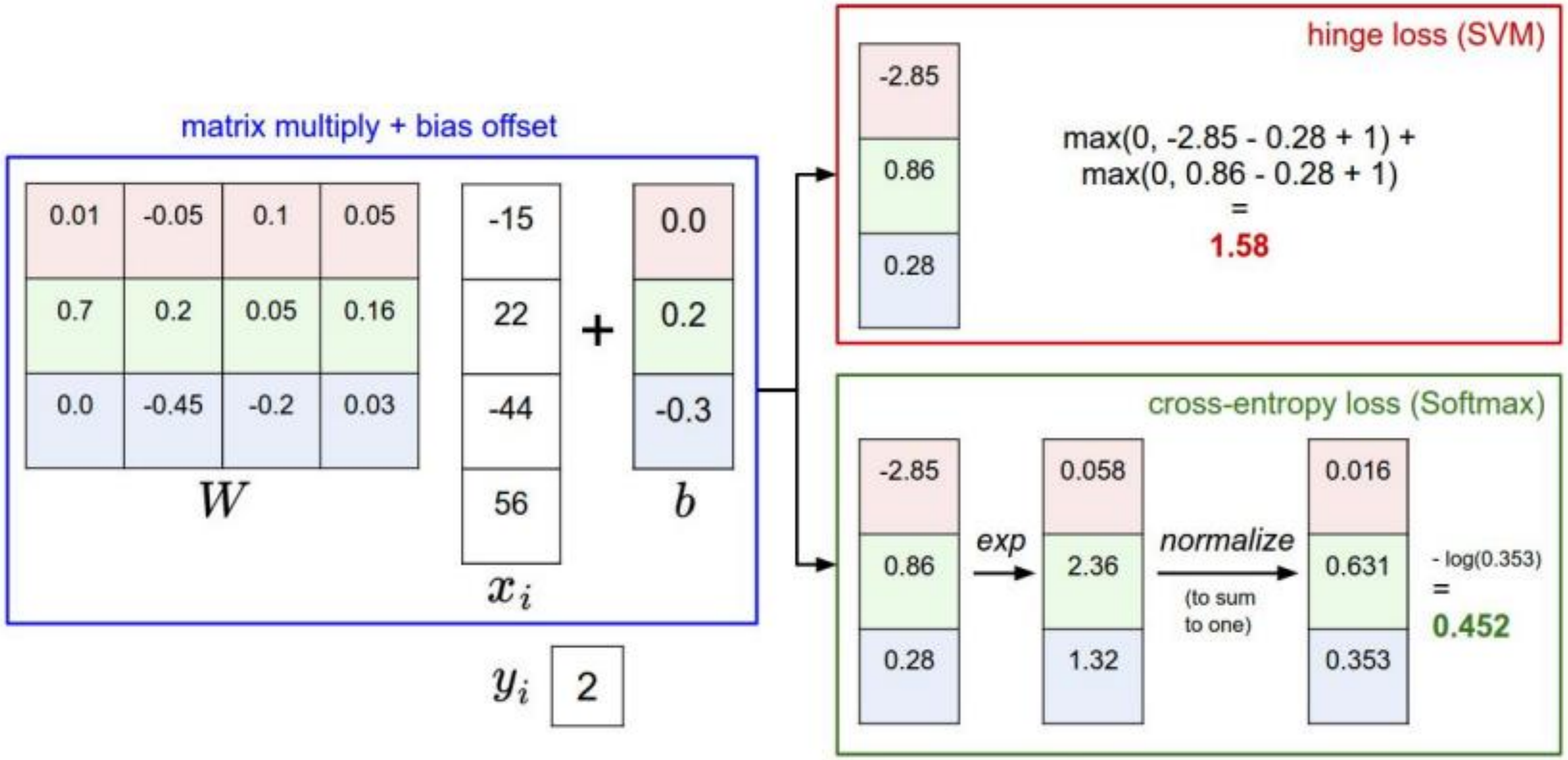
1. Loss Function

✓ Softmax Classifier



1. Loss Function

✓ Multiclass SVM loss VS Softmax



1. Loss Function

✓ Multiclass SVM loss VS Softmax

Q. 옳은 Class의 score를 약간 바꿨을 때의 계산값은?

SVM: 두 score의 비교 방식 -> 변화 없음

Softmax: 확률 계산 방식 -> 변화 발생

assume scores:
[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and $\boxed{y_i = 0}$

