모두의연구소

Image Classification Pipe line

모두의연구소
박은수 Research Director

모두의연구소

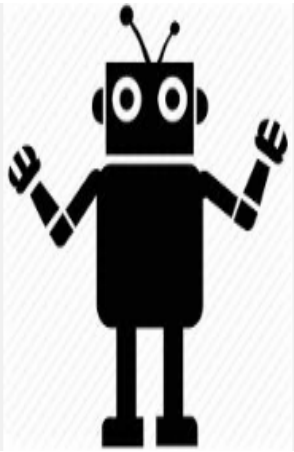
딥러닝의 30가지 적용사례

https://brunch.co.kr/@itschloe1/23

물체인식에서의 딥러닝

모두의연구소

인간 vs. 기계



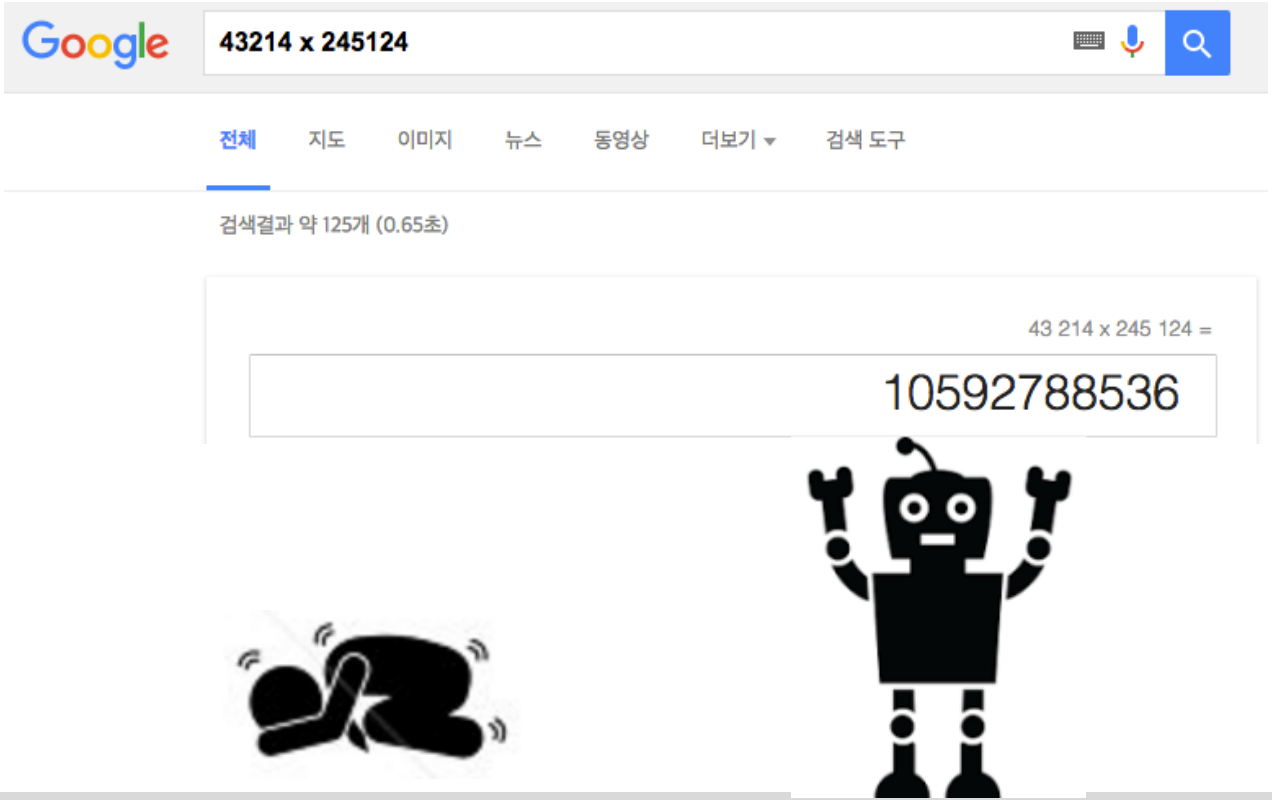
물체인식에서의 딥러닝

모두의연구소

43214 x 245124 = ?

물체인식에서의 딥러닝

모두의연구소



물체인식에서의 딥러닝

모두의연구소

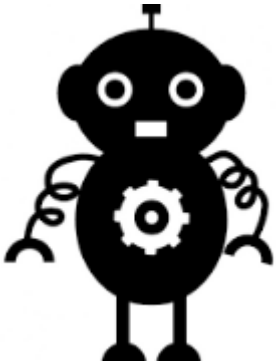
고양이랑 강아지 구분하기

물체인식에서의 딥러닝

모두의연구소

왼쪽 고양이
오른쪽 강아지 ~ !!

아...



물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식

{고양이, 강아지, 버스, 자동차, ..., 집}

→ 고양이

* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

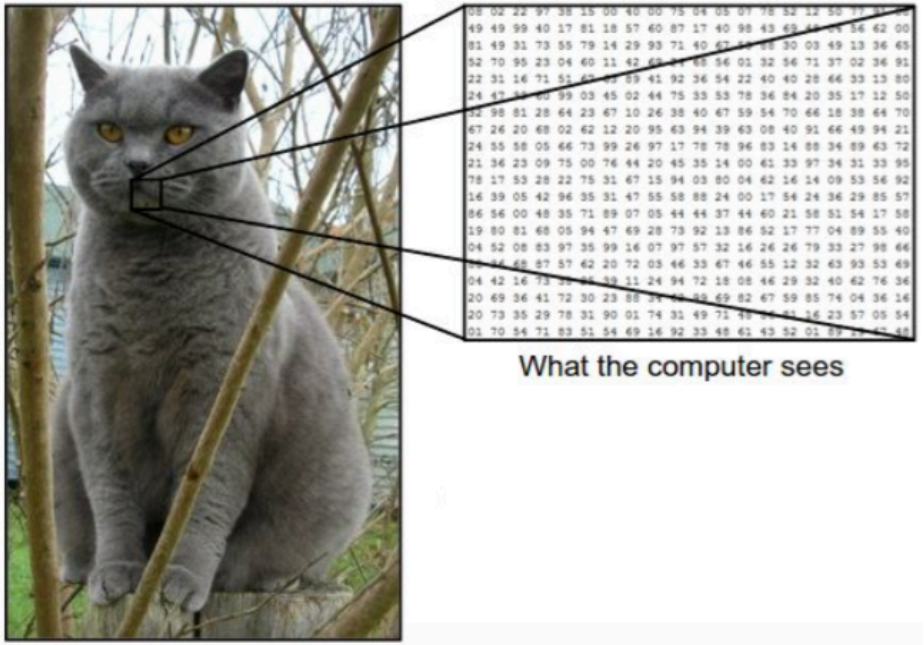
물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식

영상은 [0, 255] 범위의 값을 갖는 RGB 3채널로 구성되어 있다

옆의 고양이는 300x100x3 의 배열 (3은 RGB (Red, Green, Blue) 채널)



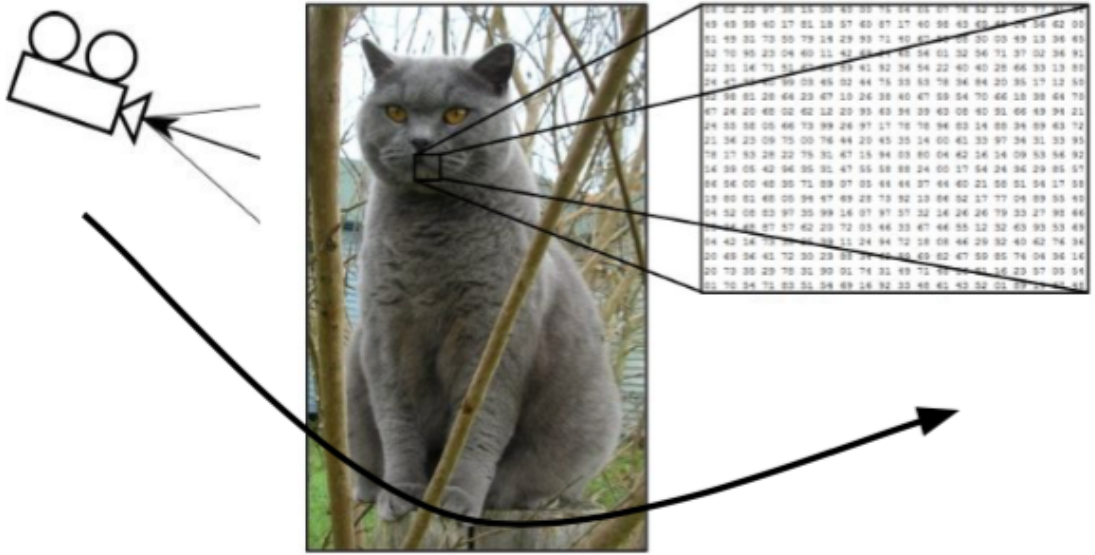
* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식의 어려움

카메라 위치에 따른
변화



* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체 인식의 어려움

조명에 따른 변화

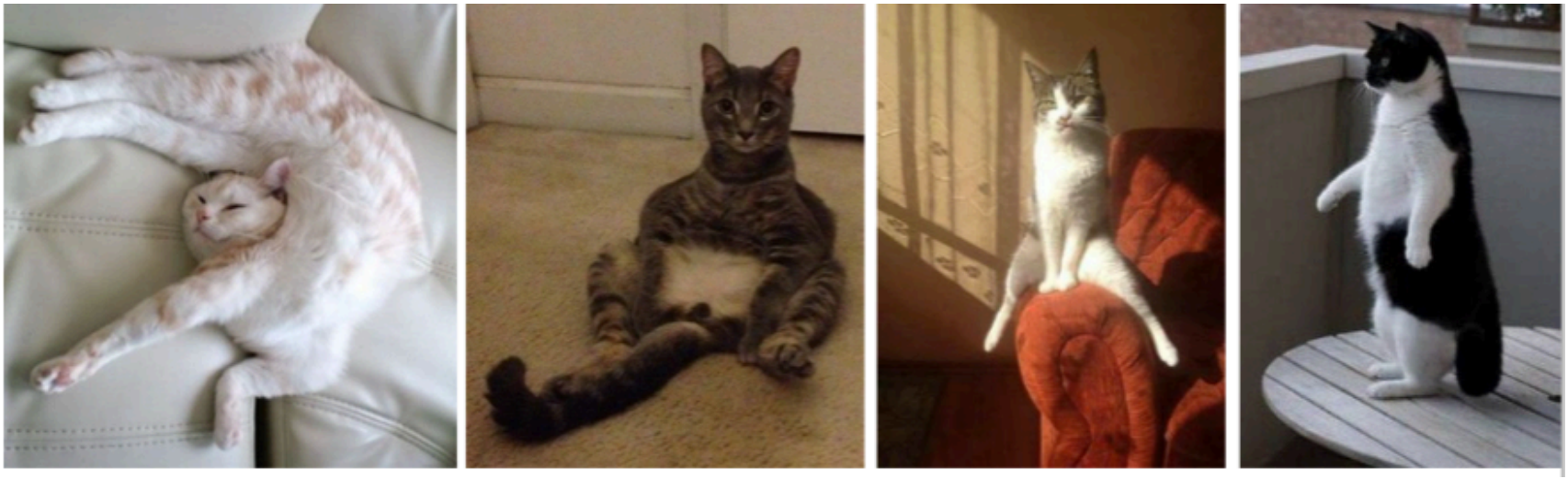
* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식의 어려움

