Linear Classifiers

발제자: 9기 박서연

Linear Classifier

: 신경망(Neural Network)을 구성하는 가장 기본적인 요소

레고 작품 전체
: Neural Network

블록 하나하나
: Linear Classifier

Parametric Approach

: 모든 데이터를 저장하는 것이 아니라 파라미터 값만 저장

Non Parametric Approach

ex) kNN
(k-nearest neighbors) 알고리즘

모든 데이터를 저장

Parametric Approach

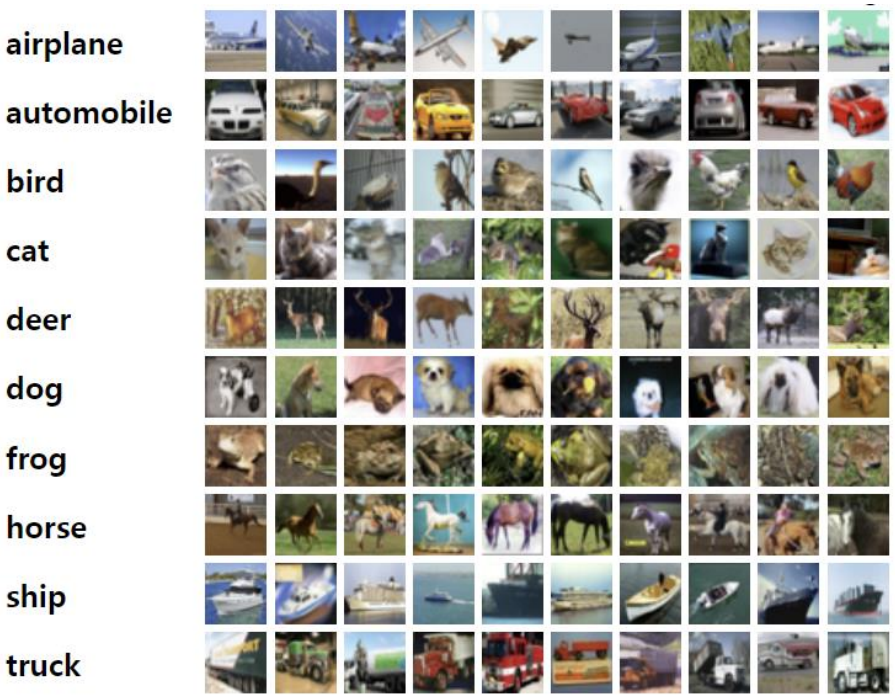
ex) Linear Classifier

Parametric Approach를 적용한 모델의
가장 단순한 형태가 바로 오늘 공부할
Linear Classifier!

Parametric Approach: Linear Classifier

설명에 이용할 데이터셋: **CIFAR-10 dataset**

https://www.cs.toronto.edu/~kriz/cifar.html



32*32 pixels

3 channels
: red, blue, green

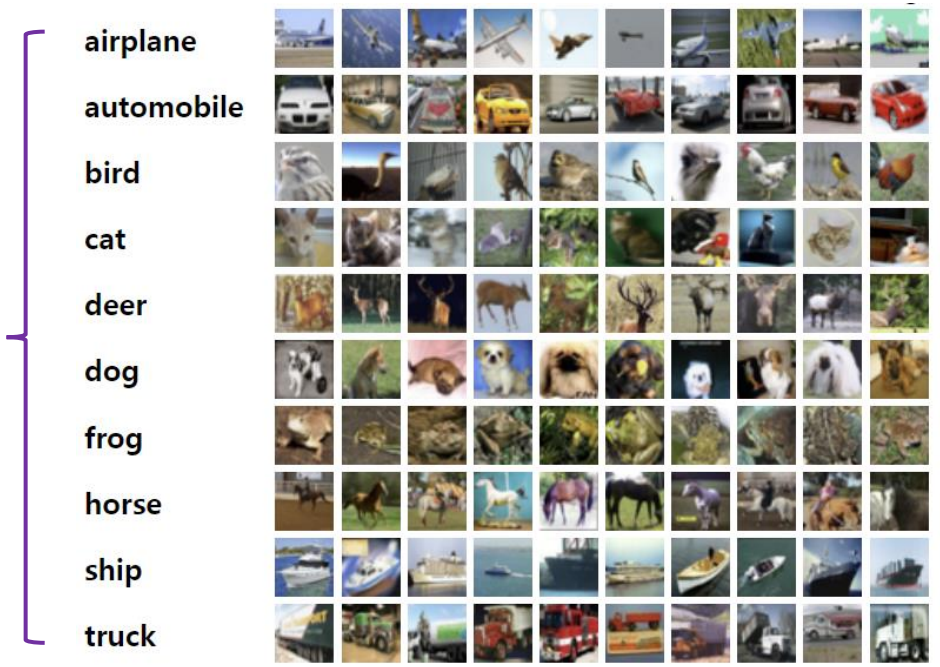
-> 32*32*3 픽셀

Parametric Approach: Linear Classifier

우리의 목표

입력 이미지를 받았을 때 아래 10개의 클래스(airplane, automobile, bird, cat…) **중 알맞은 클래스로 분류**할 수 있도록 해보자!

-> 10개의 클래스 각각에 해당하는 점수를 출력하도록 했을 때, 가장 높은 점수를 받은 클래스가 입력 이미지의 카테고리가 됨!



Parametric Approach: Linear Classifier

1) 원래 이미지

32*32*3 = 3072
-> 입력된 이미지의 픽셀을 구성

2) Flatten

이미지의 기존 구조를 파괴한 뒤 열벡터로 재구성

입력 이미지 (32, 32, 3) → **하나의 열벡터 (3072, 1)**

flatten

Parametric Approach: Linear Classifier

3) 함수

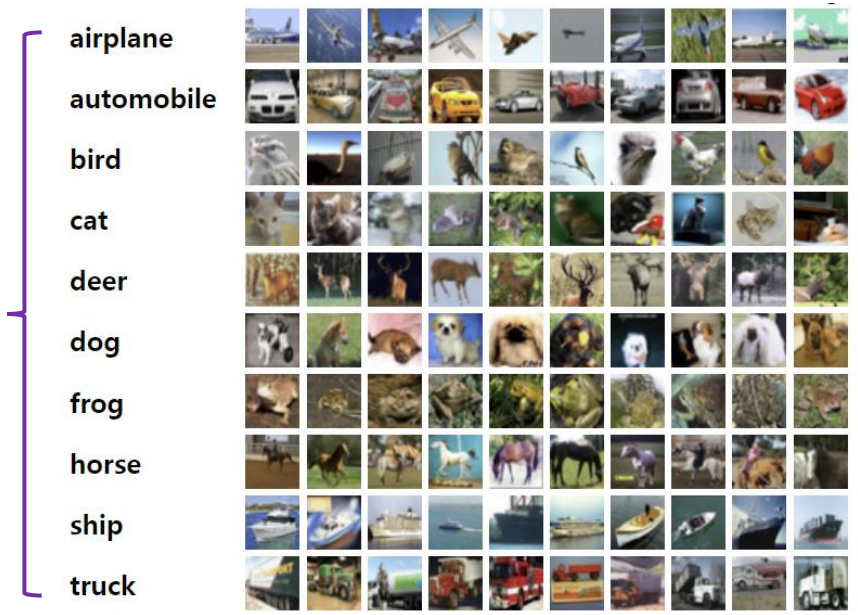
입력: 이미지
출력: 10개 클래스에 대한 점수

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

우리의 목표

입력 이미지를 받았을 때 10개의 클래스 중 알맞은 클래스로 분류할 수 있도록 해보자!

-> 10개 클래스 각각에 해당하는 점수를
결과값으로 뽑고 싶다!
함수에 들어가는 파라미터들의 shape은 어떻게 될까?



Parametric Approach: Linear Classifier

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

bias!
(10, 1)

입력 이미지의
재구성된 픽셀
(3072, 1)

가중치
(10, 3072)

크기 10의 벡터로 재탄생
(10, 1)

입력 이미지의 픽셀

분류하고자 하는 클래스의 수

W(가중치)

결과값으로 10개 클래스에 해당하는 점수가 나와야 하므로
(10, 3072)의 가중치를 곱해줌으로써 결과값을 구함

Parametric Approach: Linear Classifier

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

bias!
(10, 1)

입력 이미지의
재구성된 픽셀
(3072, 1)

가중치
(10, 3072)