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

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

1. Loss Function

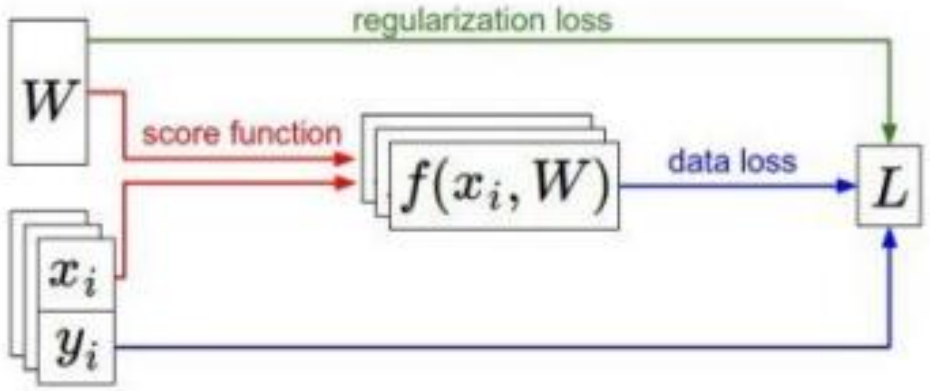
Softmax

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

SVM

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

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



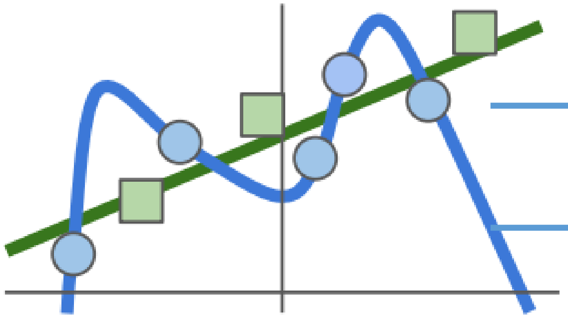
2. Regulation

✓ 정의

Loss의 최소화만 이용할 시, Classifier는 Training set에만 Fit 하게 됨

Regulation은 이를 해결하기 위한 방법 – Loss function에 W의 복잡도에 대한 값 추가

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Model should be “simple”, so it works on test data}}$$



Regulation 고려하여 Classified
: Test Data에 적용 가능

Data Loss만 고려하여 Classified
: Training Data에 fitted

2. Regulation

✓ 방법

W의 복잡도를 계산하는 방식

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

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

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

Max norm regularization (might see later)

Dropout (will see later)

Fancier: Batch normalization, stochastic depth

λ = regularization strength (hyperparameter)

Parameter의 설정이 결과값에 중요

2. Regulation

✓ **방법**

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

L2

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

W2가 더 작은 Regulation 값 가짐

L1

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

W1인 더 작은 Regulation 값 가짐

W의 복잡도를 계산하는 방법에 따라 Loss 값이 달라짐

3. Optimization

✓ 정의

가장 좋은 W를 찾기 위한 방법

=Loss값이 작은 W를 선택하는 방법

✓ 방법

1. Random search

W를 무작위적으로 생성해 가장 좋은 W를 선택한다.

계산량이 많고 시간이 오래 걸림

3. Optimization

✓ 방법

2. Follow the Slope

Loss 값이 적어지는 방향-> 정확도가 높은 방향

따라서 Loss가 감소하는 방향을 따라가면 됨

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

Gradient

1차원의 Slope = derivation

고차원의 Slope = gradient

3. Optimization

✓ 방법

2. Follow the Slope

계산법

a. Numerical Gradient

근사값으로 계산하는 방법, 손계산 쉬움, 느림, **디버깅에 사용**

b. Analytic Gradient

분석적으로 계산하는 방법. 정확, 빠름, 실제로 사용

3. Optimization

✓ 방법

2. Follow the Slope – Numerical Caculate

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

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

3. Optimization

✓ 방법

2. Follow the Slope – Analytic Calculate

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

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

$$s = f(x; W) = Wx$$

want $\nabla_W L$

= loss 함수와 W gradient 의 내적
= Loss가 작아지는 W의 방향을 알 수 있다.