객체 형태의 변화



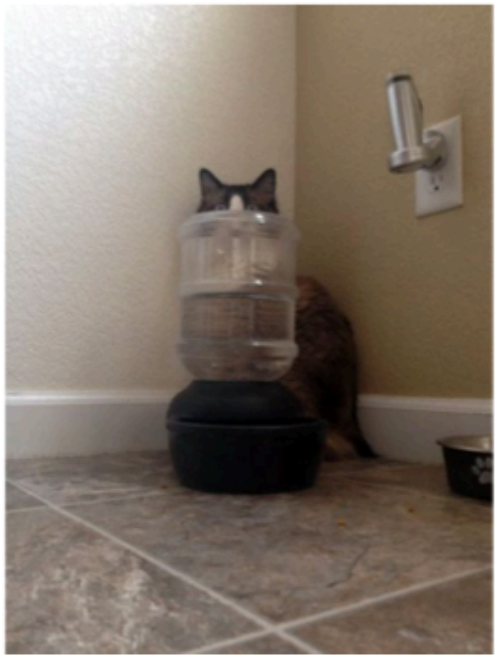
* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

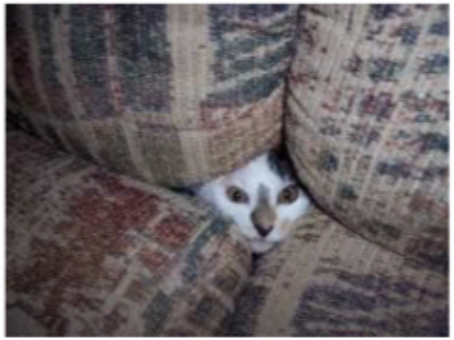
물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식의 어려움

가려짐





* From : Lecture slide 2 of “CS231n : Convolutional Neural Networks for Visual Recognition”

물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식의 어려움

배경과의 유사성

* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식의 어려움

같으면서 다른 형태의 물체

* From : Lecture slide 2 of “CS231n : Convolutional Neural Networks for Visual Recognition”

물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식의 어려움

```
def predict(image):
    # ????
    return class_label
```

고양이와 개를 구분하는 알고리즘을 만들어 보세요

물체인식에서의 딥러닝

모두의연구소

컴퓨터 비젼에서의 물체인식의 어려움

죄송합니다

어려운걸 알고있습니다

Data-driven 방법

모두의연구소

1. 이미지와 그에 해당하는 레이블(label)을 모음
2. 머신러닝 방법론으로 분류기를 학습
3. 학습된 분류기를 테스트 이미지에 평가

```
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```

* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

첫번째 분류기 : Nearest Neighbor Classifier

모두의연구소

간단한 분류기를 만들어 봅시다 : Nearest Neighbor Classifier

```
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```

- 모든 학습용 이미지와 그들의 레이블(label)을 저장
- 학습용 이미지 중 가장 유사한 이미지의 레이블 반환

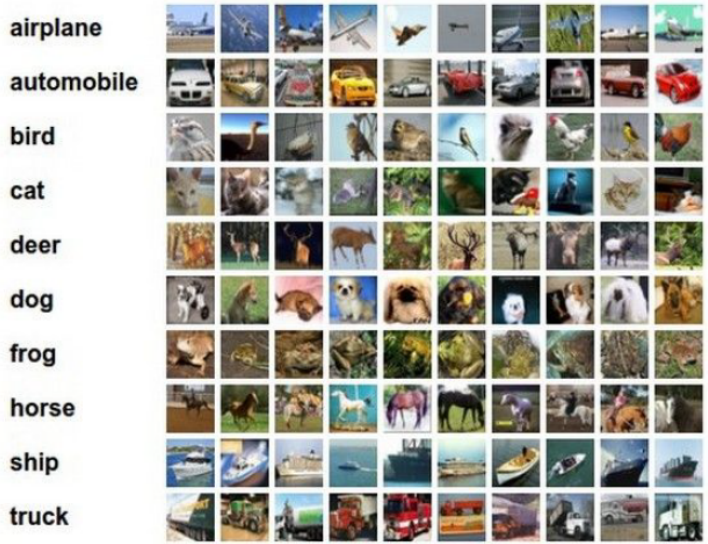
* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

첫번째 분류기 : Nearest Neighbor Classifier

모두의연구소

Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.

32x32 크기임



For every test image (first column), examples of nearest neighbors in rows

* From : Lecture slide 2 of "CS231n : Convolutional Neural Networks for Visual Recognition"

가장 유사한 그림은 어떻게 찾는가?

모두의연구소

Distance 를 측정

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

test image					training image					pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	=	46	12	14	1	add → 456
90	23	128	133		8	10	89	100		82	13	39	33	
24	26	178	200		12	16	178	170		12	10	0	30	
2	0	255	220		4	32	233	112		2	32	22	108	

Nearest Neighbor classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

remember the training data

Nearest Neighbor classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

for every test image:

- find nearest train image with L1 distance
- predict the label of nearest training image

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

질문 : 학습용 이미지의 수와 우리가 만든 Nearest Neighbor Classifier의 처리 속도와의 관계는?

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

질문 : 학습용 이미지의 수와 우리가 만든 Nearest Neighbor Classifier의 처리 속도와의 관계는?

선형적임 ☹

이건 좋지않다.

- 실제 어플리케이션에선 테스트 타임이 중요하다.
- CNN은 반대임. 트레이닝이 오래 걸리고, 테스트는 빠름

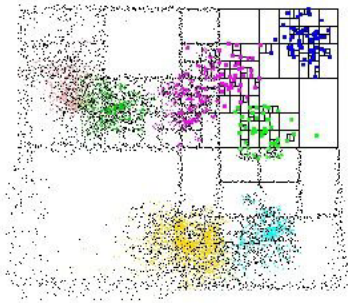
Aside: Approximate Nearest Neighbor
find approximate nearest neighbors quickly

ANN: A Library for Approximate Nearest Neighbor Searching

David M. Mount and Sunil Arya

Version 1.1.2

Release Date: Jan 27, 2010



What is ANN?

ANN is a library written in C++, which supports data structures and algorithms for both exact and approximate nearest neighbor searching in arbitrarily high dimensions.

In the nearest neighbor problem a set of data points in d-dimensional space is given. These points are preprocessed into a data structure, so that given any query point q, the nearest or generally k nearest points of P to q can be reported efficiently. The distance between two points can be defined in many ways. ANN assumes that distances are measured using any class of distance functions called Minkowski metrics. These include the well known Euclidean distance, Manhattan distance, and max distance.

Based on our own experience, ANN performs quite efficiently for point sets ranging in size from thousands to hundreds of thousands, and in dimensions as high as 20. (For applications in significantly higher dimensions, the results are rather spotty, but you might try it anyway.)

The library implements a number of different data structures, based on kd-trees and box-decomposition trees, and employs a couple of different search strategies.

The library also comes with test programs for measuring the quality of performance of ANN on any particular data sets, as well as programs for visualizing the structure of the geometric data structures.

FLANN - Fast Library for Approximate Nearest Neighbors

- Home
- News
- Publications
- Download
- Changelog
- Repository

What is FLANN?

FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. It contains a collection of algorithms we found to work best for nearest neighbor search and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset.

FLANN is written in C++ and contains bindings for the following languages: C, MATLAB and Python.

News

- (14 December 2012) Version 1.8.0 is out bringing incremental addition/removal of points to/from indexes
- (20 December 2011) Version 1.7.0 is out bringing two new index types and several other improvements.
- You can find binary installers for FLANN on the Point Cloud Library project page. Thanks to the PCL developers!
- Mac OS X users can install flann though MacPorts (thanks to Mark Moll for maintaining the Portfile)
- New release introducing an easier way to use custom distances, kd-tree implementation optimized for low dimensionality search and experimental MPI support
- New release introducing new C++ templated API, thread-safe search, save/load of indexes and more.
- The FLANN license was changed from LGPL to BSD.

How fast is it?

In our experiments we have found FLANN to be about one order of magnitude faster on many datasets (in query time), than previously available approximate nearest neighbor search software.

Publications

More information and experimental results can be found in the following papers:

- Marius Muja and David G. Lowe: **"Scalable Nearest Neighbor Algorithms for High Dimensional Data"**. Pattern Analysis and Machine Intelligence (PAMI), Vol. 36, 2014. [PDF] [BibTeX]
- Marius Muja and David G. Lowe: **"Fast Matching of Binary Features"**. Conference on Computer and Robot Vision (CRV) 2012. [PDF] [BibTeX]
- Marius Muja and David G. Lowe, **"Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration"**, in International Conference on Computer Vision Theory and Applications (VISAPP'09), 2009 [PDF] [BibTeX]

어떤 Distance 함수를 사용할까?

모두의연구소

Distance 함수는 여러개 입니다. 여기서
어떤 걸 사용하는게 좋을까요?