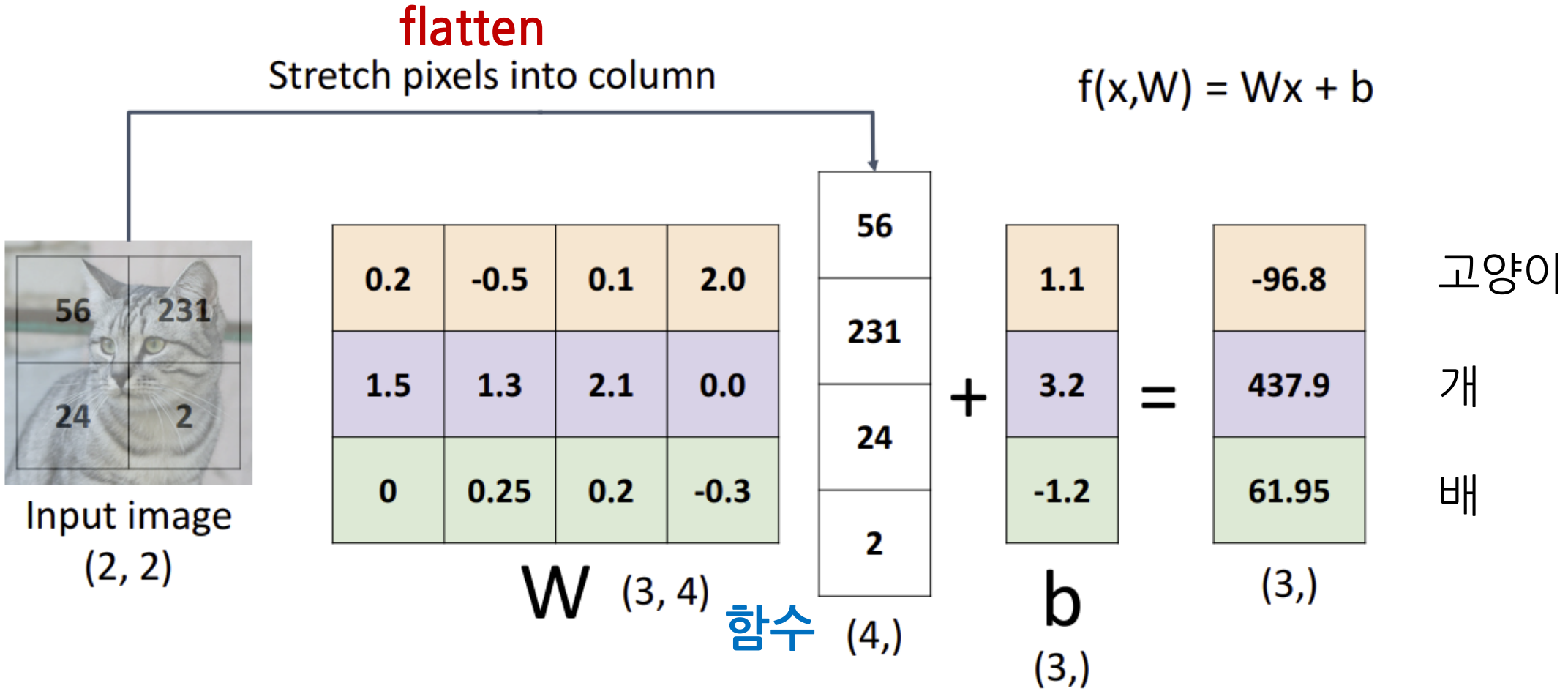
크기 10의 벡터로 재탄생
(10, 1)

b(bias)

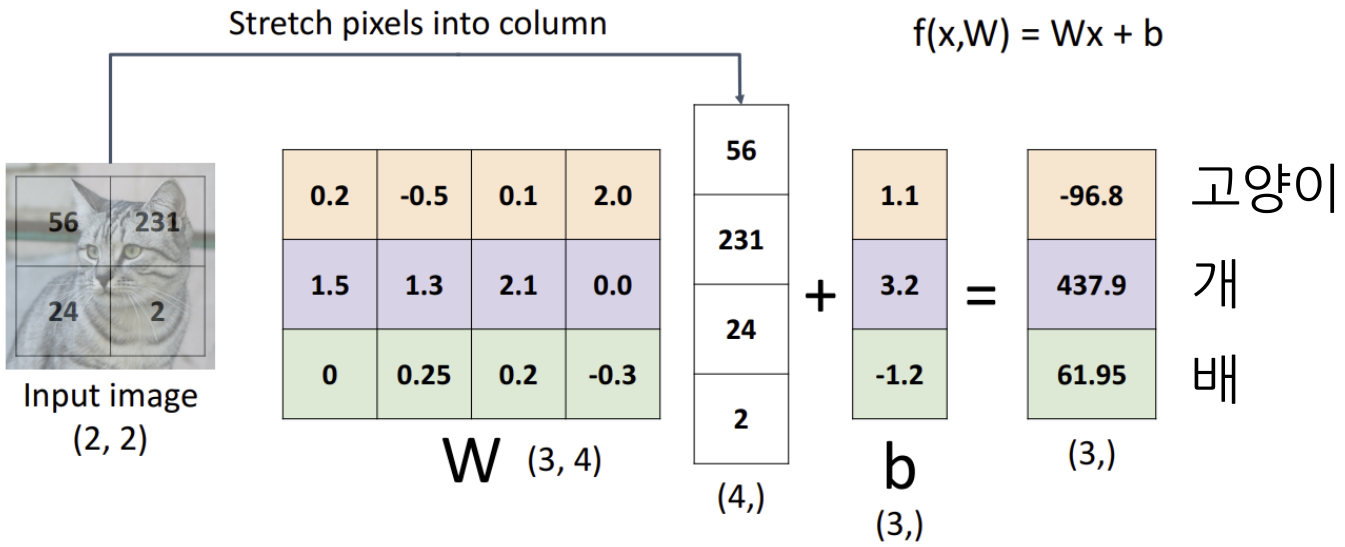
만일 데이터셋이 unbalance하다면(ex. 특정 클래스에 편중되어 있는 등)
기존 데이터셋과 무관하게 특정 클래스에 우선권을 부여하는 상수

Linear Classifier: Algebraic Viewpoint

: 입력 이미지 2*2를 3개의 class(고양이, 개, 배)로 분류하는 상황 가정



Linear Classifier: Algebraic Viewpoint



첫 번째 행

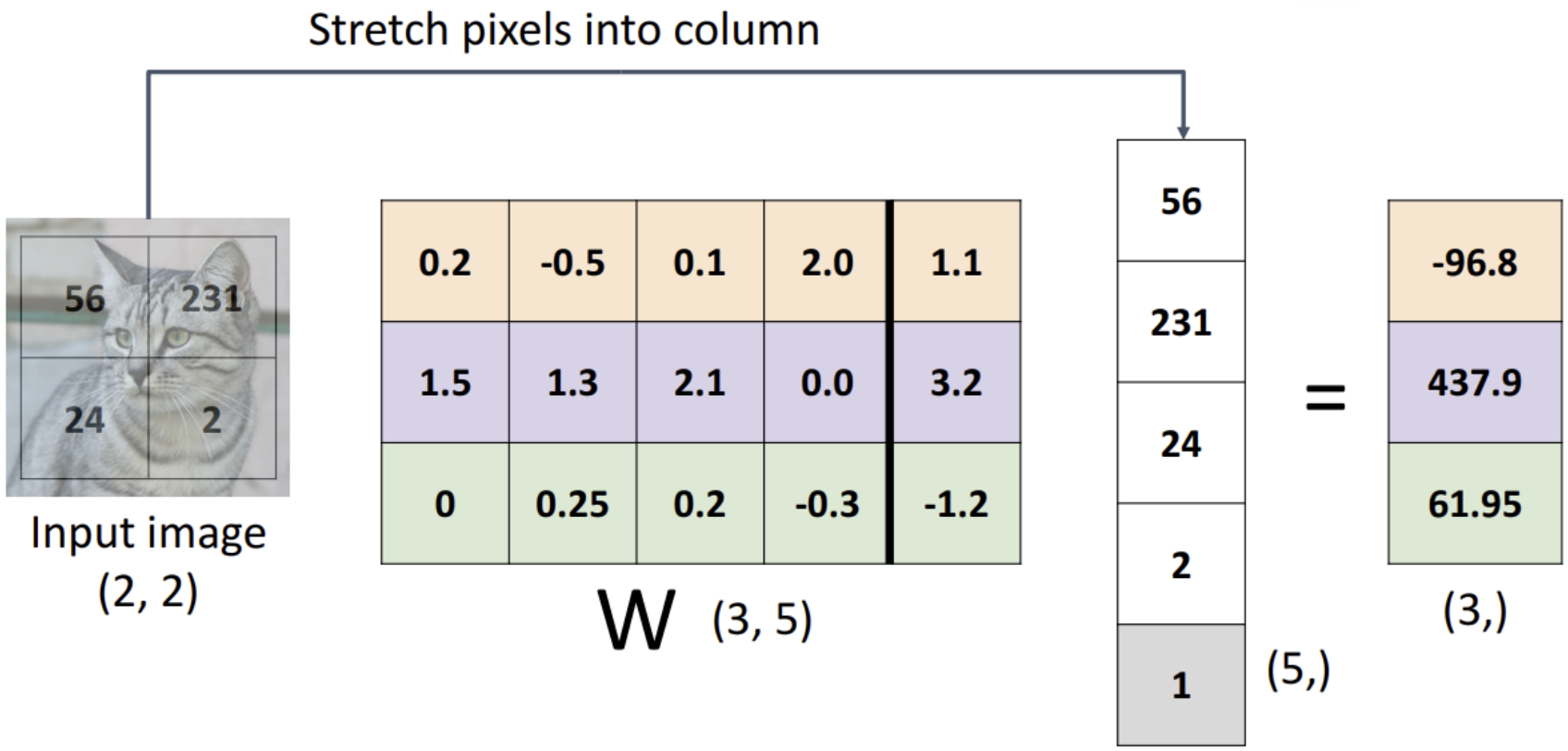
Wx = 0.2*56 + (-0.5)*231 + 0.1*24 + 2.0*2 = -97.9
b = 1.1