3. Optimization

이화여자대학교
EWHA WOMANS UNIVERSITY

✓ 학습

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

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

loss 감소방향으로
Step size만큼의 가중치를 부여

```
# Vanilla Gradient Descent

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

*** Step size : 학습 정도를 결정하는 중요 Hyperparameter**

3. Optimization

이화여자대학교
EWHA WOMANS UNIVERSITY

✓ 학습

실제 Data 값이 매우 많으므로 W gradient를 일일이 계산하는 것에 한계가 있음.

Minibatch 사용
2의 제곱수의 example 추출해 W와 W gradient의 추정치 계산

```
# 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
```

3. Optimization

이화여자대학교
EWHA WOMANS UNIVERSITY

✓ 학습

실제 Data 값이 매우 많으므로 W gradient를 일일이 계산하는 것에 한계가 있음.

Minibatch 사용
2의 제곱수의 example 추출해 W와 W gradient의 추정치 계산

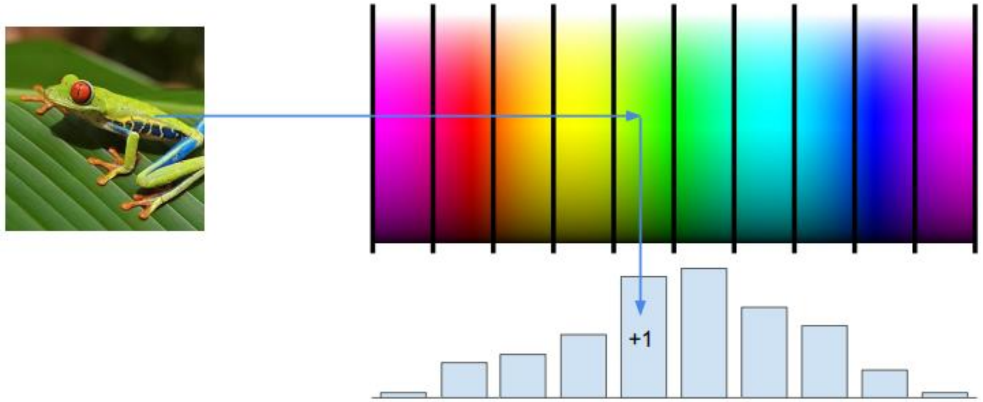
```
# Vanilla Minibatch Gradient Descent