이렇게 변형가능한 우리의 설정 같은 것들을 하이버 파라미터라고 합니다

일반적으로 아래의 2개를 사용합니다

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

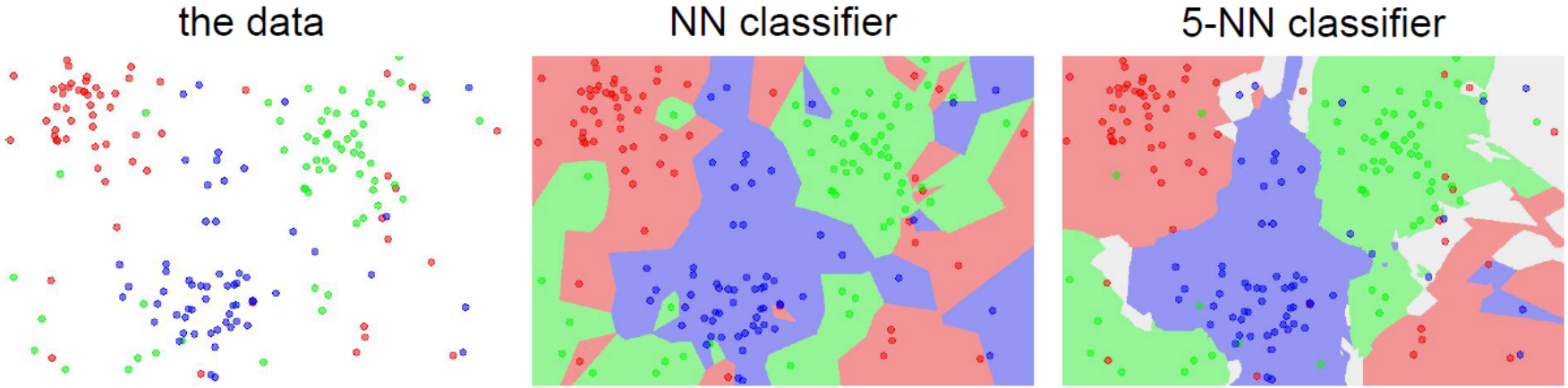
L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

Nearest Neighbor은 알겠습니다. 제일 비슷한 것 찾아서 그 비슷한 것의 레이블을 반환하는 것이군요

모두의연구소

k-Nearest Neighbor은 무엇인가?



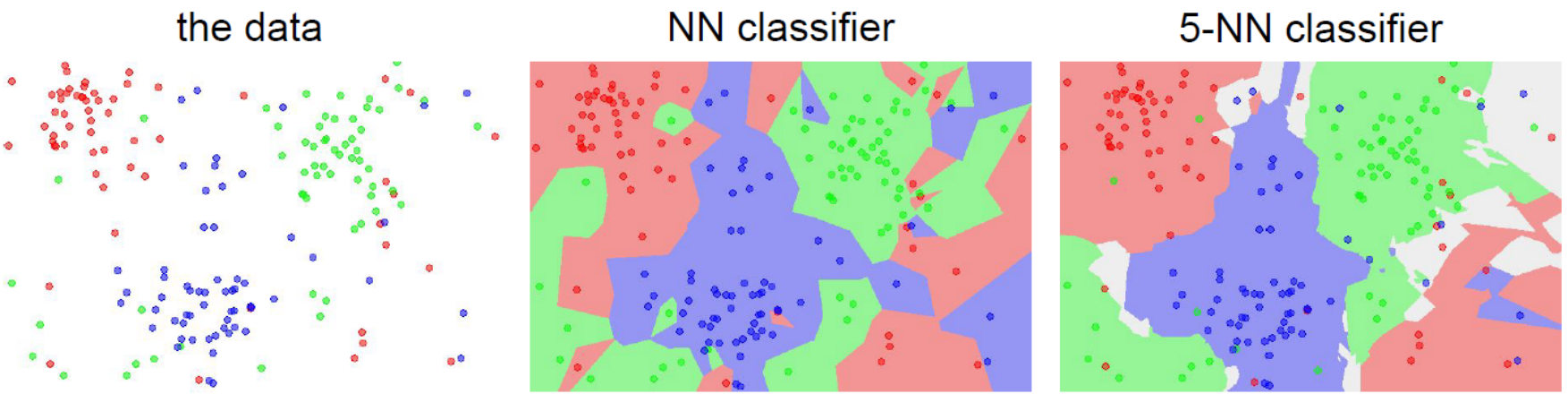
http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Nearest Neighbor은 알겠습니다. 제일 비슷한 것 찾아서 그 비슷한 것의 레이블을 반환하는 것이군요

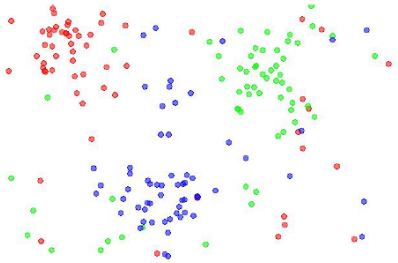
모두의연구소

k-Nearest Neighbor은 무엇인가?

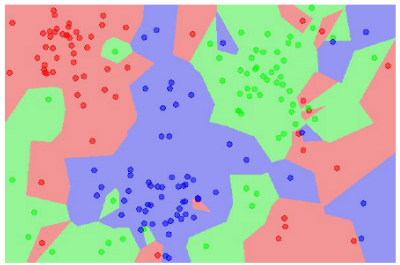
k개의 Nearest Neighbor를 찾고, k개 중 많이 나온 레이블을 선택 (voting)



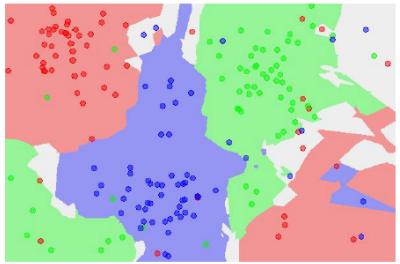
the data



NN classifier



5-NN classifier



이제 NN (Nearest Neighbor) 분류기를 만들었다고 치자

모두의연구소

질문 : NN 분류기(Nearest Neighbor Classifier)를 학습할때 사용한 이미지를 테스트 할때 넣어보면 정확도는 ?

질문 : k-NN 분류기(Nearest Neighbor Classifier)에 학습할때 사용한 이미지를 테스트 할때 넣어보면 정확도는 ?

고민이 생깁니다 ...

모두의연구소

- Distance 함수로는 뭘 사용하는게 좋을까?
- k 값을 뭘로 결정하는게 좋을까?

이러한 파라미터들을 하이퍼 파라미터 (hyper parameter)라고 합니다

고민이 생깁니다 ...

모두의연구소

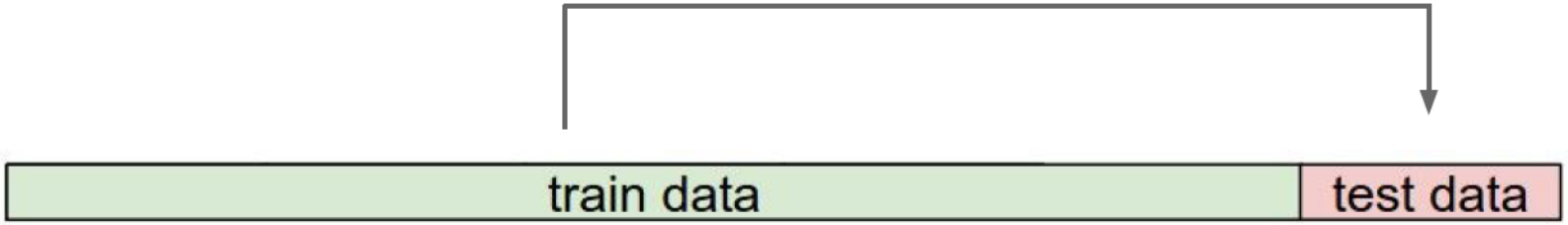
- Distance 함수로는 뭘 사용하는게 좋을까?
- k 값을 뭘로 결정하는게 좋을까?

이러한 파라미터들을 하이퍼 파라미터 (hyper parameter)라고 합니다

실제 풀고자 하는 문제에 따라 다르다

그러니 가능한 경우의 수들을 실험해 보고 결과를 살펴봐야 한다

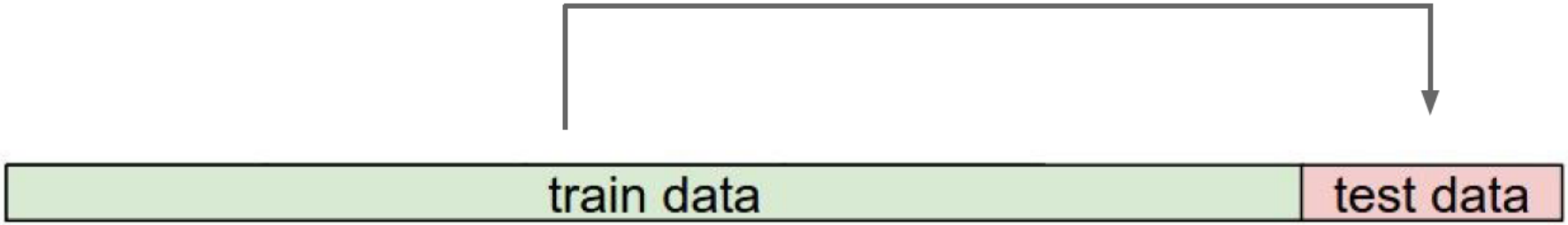
하이퍼 파라미터를 선택 한 후 테스트 데이터를 통해 확인

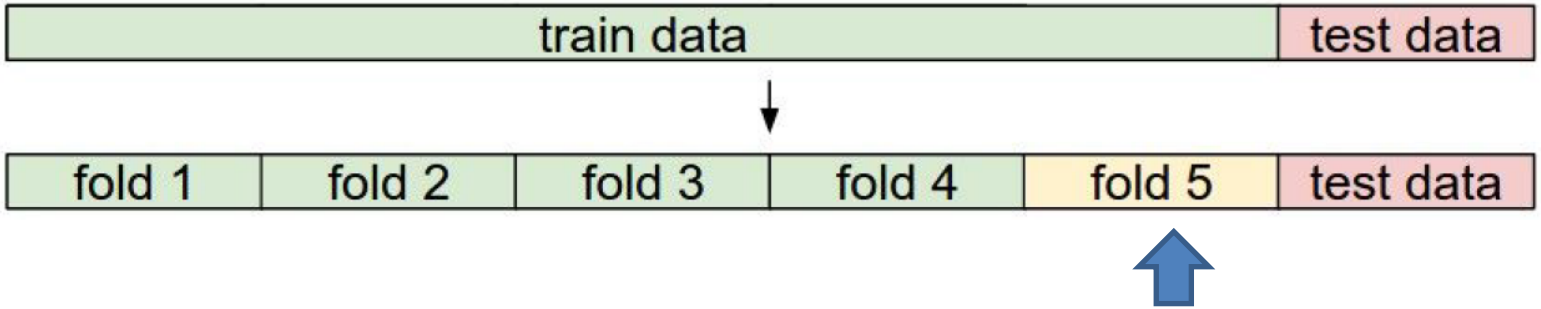


하이퍼 파라미터를 선택 한 후 테스트 데이터를 통해 확인

안 좋은 생각이다. 테스트 데이터는 실제 환경에서도 잘 작동됨을 보여야 할 최후의 보루

마지막을 위해 아껴라~!!