Wx+b = -96.8

Linear Classifier: Algebraic Viewpoint

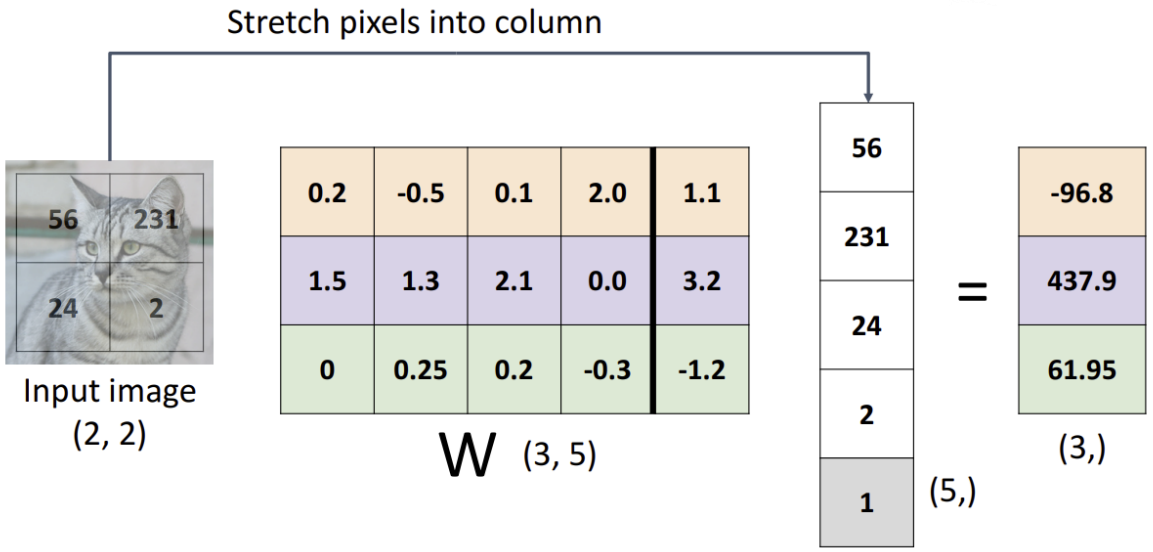
Bias Trick

: 앞서와 똑같은 상황인데 bias를 따로 떨어진 상수항으로 취급하는 것이 아니라 가중치 행렬에 포함시켜 버리자!



Linear Classifier: Algebraic Viewpoint

Bias Trick에서의 벡터 연산은 아까와 같다



첫 번째 행

Wx = 0.2*56 + (-0.5)*231 + 0.1*24 + 2.0*2 + 1.1*1 = -96.8
(이때 W는 b를 포함)

Linear Classifier: Algebraic Viewpoint

Bias Trick의 장단점

장점

- bias항을 없앰으로써 구조를 비교적 단순하게 만들어서 Algebraic(대수적) 관점에서 Linear Classifier를 이해하는 도움이 됨.

단점

- Computer vision에서는 bias trick을 잘 사용하지 않는 편. Linear classifiers to convolutions 파트로 넘어갈 때 매끄럽지 않음.

- 파라미터들을 초기화 혹은 정규화할 때 가중치와 bias를 따로 보는 것이 유리할 때가 있음.

Linear Classifier: Algebraic Viewpoint

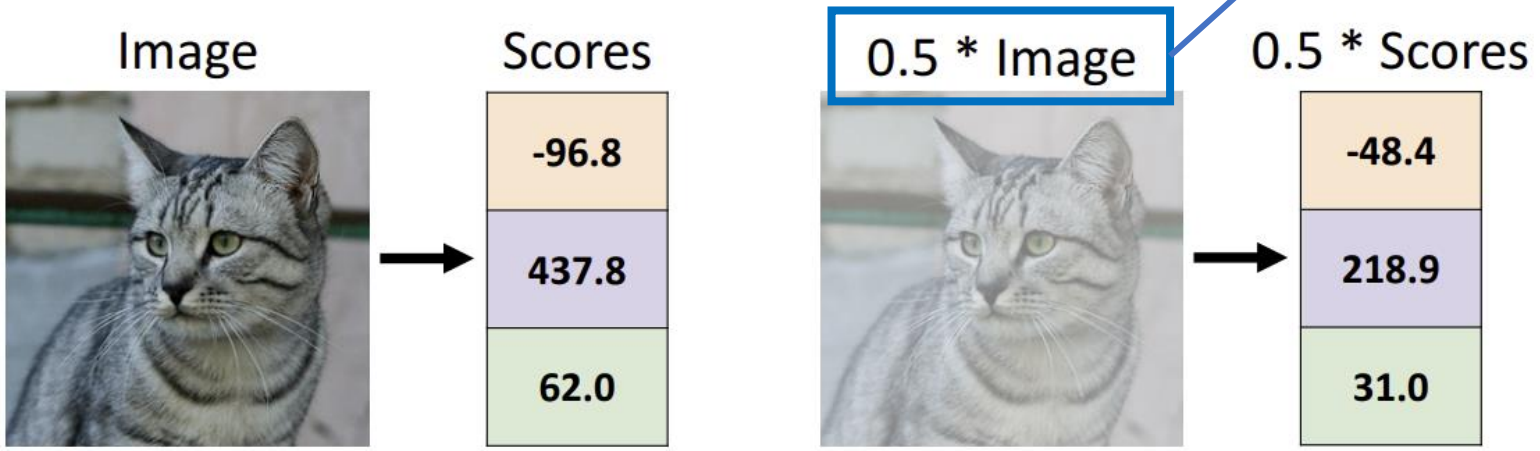
Predictions are Linear!

: 입력 이미지에 상수곱의 변화가 생긴다면 이것이 결과에 동일하게 반영된다.
bias항이 없다고 가정해보자.