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
```

4. Image Feature

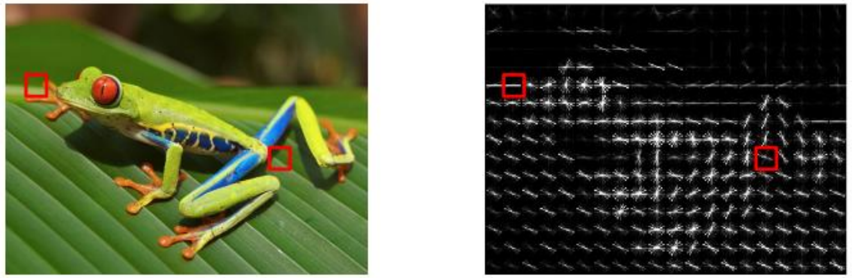
✓ 동기

이미지의 특징을 추출하고 재배열하여 분류를 쉽게 만듦.

Example: Color Histogram



Example: Histogram of Oriented Gradients (HoG)



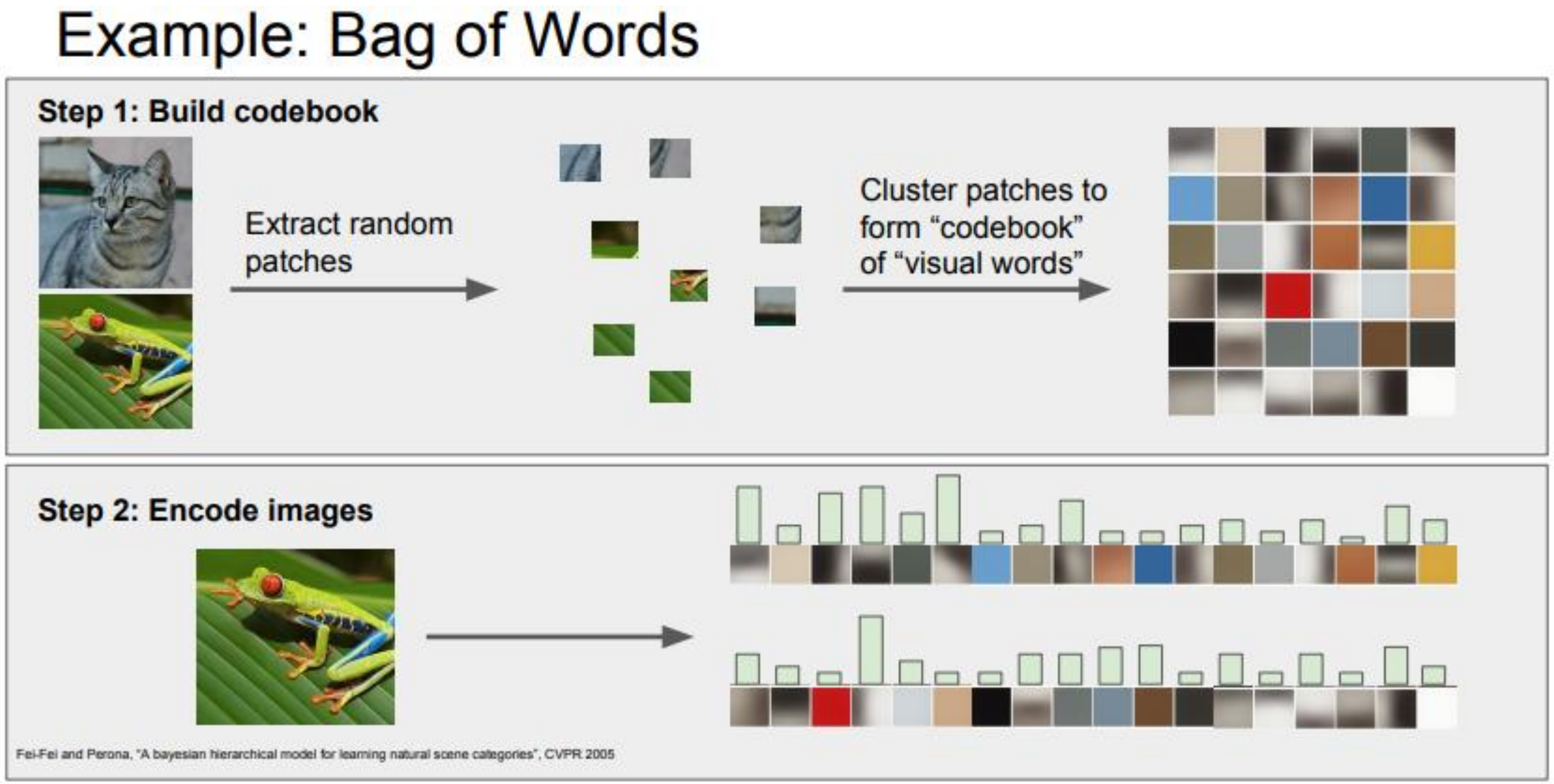
Divide image into 8x8 pixel regions
Within each region quantize edge
direction into 9 bins

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

픽셀의 색을 추출, 히스토그램화

픽셀의 Oriented gradients 추출, Edge feature

4. Image Feature



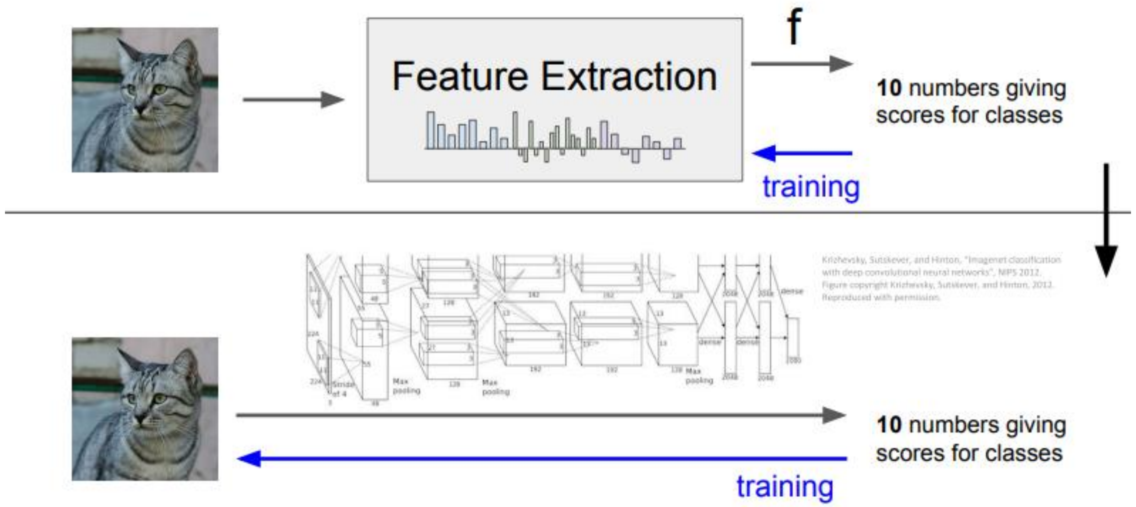
이미지 조각을 (Patch) 무작위적으로 추출하여 클러스터링

4. Image Feature

✓ Image feature VS ConvNets

Image feature
이미지의 특징을 추출 후 input -> 결과로 Training

ConvNets
Feature extraction과 Function이 별도로 나누어져 있지 않음



감사합니다.