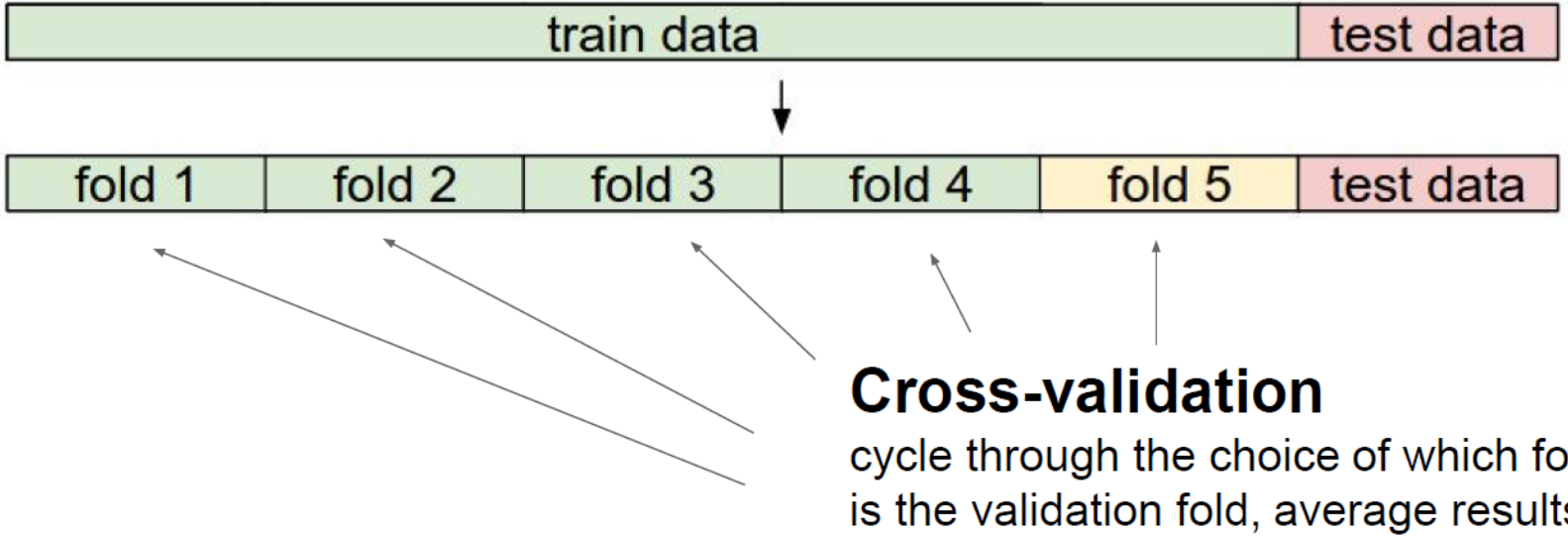


트레이닝(training)에 사용 할 데이터의 일정 부분을 떼어 내어 하이퍼파라미터를 튜닝하자

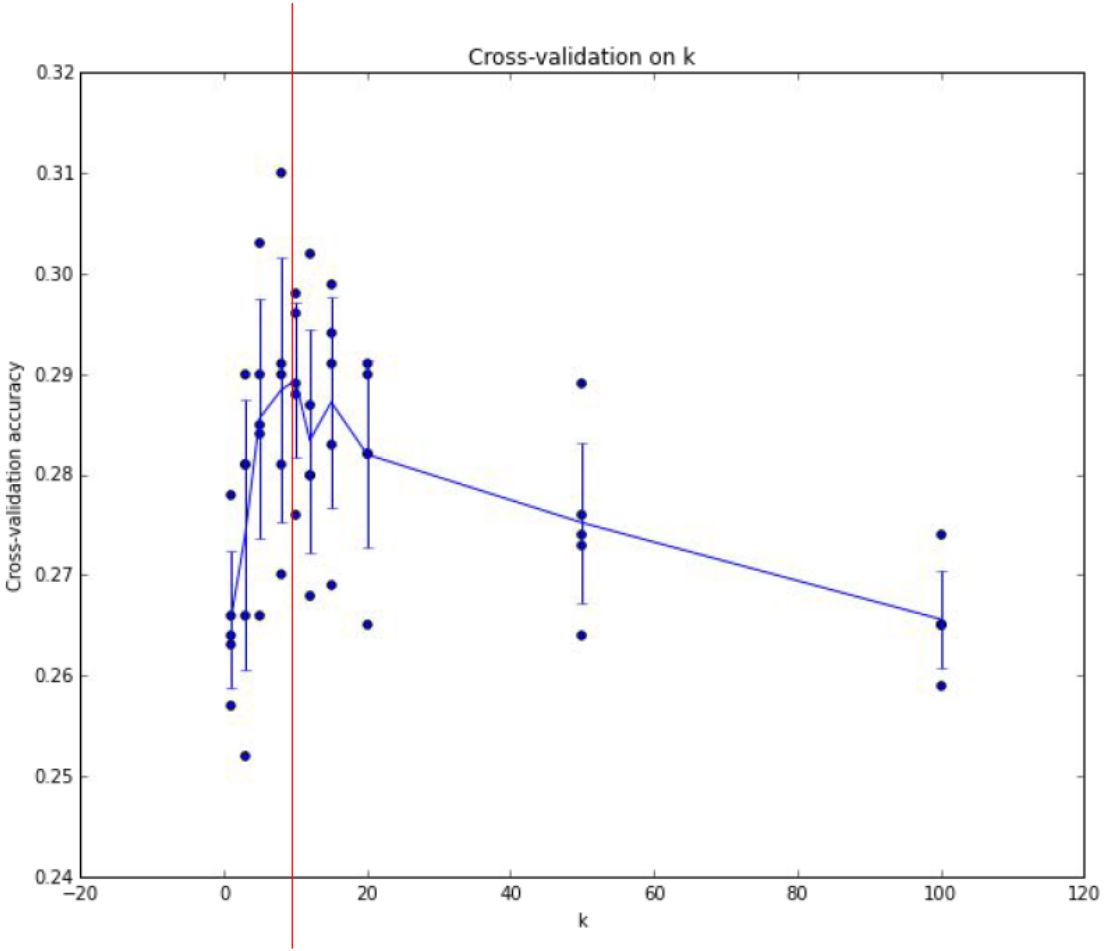
Validation Data

즉, fold1, fold2, fold3, fold4를 트레이닝에 사용하고 테스트를 fold5에 수행해본다

Cross-Validation



이제 테스트에 루프를 돌면서 트레이닝에 사용할 데이터와 테스트에 사용할 데이터 셋을 바꿔가면서 실험해 봅니다



Example of
5-fold cross-validation
for the value of **k.**

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best
for this data)

모두의연구소

- 참고하면 좋은 동영상
- Lecture 2 | Image Classification
    - https://www.youtube.com/watch?v=0oUX-nOEjG0&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv
- Andrew Ng: Artificial Intelligence is the New Electricity
  - https://www.youtube.com/watch?v=21EiKfQYZXc&t=3105s

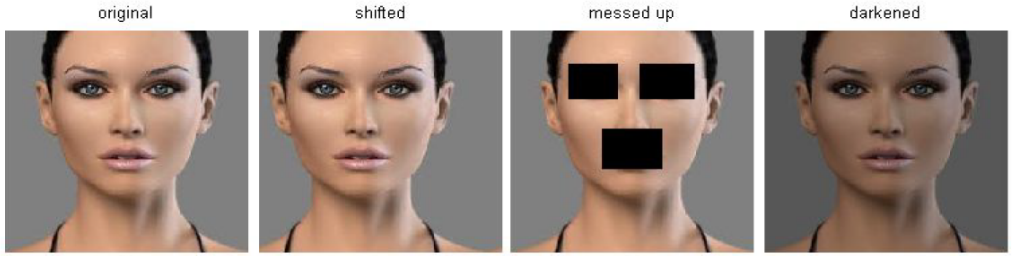
모두의연구소

Linear Classification

모두의연구소
박은수 Research Director

k-Nearest Neighbor 로 영상인식은 하지 않습니다

- 이걸 적용할 경우 테스트 결과가 매우 안좋습니다
- 단순한 픽셀 단위의 Distance metrics로는 좋은 결과를 기대하기 어렵습니다



(all 3 images have same L2 distance to the one on the left)

그럼 뭘로 하나요 ?

픽셀값들을 조합하여 단순히 각 레이블에 대한 점수를 부여하면 어떻까요? Score function?

Parametric Approach

모두의연구소

- 분류기의 구성
  - Score function
  - Loss function
  - Optimization

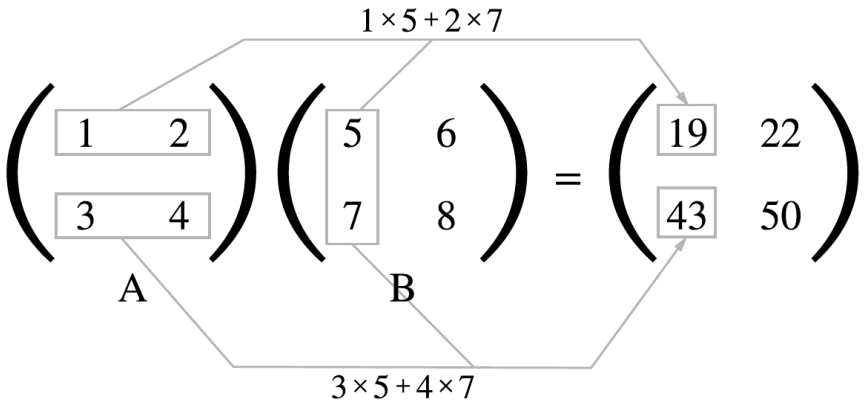
모두의연구소

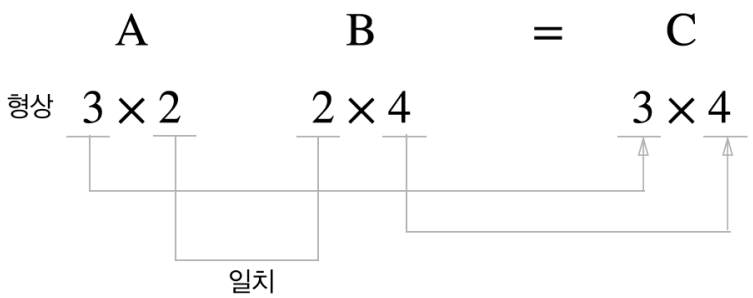
들어가기 전에 ....