$$f(x, W) = Wx \quad \text{(ignore bias)}$$

$$f(cx, W) = W(cx) = c * f(x, W)$$

원래의 입력 이미지에 0.5를 곱해 채도를 낮춘 상황이라고 생각



Linear Classifier: Visual Viewpoint

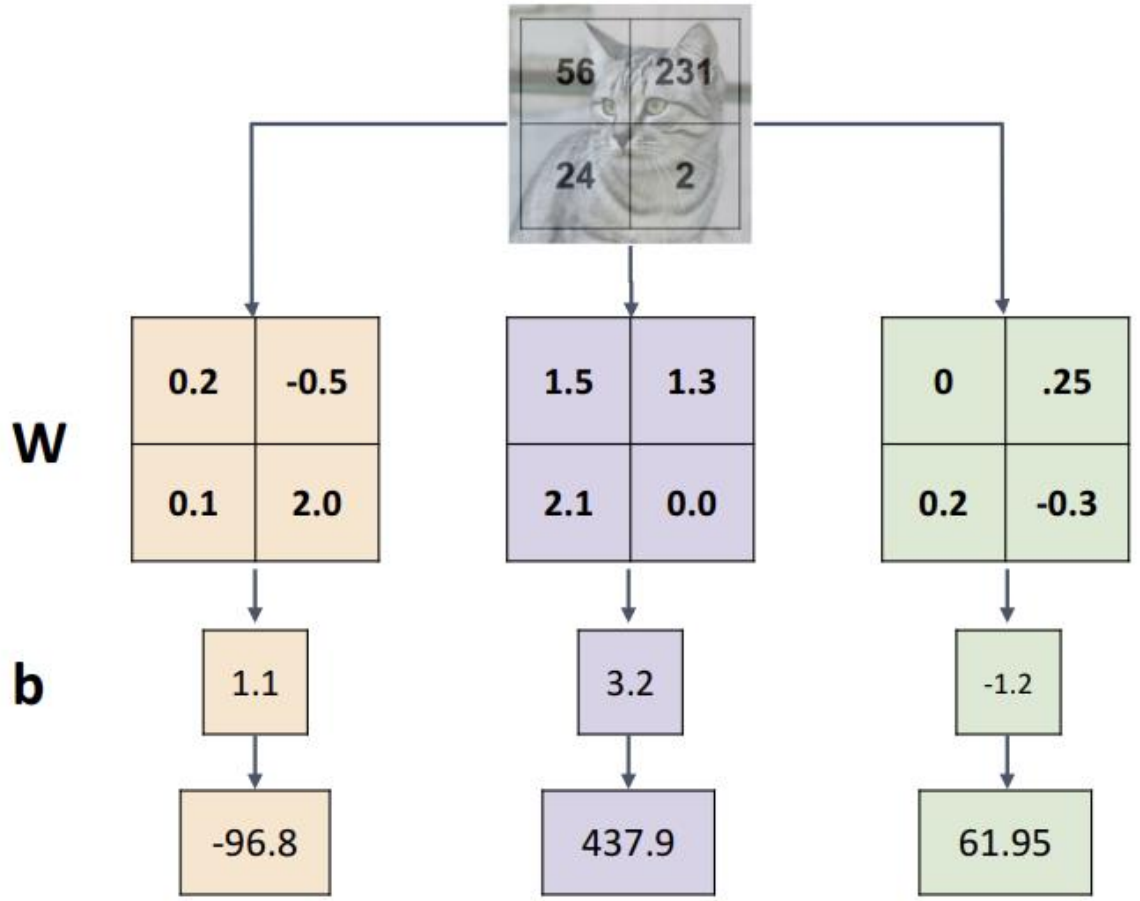
Algebraic Viewpoint

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

Visual Viewpoint

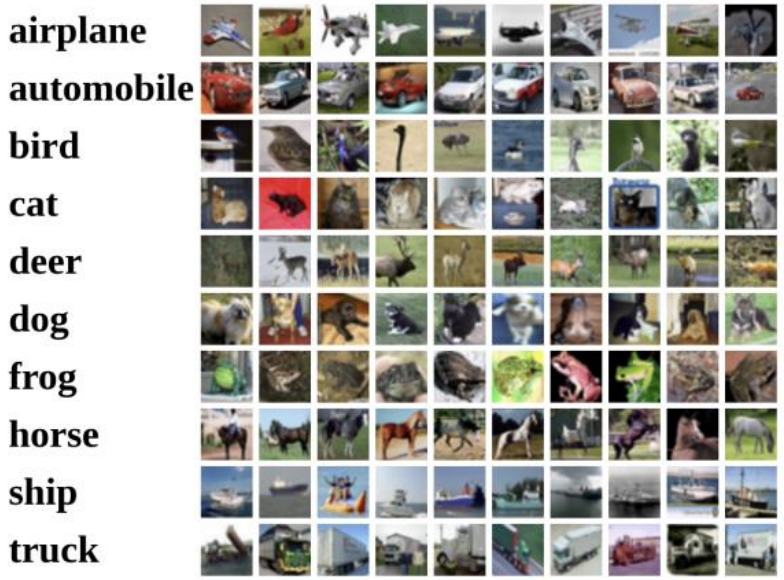
Algebraic Viewpoint와 달리,

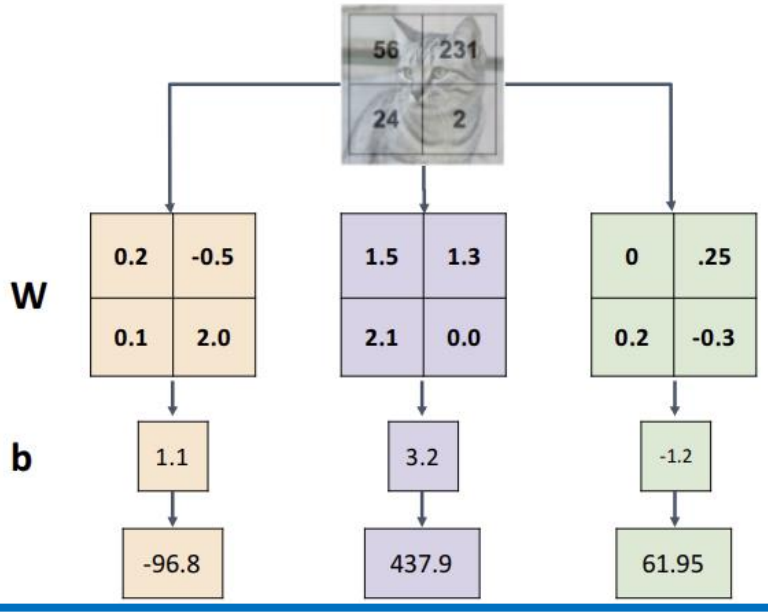
가중치 행렬의 각 행이 이미지와 동일한 shape을 갖도록 함!



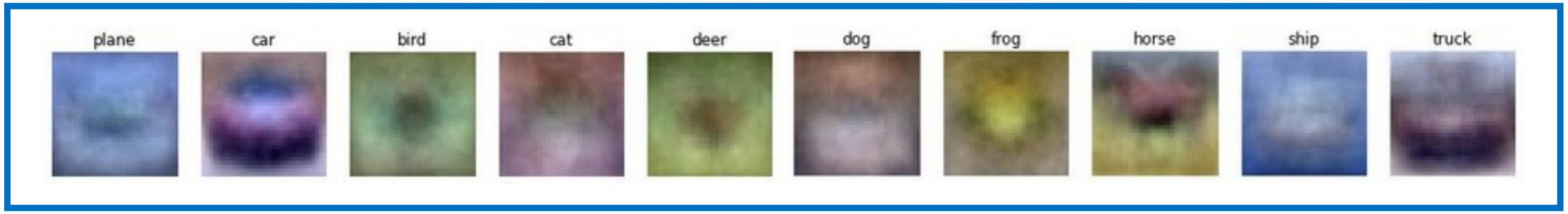
Linear Classifier: Visual Viewpoint

가중치 행렬의 각 행이 이미지와 동일한 shape를 갖기 때문에 가중치 각각에 대한 시각화를 할 수 있고, 그 결과 클래스 당 하나씩 총 10개의 '템플릿' 생성



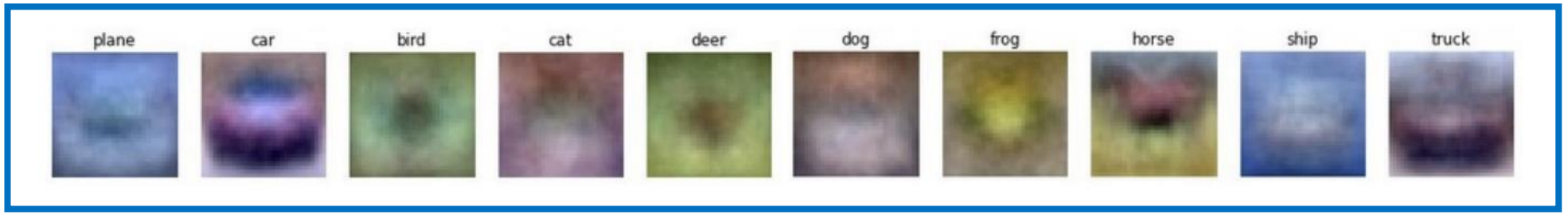


템플릿:



Linear Classifier: Visual Viewpoint

템플릿:



우리가 구분하고자 하는 카테고리 당 하나의 템플릿을 학습!

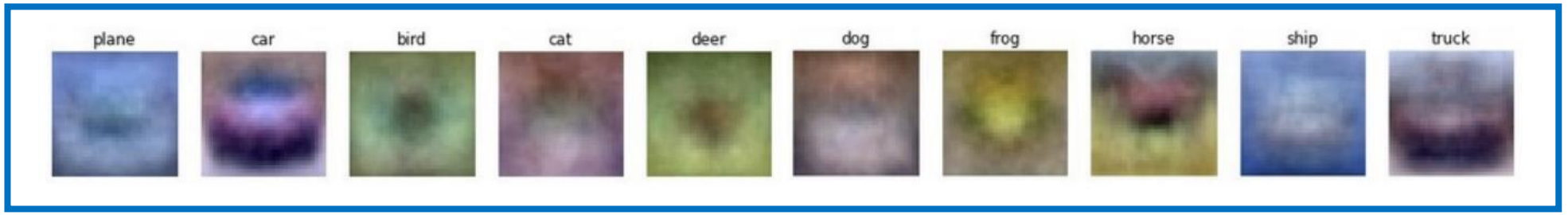
Algebraic Viewpoint에서 보았던 것처럼 벡터 간 내적 연산을 수행하여,
입력된 이미지가 각 템플릿과 얼마나 잘 맞는지를 계산

템플릿을 보면 Linear Classifier가 클래스 분류를 위해 유심히 보는 부분이 무엇인지를 알 수 있다

Linear Classifier: Visual Viewpoint

Visual Viewpoint에서의 Linear Classifier 실패 요인 1.

템플릿:



ex) 비행기

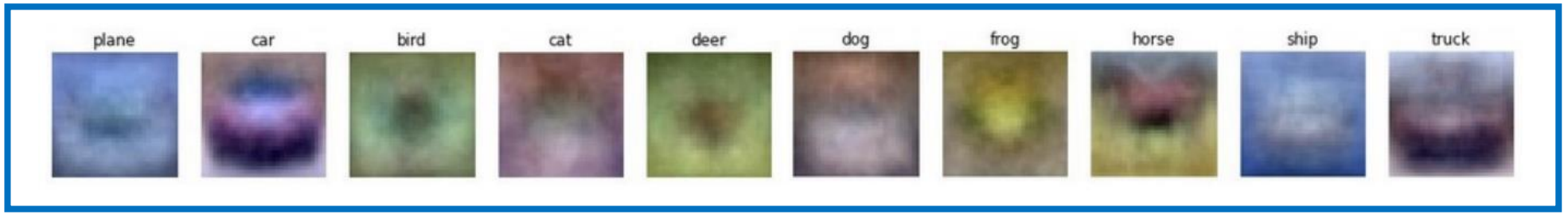
가운데 무언가가 있고, 전체적으로 파란색 이미지

ex) 사슴

가운데 갈색의 무언가가 있고, 녹색 배경을 가진 이미지

Linear Classifier: Visual Viewpoint

템플릿:

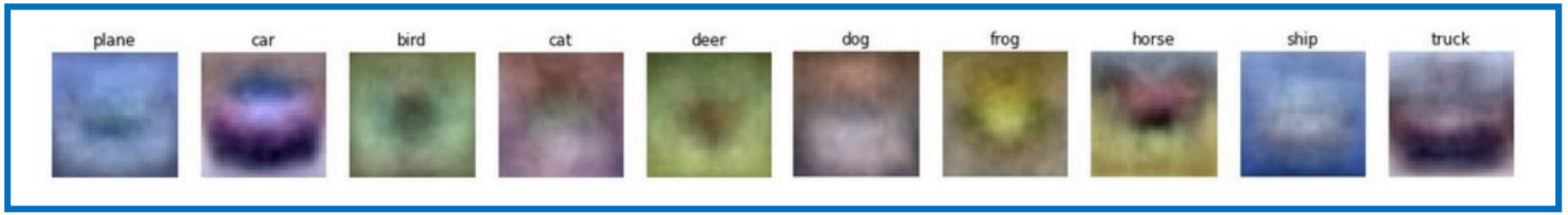


이렇게 숲 속에 주차된 차 이미지가
입력 이미지로 주어진다면

Linear Classifier에게 혼돈을 불러일으킬 수 있다!

Linear Classifier: Visual Viewpoint

템플릿:



녹색 배경은 deer template에 더 잘 매칭
(더 높은 점수를 받음)

반면에 차 형태는 car template에 더 잘 매칭

Linear Classifier: Visual Viewpoint

Visual Viewpoint에서의 Linear Classifier 실패 요인 2.

Linear Classifier가 하나의 클래스 당 하나의 템플릿만 학습할 수 있다는 데서 오는 문제
-> 하나의 카테고리에 속한 이미지들이 다양한 자세/형태를 가질 수 있다면?

ex) 말은 왼쪽을 보고 있을 수도 있고 오른쪽을 보고 있을 수도 있음

Linear Classifier: Visual Viewpoint

Visual Viewpoint에서의 Linear Classifier 실패 요인 2.

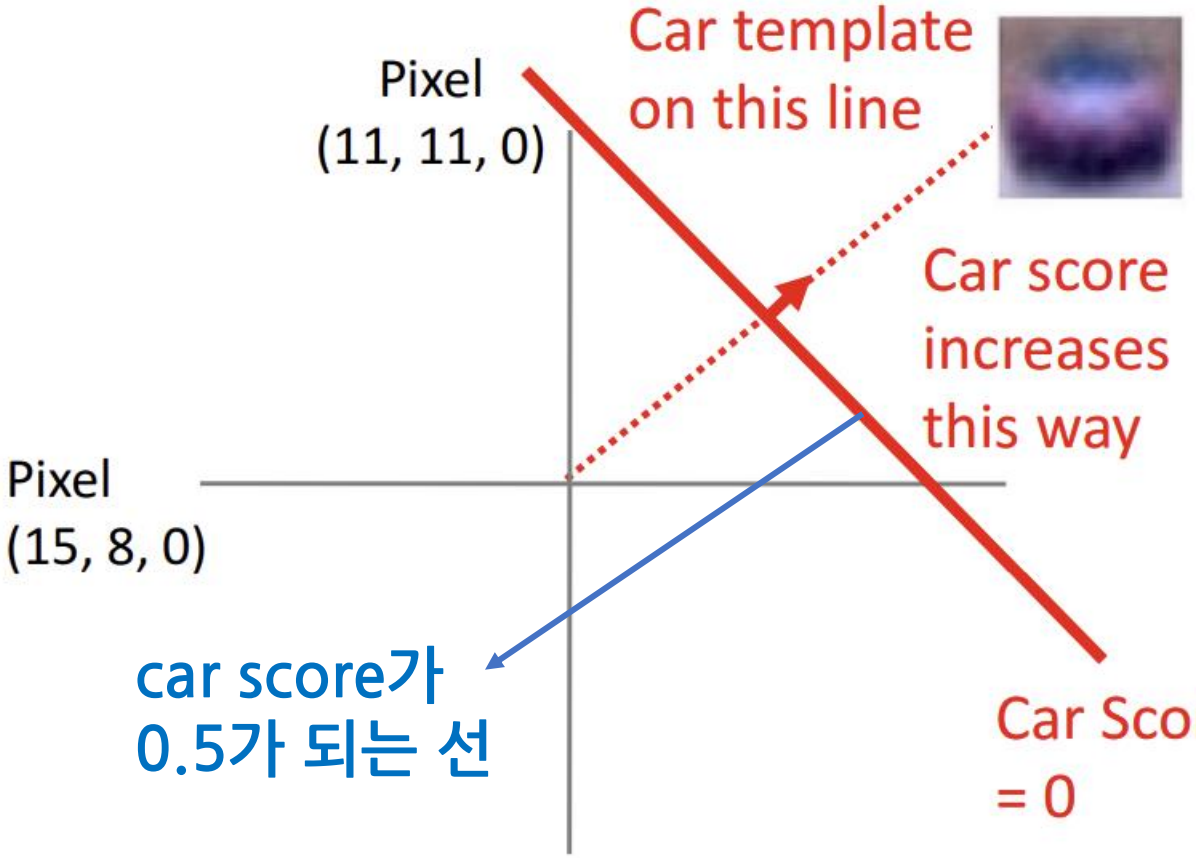
그런데 Linear Classifier는 이렇듯 서로 다른 방향을 보고 있는 말에 대해서 각기 다른 템플릿으로 학습할 수가 없음.

-> Linear Classifier가 하나의 템플릿으로 서로 다른 방향을 보고 있는 말 이미지를 학습하기 위해 선택한 최선의 방법……,

머리가 두 개 달린 말 템플릿 탄생

Linear Classifier: Geometric Viewpoint

이미지에서 두 개의 픽셀을 잡아 각각을 x축, y축에 대응시킴

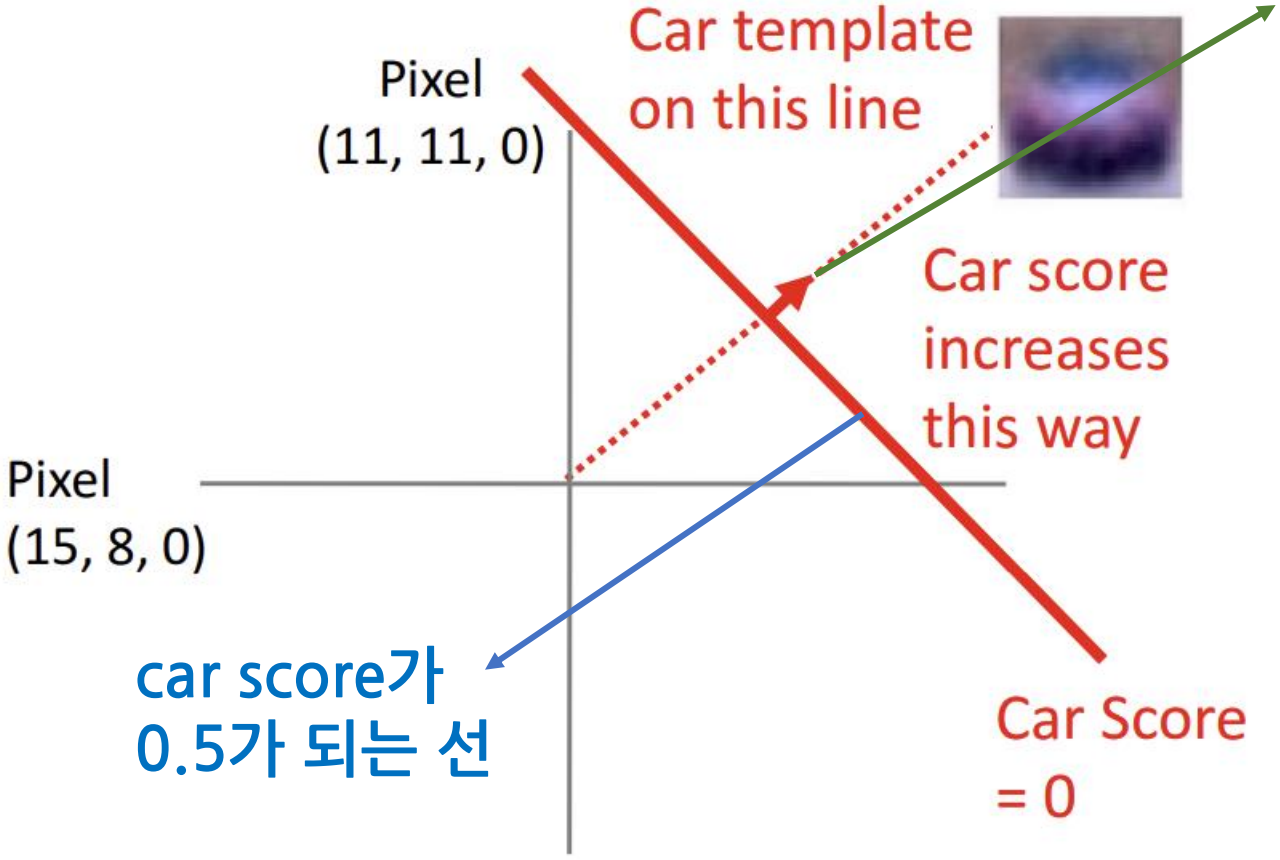


$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

Linear Classifier: Geometric Viewpoint

이미지에서 두 개의 픽셀을 잡아 각각을 x축, y축에 대응시킴



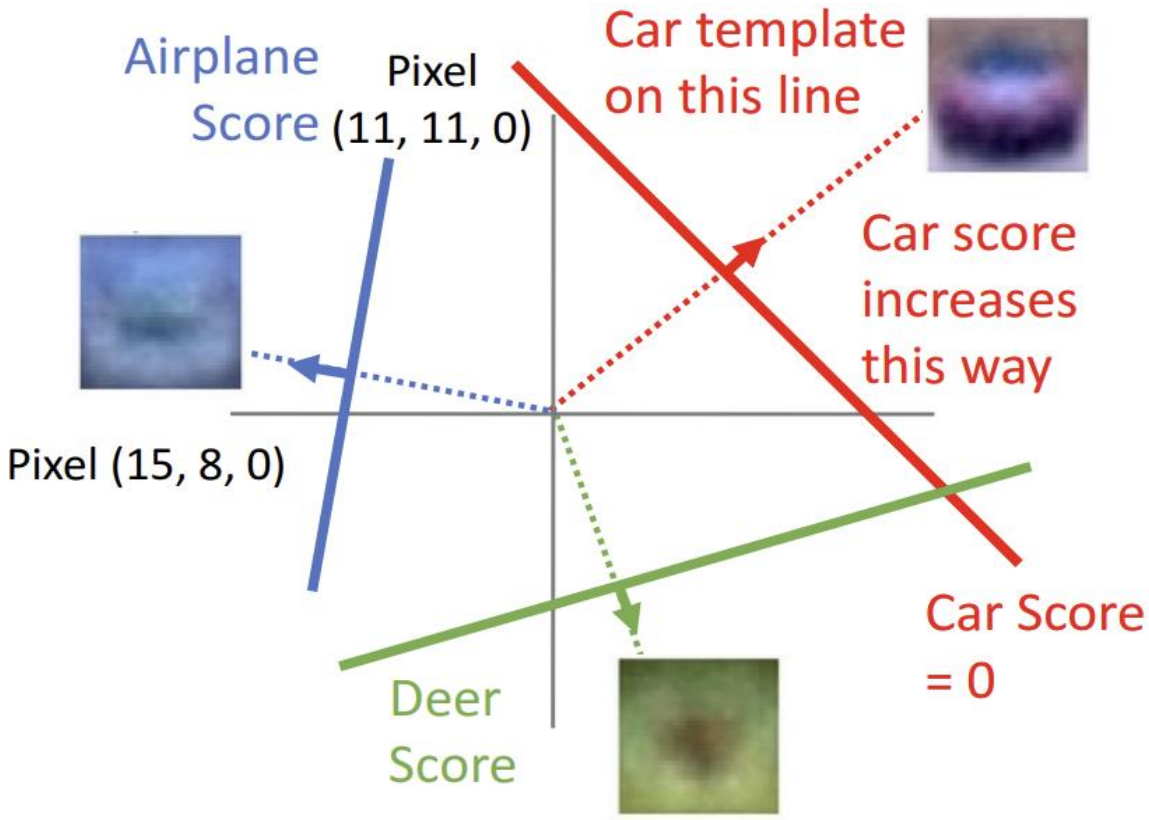
car score=0.5 선에 수직인
방향으로 car score 선형적으로 증가
car 템플릿은 이 수직선 위에 있게 됨