다차원 배열의 계산

모두의연구소

- 다차원 배

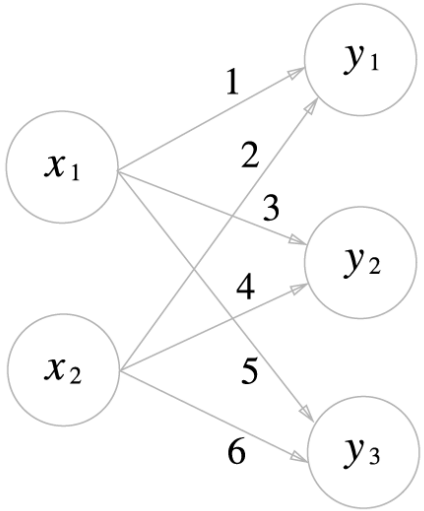




다차원 배열의 계산

모두의연구소

- **다차원 배열**



$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

$$\underset{2}{X} \quad \underset{2\times 3}{W} \quad = \quad \underset{3}{Y}$$

일치

Matrix 연산 리뷰 : Matrix-vector

모두의연구소

Example

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 16 \\ 4 \\ 7 \end{bmatrix}$$

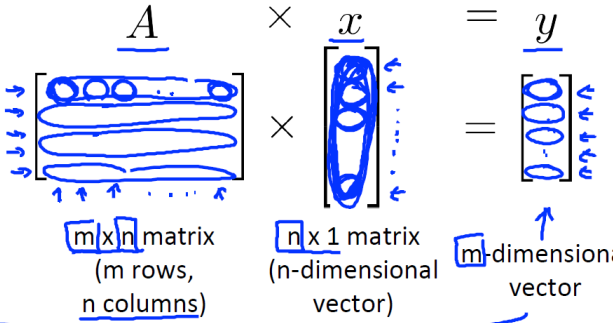
3×2 2×1 3×1 matrix

1×1 + 3×5 = 16

4×1 + 0×5 = 4

2×1 + 1×5 = 7

Details:



To get y_i, multiply A's i^{th} row with elements of vector x, and add them up.

Matrix 연산 리뷰 : Matrix-vector

모두의연구소

Example

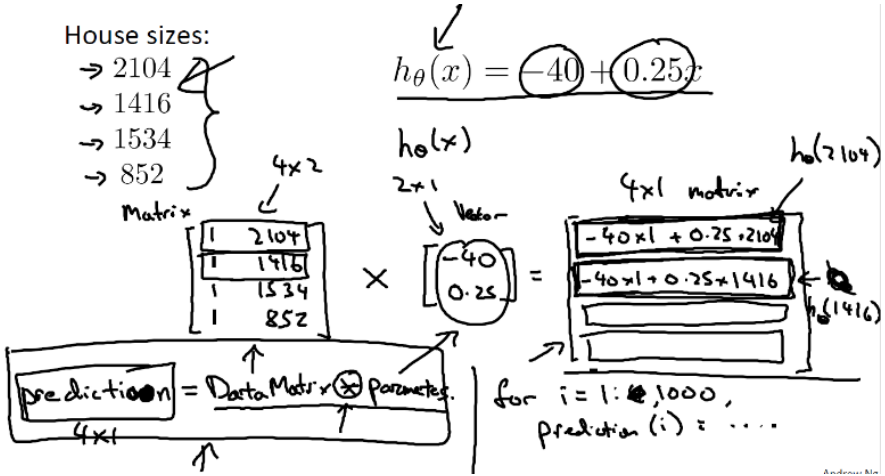
$$\begin{bmatrix} 1 & 2 & 1 & 5 \\ 0 & 3 & 0 & 4 \\ -1 & -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 14 \\ 13 \\ -7 \end{bmatrix}$$

3×4 · 4×1 · 3×1

$1 \times 1 + 2 \times 3 + 1 \times 2 + 5 \times 1 = 14$

$0 \times 1 + 3 \times 3 + 0 \times 2 + 4 \times 1 = 13$

$-1 \times 1 + (-2) \times 3 + 0 \times 2 + 0 \times 1 = -7$



Matrix 연산 리뷰 : Matrix-matrix

모두의연구소

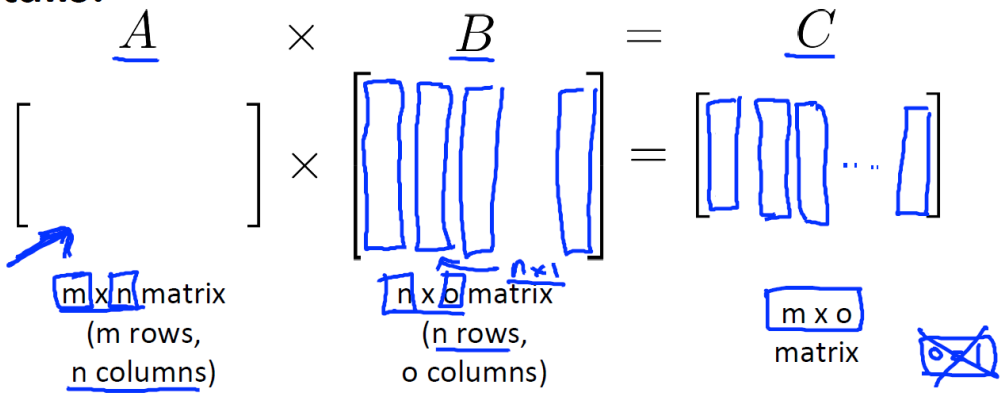
Example

$$\begin{bmatrix}1 & 3 & 2\\ 4 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 3\\ 0 & 1\\ 5 & 2\end{bmatrix}=\begin{bmatrix}11 & 10\\ 9 & 14\end{bmatrix}$$

$$\begin{bmatrix}1 & 3 & 2\\ 4 & 0 & 1\end{bmatrix}\times\begin{bmatrix}1\\ 0\\ 5\end{bmatrix}=\begin{bmatrix}11\\ 9\end{bmatrix}$$

$$\begin{bmatrix}1 & 3 & 2\\ 4 & 0 & 1\end{bmatrix}\times\begin{bmatrix}3\\ 1\\ 2\end{bmatrix}=\begin{bmatrix}10\\ 14\end{bmatrix}$$

Details:



The i^{th} column of the matrix C is obtained by multiplying A with the i^{th} column of B. (for i = 1,2,…,o)

From : Andrew Ng, "Machine Learning : Linear Algebra review", Lecture note; Coursera

Matrix 연산 리뷰 : Matrix-matrix

모두의연구소

House sizes:

$$\begin{Bmatrix} 2104 \\ 1416 \\ 1534 \\ 852 \end{Bmatrix}$$

Have 3 competing hypotheses:

1. $h_\theta(x) = -40 + 0.25x$
2. $h_\theta(x) = 200 + 0.1x$
3. $h_\theta(x) = -150 + 0.4x$

Matrix

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} = \begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$

Matrix

Prediction of First ho

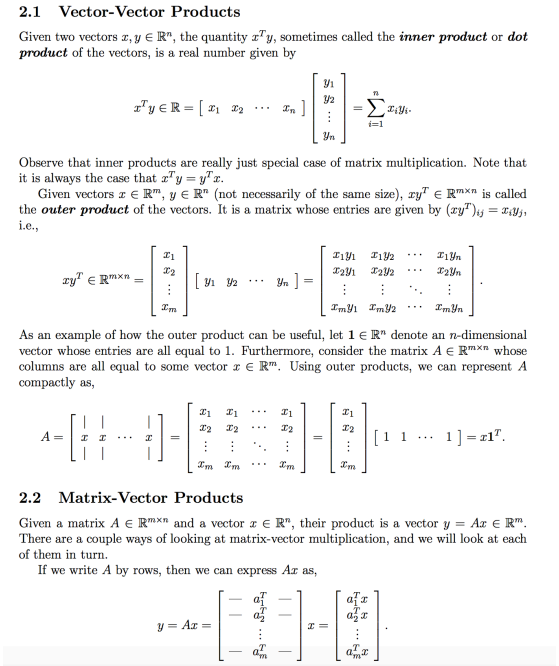
Predictions of 2nd ho

선형대수 리뷰

모두의연구소

http://cs229.stanford.edu/section/cs229-linalg.pdf

좀 더 깊은 리뷰를 보시려면 참고하세요



2.1 Vector-Vector Products

Given two vectors $x, y \in \mathbb{R}^n$, the quantity $x^T y$, sometimes called the ***inner product*** or ***dot product*** of the vectors, is a real number given by

$$x^T y \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i.$$

Observe that inner products are really just special case of matrix multiplication. Note that it is always the case that $x^T y = y^T x$.

Given vectors $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$ (not necessarily of the same size), $xy^T \in \mathbb{R}^{m \times n}$ is called the ***outer product*** of the vectors. It is a matrix whose entries are given by $(xy^T)_{ij} = x_i y_j$, i.e.,

$$xy^T \in \mathbb{R}^{m \times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}.$$

As an example of how the outer product can be useful, let $\mathbf{1} \in \mathbb{R}^n$ denote an n-dimensional vector whose entries are all equal to 1. Furthermore, consider the matrix $A \in \mathbb{R}^{m \times n}$ whose columns are all equal to some vector $x \in \mathbb{R}^m$. Using outer products, we can represent A compactly as,

$$A = \begin{bmatrix} | & | & & | \\ x & x & \cdots & x \\ | & | & & | \end{bmatrix} = \begin{bmatrix} x_1 & x_1 & \cdots & x_1 \\ x_2 & x_2 & \cdots & x_2 \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_m & \cdots & x_m \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} = x\mathbf{1}^T.$$

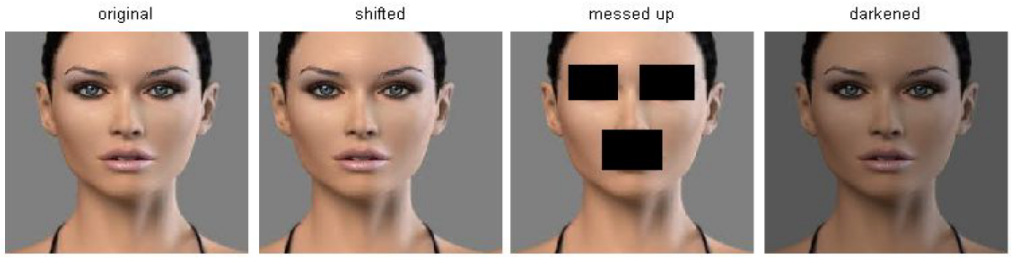
2.2 Matrix-Vector Products

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $x \in \mathbb{R}^n$, their product is a vector $y = Ax \in \mathbb{R}^m$. There are a couple ways of looking at matrix-vector multiplication, and we will look at each of them in turn.