Array of **32x32x3** numbers
(3072 numbers total)

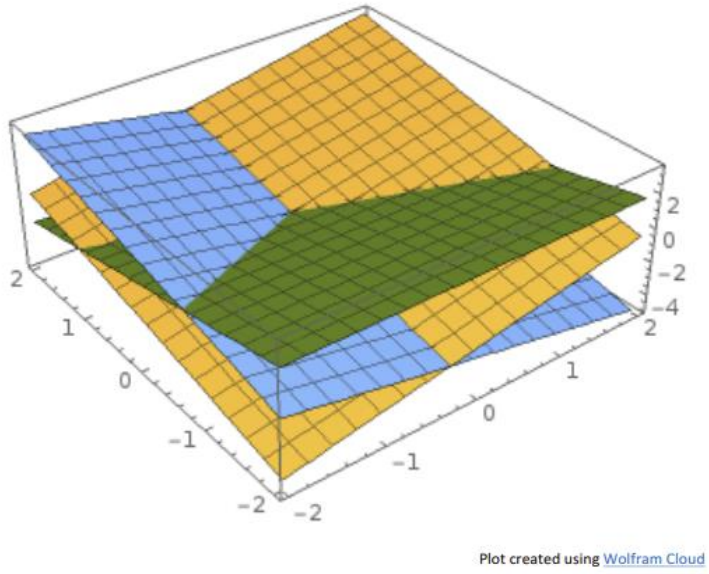
Linear Classifier: Geometric Viewpoint

고차원으로 확장한다면 Linear Classifier는 단순한 직선을 넘어 hyperplane이 될 것

ex) 2차원



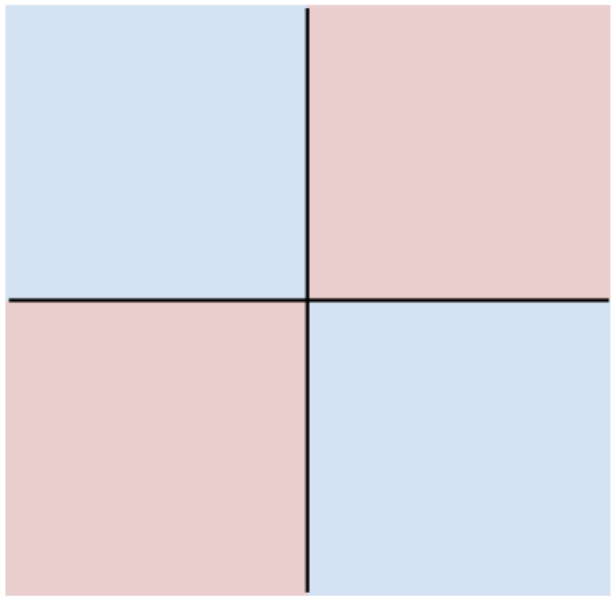
ex) 3차원

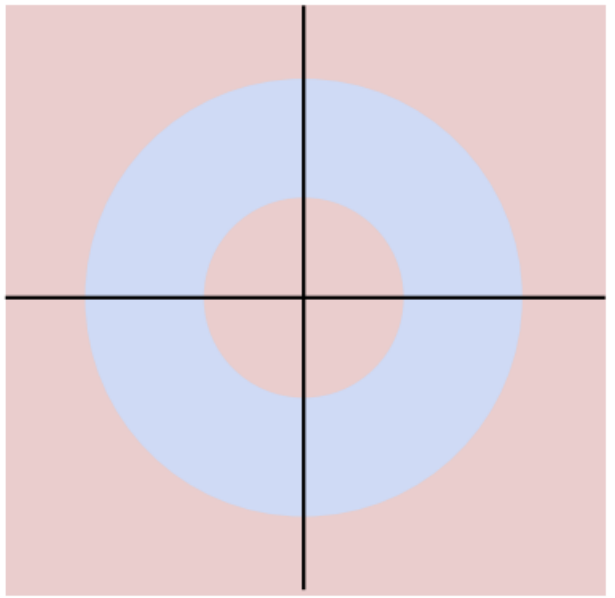


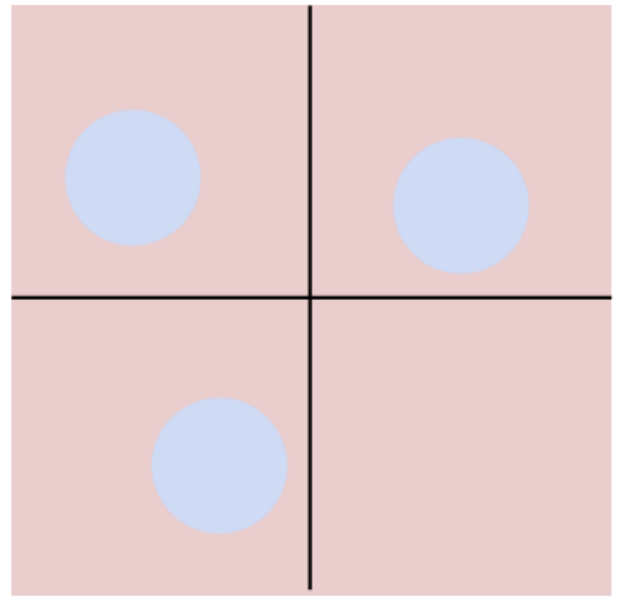
Hard Cases for a Linear Classifier

선형 분류가 안 되는 경우들 -> XOR problem과도 연결

빨간색, 파란색 영역이 서로 다른 카테고리라고 생각했을 때, 이 둘을 분류할 수 있는 하나의 선을 그을 수 없음.



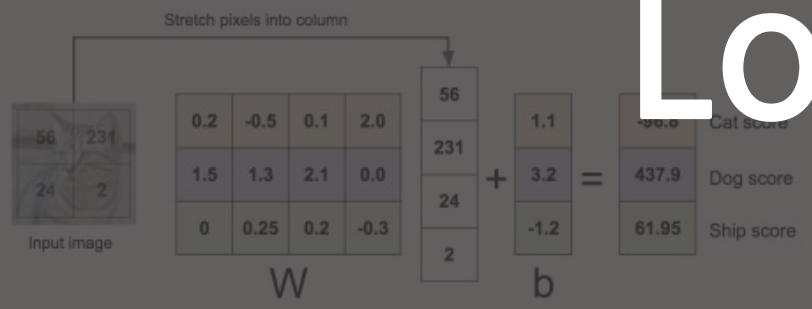




Linear Classifier: Three Viewpoints

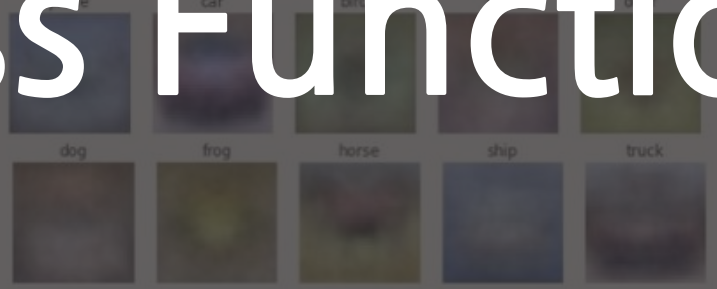
Algebraic Viewpoint

$f(x,W) = Wx$



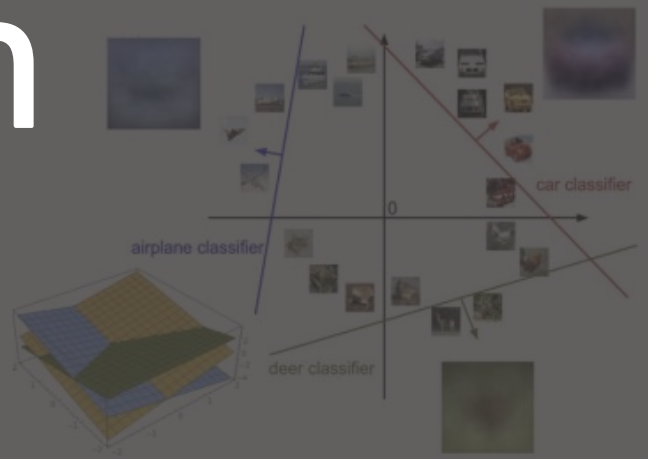
Visual Viewpoint

per class



Geometric Viewpoint

Hyperplanes
cutting up space



세 가지 관점에서 공통적으로 등장하는 '가중치',
가중치 W를 어떻게 찾을 수 있는가?