If we write A by rows, then we can express Ax as,

$$y = Ax = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix} x = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}.$$

다시 돌아와서 …



(all 3 images have same L2 distance to the one on the left)

픽셀값들을 조합하여 단순히 각 레이블에 대한 점수를 부여하면 어떻까요? Score function?

Parametric approach

모두의연구소

- Score function

f( , W)

- 강아지 점수
- 고양이 점수

f( , W)

- 강아지 점수
- 고양이 점수

점수가 높은것으로 분류

Parametric approach

- Score function : Simple Linear Classifier

$$f(x, W) = Wx \quad 3{,}072 \times 1$$

32x32x3 영상 =
3,072 픽셀

1	155
2	200
...	...
3071	110
3072	78

x

$f(x, W)$

- 강아지 점수
- 고양이 점수

Parametric approach

- Score function : Simple Linear Classifier

$$\underset{2\times1}{f(x, W)} = \underset{3,072\times1}{Wx}$$

32x32x3 영상 =
3,072 픽셀

1	155
2	200
...	...
3071	110
3072	78

x

f(x, W)

- 강아지 점수
- 고양이 점수

0.3
0.7

Parametric approach

- Score function : Simple Linear Classifier

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

2x1 · 3,072x1 · 2x3,072

32x32x3 영상 = 3,072 픽셀

1	155
2	200
...	...
3071	110
3072	78

x

f(x, W)

0.2	0.1	...	0.3	0.7
0.7	0.8	...	0.9	0.4

- 강아지 점수
- 고양이 점수

0.3
0.7

Parametric approach

- Score function : Simple Linear Classifier

3,072x1

2x1 $f(x, W) = Wx + b$ 2x1

2x3,072

32x32x3 영상 =
3,072 픽셀

1	155
2	200
...	...
3071	110
3072	78

x

$f(x, W)$

- 강아지 점수
- 고양이 점수

0.3
0.7

0.2	0.1	...	0.3	0.7
0.7	0.8	...	0.9	0.4

Parametric approach

모두의연구소

- Score function : Simple Linear Classifier

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

3,072x1 (x) · 2x1 (f) · 2x3,072 (W) · 2x1 (b)

32x32x3 영상 = 3,072 픽셀

강아지 파라미터	0.2	0.1	...	0.3	0.7
고양이 파라미터	0.7	0.8	...	0.9	0.4

W

x

155
200
...
110
78

x

+

0.1
0.2

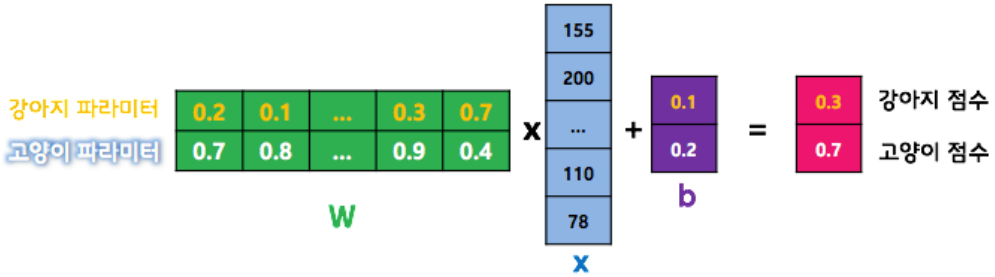
b

=

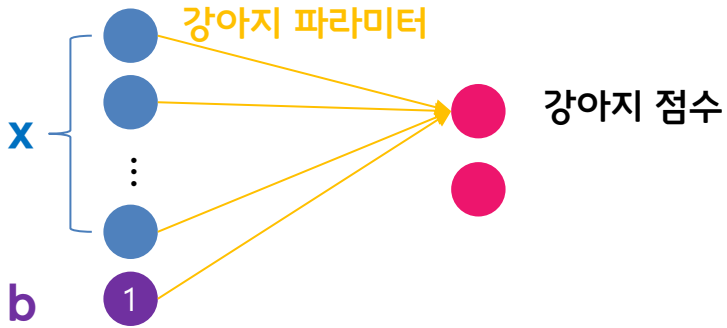
0.3	강아지 점수
0.7	고양이 점수

Parametric approach

- Score function : Simple Linear Classifier

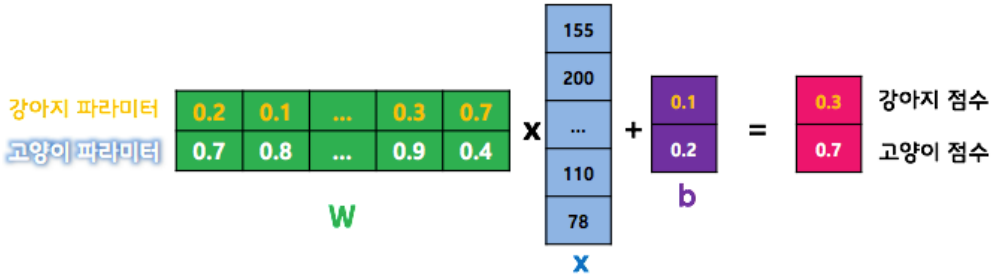


선형분류기의
뉴럴 네트워크 표현

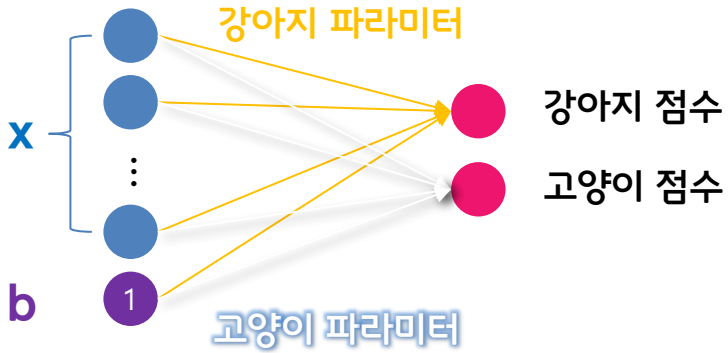


Parametric approach

- Score function : Simple Linear Classifier

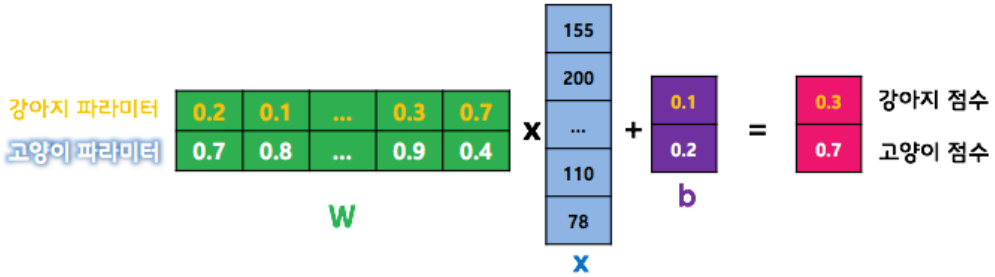


선형분류기의
뉴럴 네트워크 표현

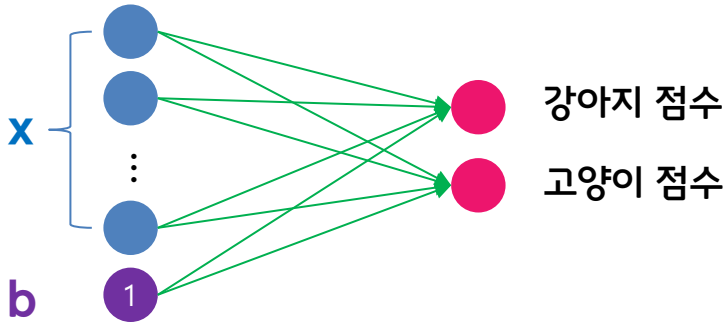


Parametric approach

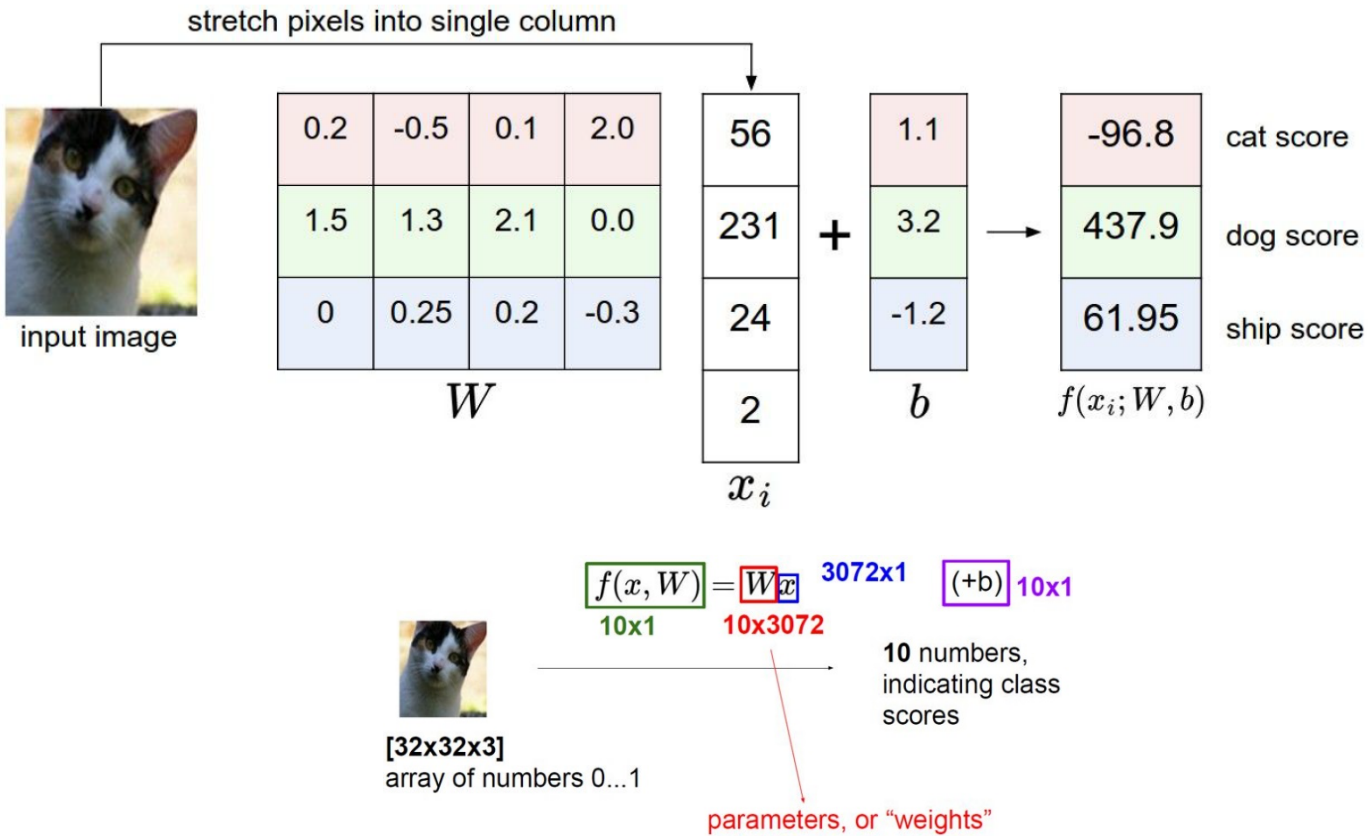
- Score function : Simple Linear Classifier



선형분류기의
뉴럴 네트워크 표현

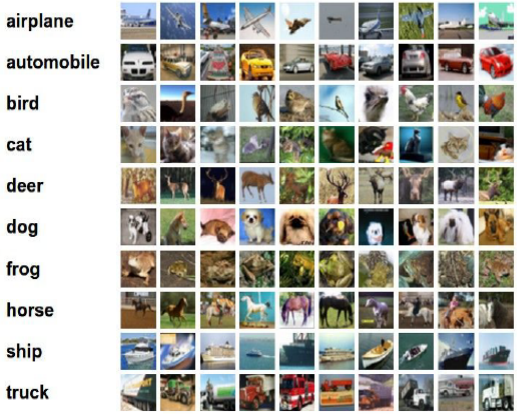


Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



W를 열어보면 ...