Loss Function

좋은 가중치 W를 선택하기 위한 Two step

1. Weight 값이 얼마나 좋은지 정량화하는 **Loss Function**

2. Loss Function을 최소화할 수 있는 Weight 찾기(**Optimization**)

Loss Function

Classifier가 잘 동작하는지 나타내 주는 지표

Low loss == good classifier
High loss == bad classifier

Loss Function의 다른 이름

object function, cost function

Negative Loss Function의 다른 이름

reward function, profit function, utility function, fitness function…

Loss Function

수식적으로 봅시다!

Given a dataset of examples

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

x: 입력 이미지

y: 결과값, 10개의 카테고리 각각에 해당하는 라벨
(1~10까지 정수로 표현)

Loss Function

수식적으로 봅시다!

Loss for a single example is

$$L_i(f(x_i, W), y_i)$$

$f(x_i, W)$: 각각의 입력 이미지에 대한 linear classifier의 예측값

y: 정답, 각각의 입력 이미지에 대한 알맞은 라벨

-> 이 둘의 차이를 보는 것이 Loss Function!
(기준은 Loss Function 종류별로 다를 수 있음)

Loss Function

수식적으로 봅시다!

Loss for a single example is

$$L_i(f(x_i, W), y_i)$$

-> 개별 데이터에 대한 Loss 값을

Loss for the dataset is average of per-example losses:

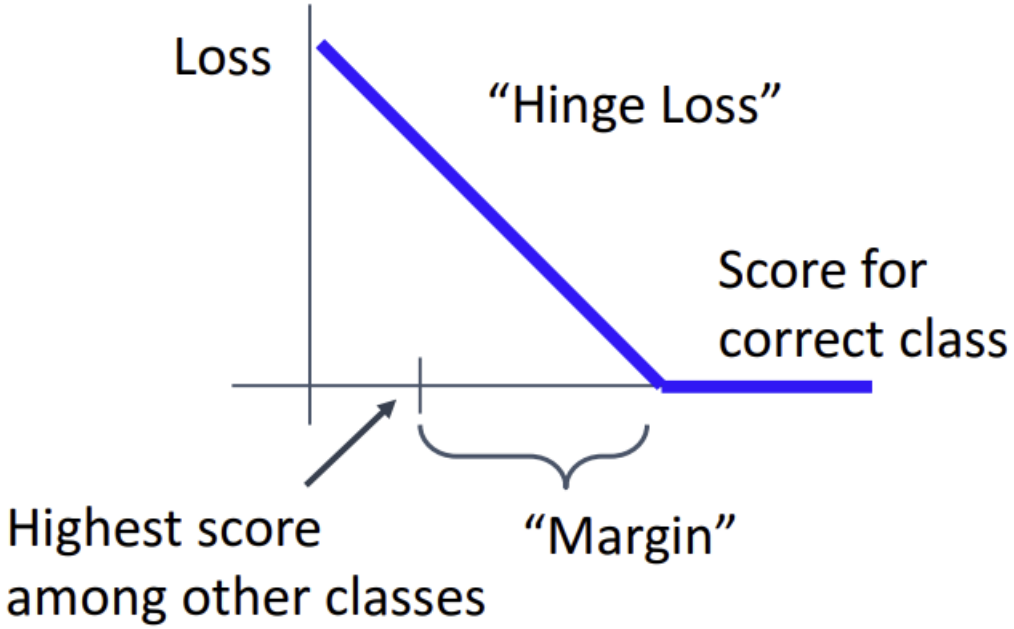
$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

-> 모든 데이터에 대하여 평균 낸 것이 전체 Loss

Multiclass SVM Loss

Classification에서 보편적으로 쓰이는 Loss function

아이디어: 정답 클래스의 score는 다른 클래스의 score에 비해 높아야 한다!

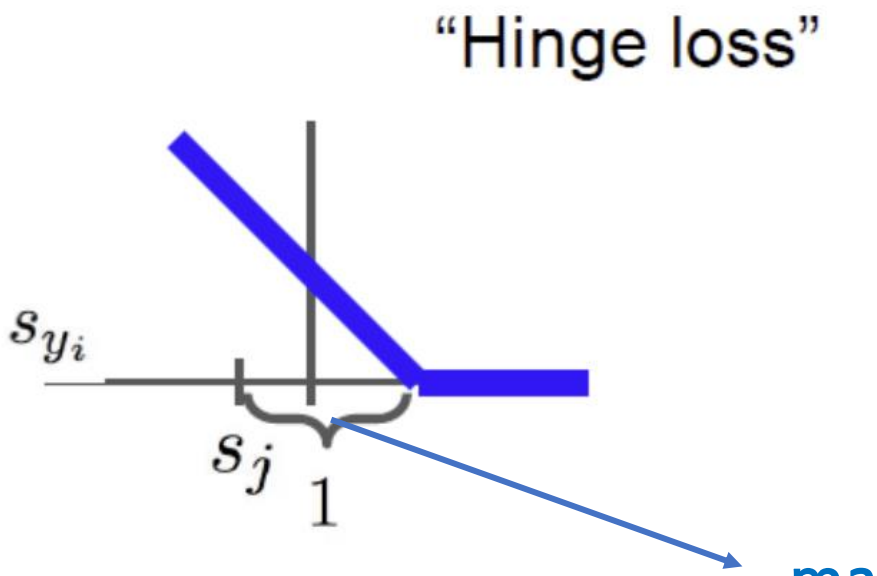


x축: 정답 클래스의 점수(Syi)
y축: Loss

로 두고 Loss Function을 그려 보면

다음과 같은 형태의 '**Hinge Loss**'
(경첩같이 생겼다 하여 Hinge!)

Multiclass SVM Loss



s_j : **오답 클래스의 점수**

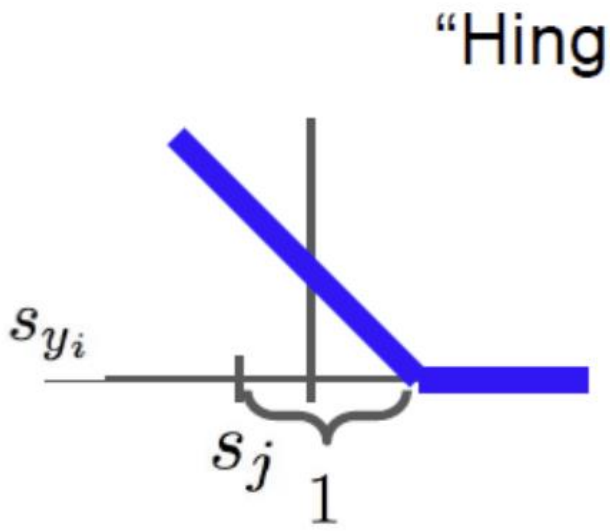
Hinge Loss를 그릴 때 Sj는 오답 클래스 점수 중 가장 높은 것을 고름.

s_{y_i} : **정답 클래스의 점수, x축**

marjin: Syi와 Sj와의 차이(Syi - Sj)

- marjin이 1 이상이 되는 score를 얻으면 Loss는 0
- 그 이하의 score를 얻으면 Loss 존재

Multiclass SVM Loss



Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

marjin(Syi - Sj)이 1 이상일 경우 Loss가 0

-> marjin이 1 이상이라면 Sj - Syi는 항상 음수가 되어,
0과 비교해서 최댓값을 출력할 때 결과 값으로 항상 0이 출력되게 됨. = Loss가 0

Multiclass SVM Loss



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

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

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

Multiclass SVM Loss



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	12.9

전체 Multiclass SVM Loss는
각각의 Loss를 평균 내어 구한다!

L = 1/3 * (2.9 + 0 + 12.9)

Multiclass SVM Loss

생각해보면 좋을 질문들

1. Score 값이 살짝 변화하면 Loss에 영향을 주는가?

한 번 옳게 예측하면 score가 살짝 변한다고 해서 Loss에 영향을 주지는 않는다.

2. Loss의 min과 max는?

0, 무한대

3. Score를 Randomly initialize한다면?

Loss 값 또한 small random 값

Multiclass SVM Loss

생각해보면 좋을 질문들

4. 만약 정답 클래스도 loss sum 계산에 포함된다면?

모든 Loss가 1씩 증가하게 되므로, 순서에는 영향을 주지 않아 결과는 달라지지 않는다.

5. Loss에 sum대신 mean을 사용한다면?

Monotonic transform(전체적인 경향성 해치지 않음)으로,
스케일만 변할 뿐 결과는 달라지지 않는다.

Multiclass SVM Loss

생각해보면 좋을 질문들

6. $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$ **이러한 Loss를 대신 사용한다면?**

더 이상 Multiclass SVM Loss라 할 수 없고, squared hinge loss라 하며,
비선형적으로 바뀌게 된다.