Interpreting a Linear Classifier



$$f(x_i, W, b) = Wx_i + b$$

Example trained weights of a linear classifier trained on CIFAR-10:

W를 어떤 템플릿으로 볼 수도 있겠군요...

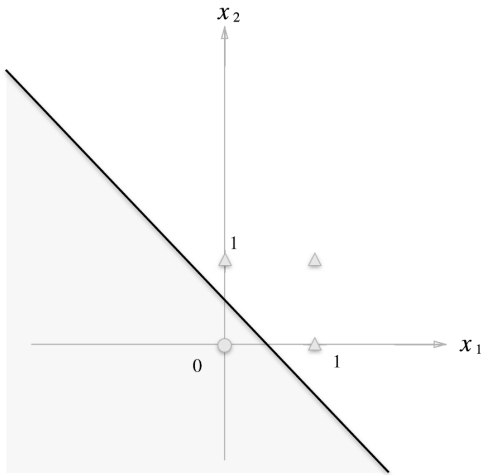
Decision Boundary의 해석

모두의연구소

OR 게이트 :

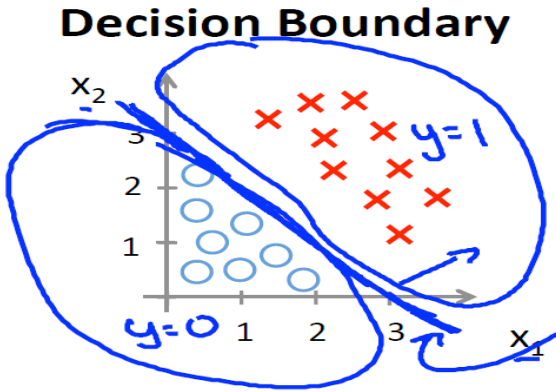
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

$$y = \begin{cases} 0 & (-0.5 + x_1 + x_2 \leq 0) \\ 1 & (-0.5 + x_1 + x_2 > 0) \end{cases}$$



Decision Boundary의 해석

모두의연구소



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

Decision boundary

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

$x_1 + x_2 \geq 3$

x_1, x_2

$h_\theta(x) = 0.5$

$x_1 + x_2 = 3$

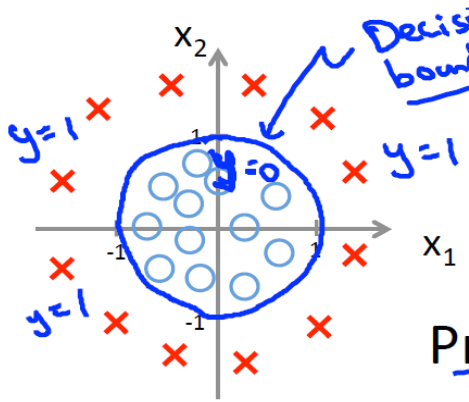
$x_1 + x_2 < 3$

$y = 0$

Decision Boundary의 해석

모두의연구소

Non-linear decision boundaries



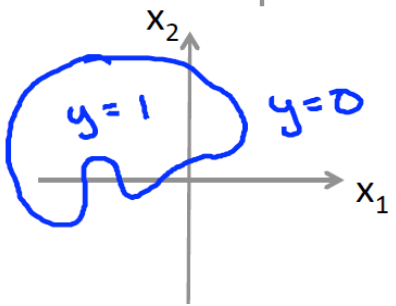
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

$x_1^2 + x_2^2 = 1$

$x_1^2 + x_2^2 \geq 1$

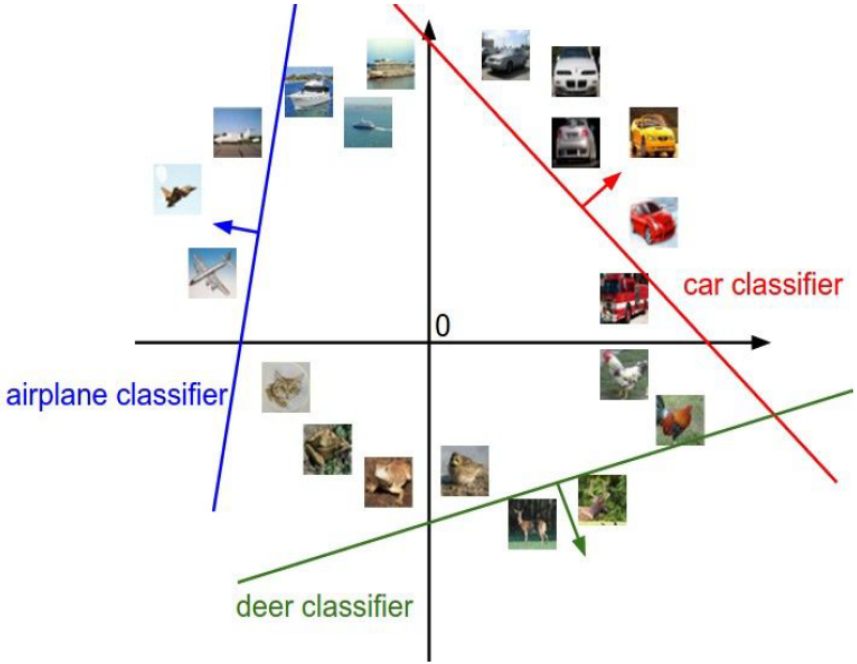


$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$

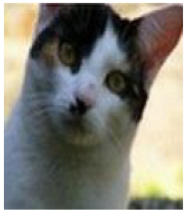
같은 방식으로 해석될 수 있습니다

모두의연구소

Interpreting a Linear Classifier



$$f(x_i, W, b) = Wx_i + b$$



[32x32x3]
array of numbers 0...1
(3072 numbers total)

지금까지 : Score function (Linear Classifier) 을 알아봤습니다

모두의연구소



$$f(x_i, W, b) = Wx_i + b$$

Example class scores for 3 images, with a random W:

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

Coming up:
- Loss function
- Optimization

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

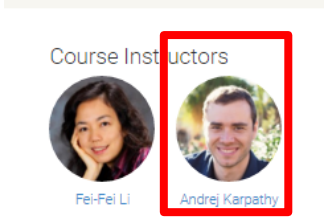
- Loss Function : 현재 내가 구성한 W가 좋은지 안좋은지 측정할 수 있게 해줍니다 (좋으면 0)

- Optimization : random W로 시작해서 Loss를 최소로 갖게 W를 변경해 줍니다

모두의연구소

기타 :

영상인식과 관련하여 우리는 어디까지 왔는가 ?



http://karpathy.github.io/2012/10/22/state-of-computer-vision/

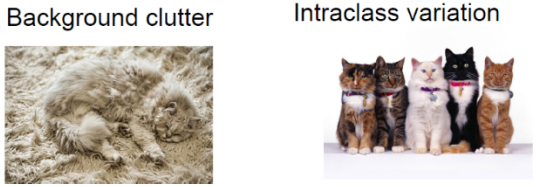
모두의연구소

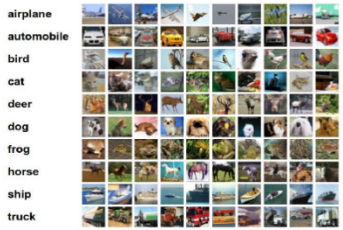
Loss Function

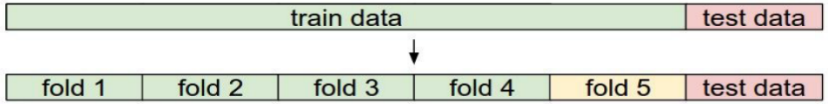
모두의연구소
박은수 Research Director

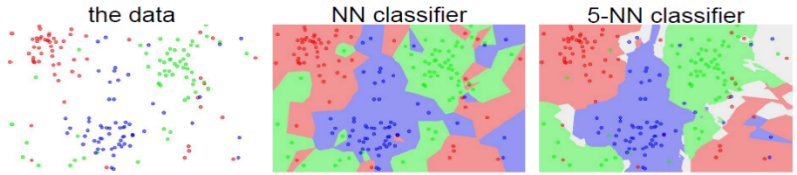
돌아보기 ...

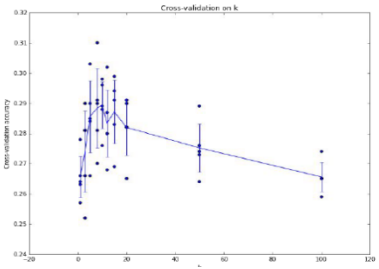

Camera pose
Illumination
Deformation
Occlusion


Background clutter
Intraclass variation


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck


train data
test data
fold 1
fold 2
fold 3
fold 4
fold 5
test data


the data
NN classifier
5-NN classifier


Cross-validation on k

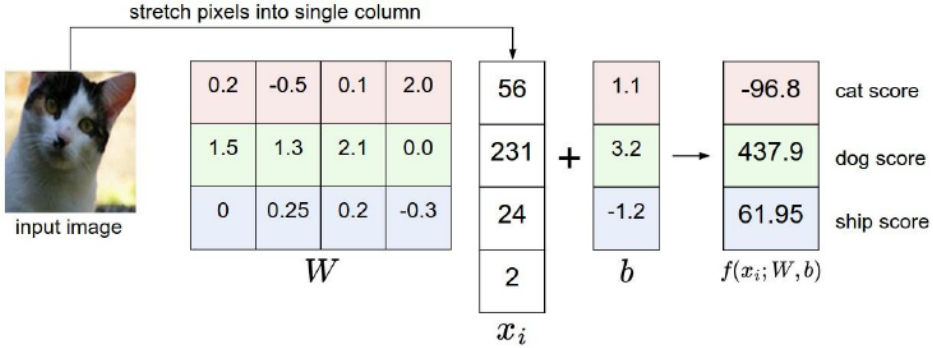
돌아보기 ...