7. L = 0이 되는 W를 찾았다! 이 W는 하나뿐인 값인가?

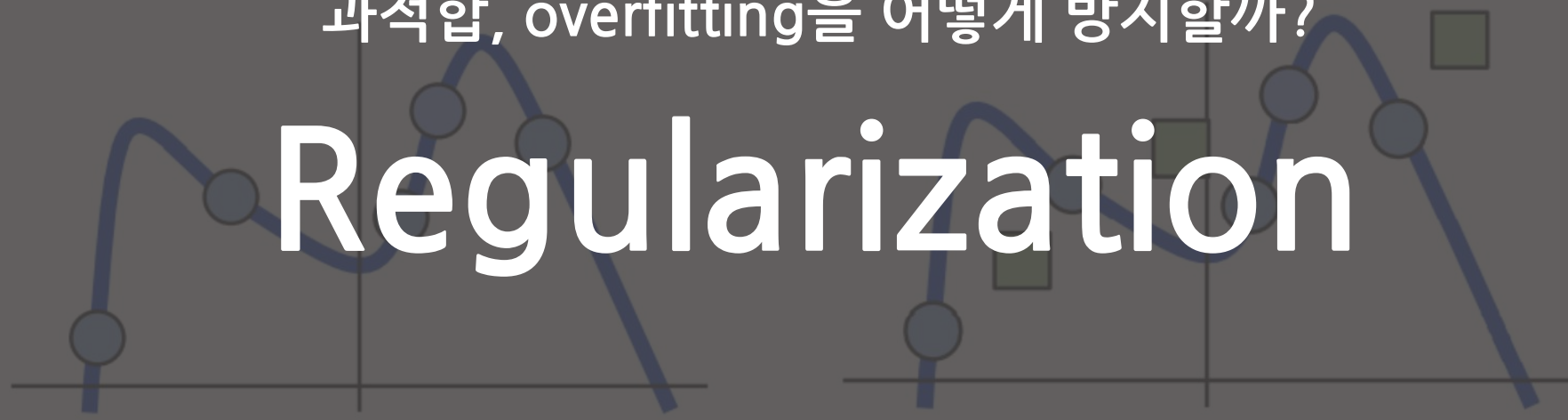
그렇지 않다. Score가 선형적인 관계이기 때문에 2W, 3W 등 W의 배수이면 전체 score도 선형적으로 바뀌기 때문에 Loss를 0으로 만든다!

Regularization: Beyond Training Error

Loss를 0으로 만드는 것이 과연 좋은 일인가?

과적합, overfitting을 어떻게 방지할까?

Regularization



파란색 동그라미: train data
연두색 네모: test data

Regularization: Beyond Training Error

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

λ = regularization strength (hyperparameter)

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

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

More complex:

Dropout

Batch normalization

Cutout, Mixup, Stochastic depth, etc…

Regularization: Beyond Training Error

L2 Regularization 예시

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

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

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

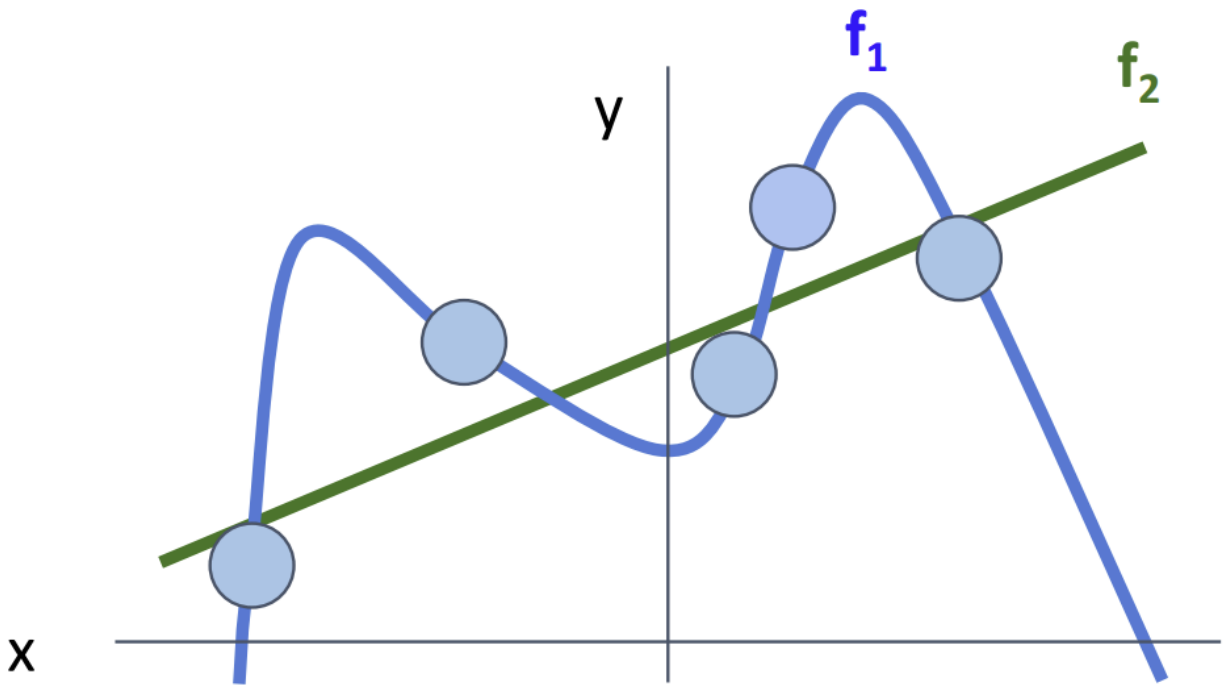
$$w_1^T x = w_2^T x = 1$$

L2 Regularization

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

L2 regularization likes to "spread out" the weights

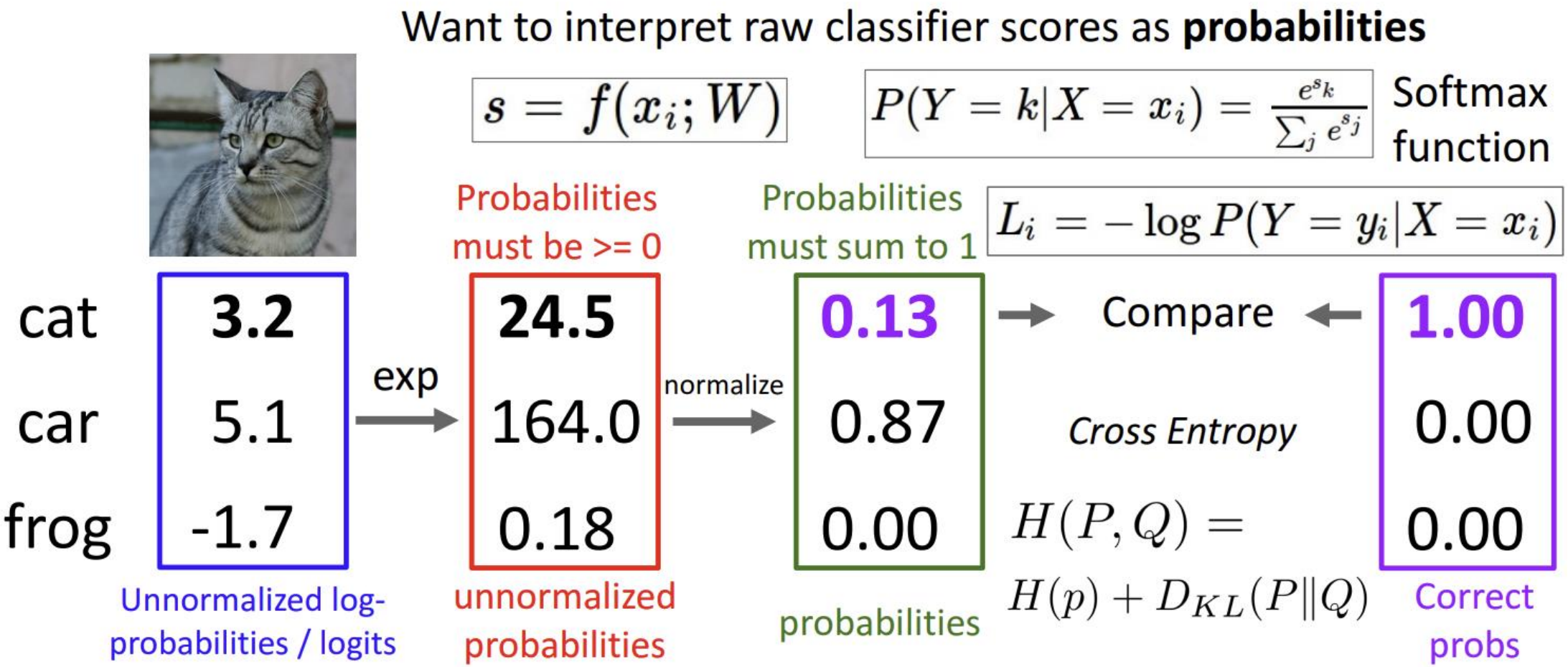
Regularization: Beyond Training Error



f1: 모든 train data에 대해서 과적합
f2: 모든 train data를 포함하고 있진 않으나 보다 심플한 형태

Cross-Entropy Loss (Multinomial Logistic Regression)

아이디어: 모델이 예측한 점수가 나올 확률과 실제 확률을 비교함으로써 Loss를 구한다!



Cross-Entropy Loss (Multinomial Logistic Regression)

생각해보면 좋을 질문들

1. Score 값이 살짝 변화하면 Loss에 영향을 주는가?

multiclass SVM Loss는 변하지 않지만, score 값 자체를 가공해서 이용하는 Cross-Entropy Loss는 변화한다.

2. Loss의 min과 max는?

0, 무한대이나, 최솟값이 0이 되는 경우는 극히 희박하다.

Cross-Entropy Loss (Multinomial Logistic Regression)

생각해보면 좋을 질문들

3. Score를 Randomly initialize한다면?

Loss 값 또한 small random 값

4. 정답 클래스의 score가 두 배가 된다면?

multiclass SVM Loss는 동일하나, Cross-Entropy Loss는 감소한다.

감사합니다