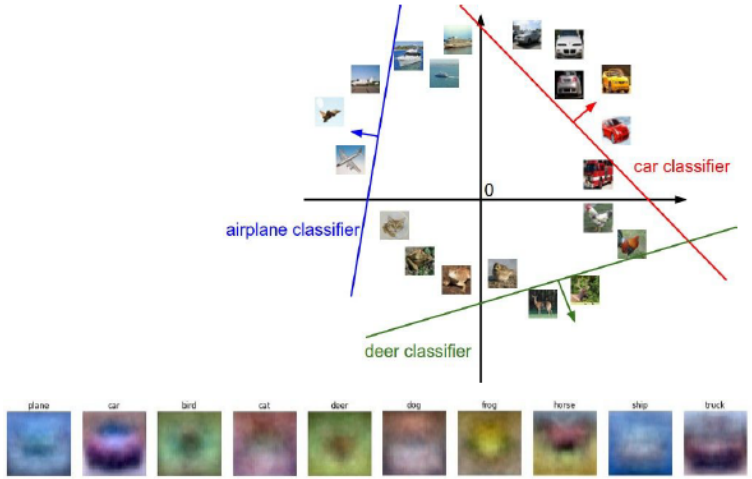
[32x32x3]
array of numbers 0...1
(3072 numbers total)

image parameters

$f(x, W)$

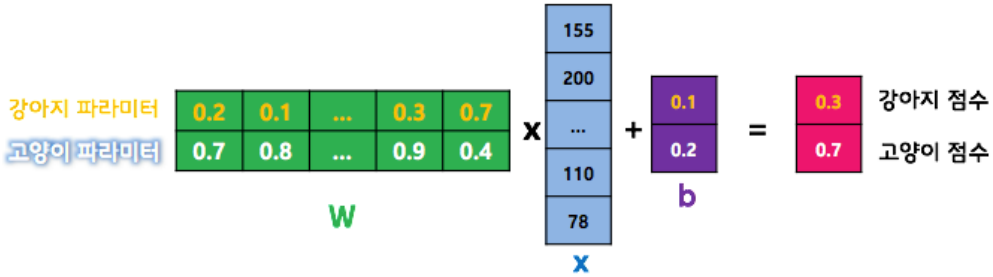
10 numbers, indicating class scores



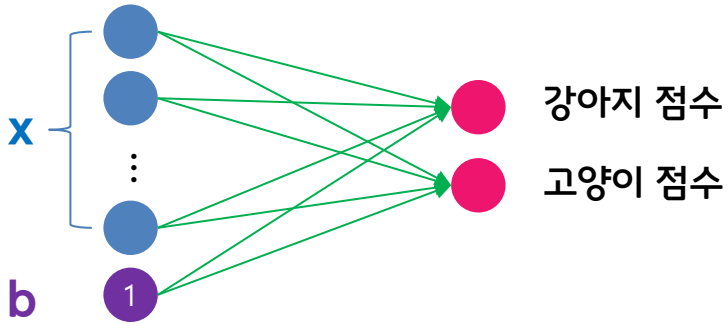


돌아보기 ...

- Score function



선형분류기의
뉴럴 네트워크 표현



모두의연구소

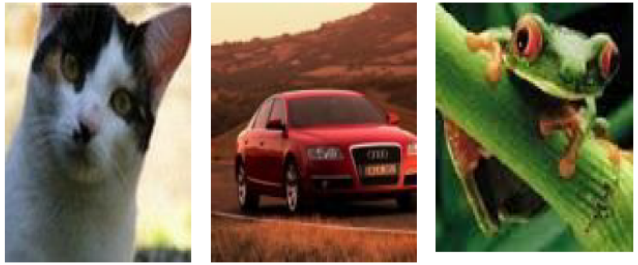
- 분류기의 구성
  - Score function
  - Loss function
  - Optimization

Loss Function

- 측정된 Score가 얼마나 안좋은지를 나타내는 하는 함수
- Optimization은 이 측정된 Loss function의 값을 줄이는 방법

이제 해야 할 것은

Score



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

1. 계산된 Score가 얼마나 후진지 알려줄 함수를 만들어야 합니다
   ⇒ Loss function

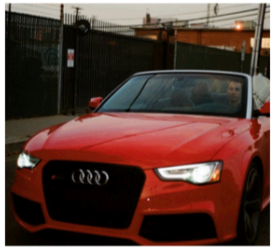
2. Loss function을 최소화 할수 있는 방법을 만들어야 합니다 (Optimization)

모두의연구소

첫번째로 살펴 볼 Loss function은 SVM Loss 입니다

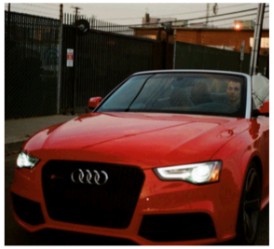
$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:


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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:


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

A **loss function** tells how good our current classifier is

Given a dataset of examples

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

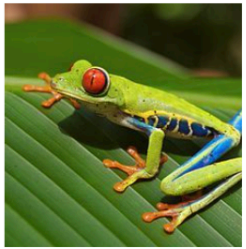
Where x_i is image and
y_i is (integer) label

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

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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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,

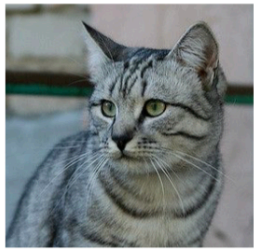
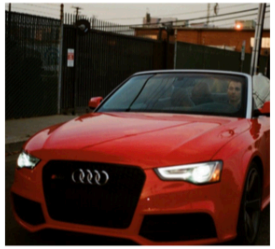
and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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:

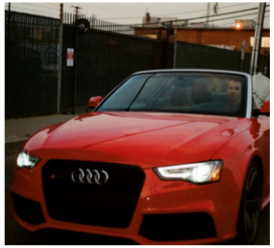
Given an example (x_i, y_i)
where x_i is the image and
where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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,

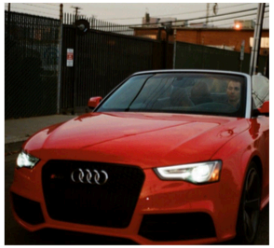
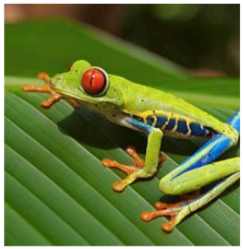
and using the shorthand for the scores vector: $s = f(x_i, W)$

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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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,

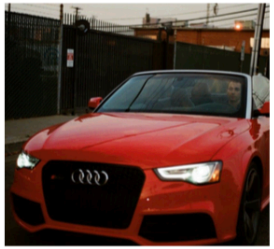
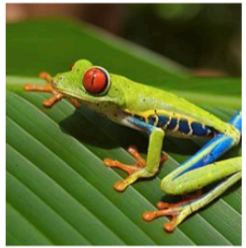
and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

= 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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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,

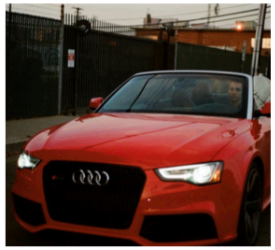
and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

= 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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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: $s = f(x_i, W)$

the SVM loss has the form:

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

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**

SVM (Hinge) Loss는 어떤 의미를 갖는가

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

i 번째 이미지의 Loss

정답이 아닌 것들만 합한다

정답이 아닌 score와 정답 score와의 차이

- 최소한 Loss가 0이 아닌 값을 가지려면 정답이 아닌 녀석들이 저 붉은 delta위치에는 오거나 정답 Score보다 크면 되겠네요

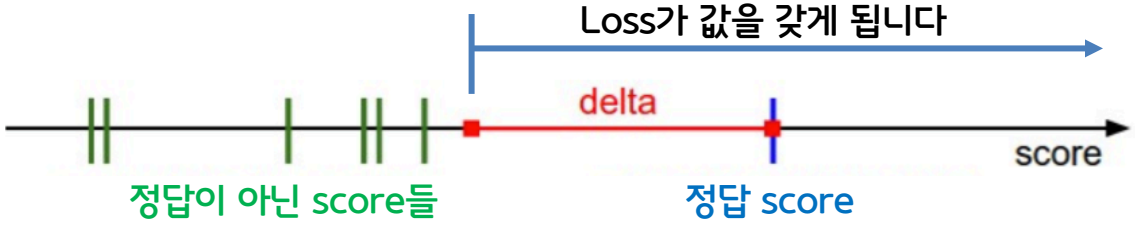
SVM (Hinge) Loss는 어떤 의미를 갖는가

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

i 번째 이미지 의 Loss

정답이 아닌 것들 만 합한다

정답이 아닌 score와 정답 score와의 차이



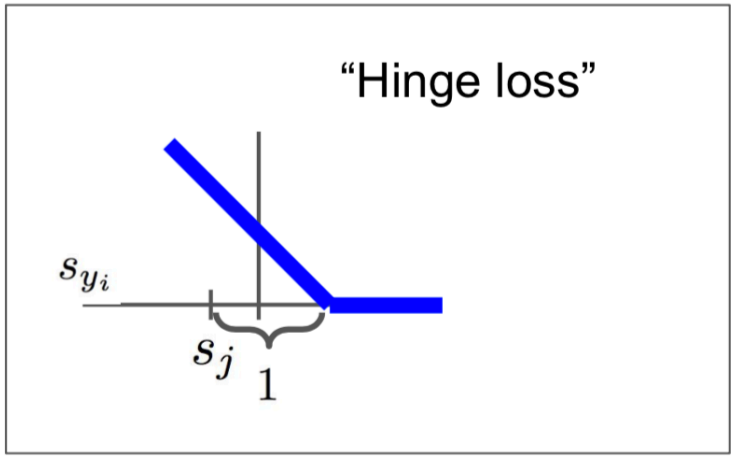
- 최소한 Loss가 0이 아닌 값을 가지려면 정답이 아닌 녀석들이 저 붉은 delta위치에는 오거나 정답 Score보다 크면 되겠네요

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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:



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

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

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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: $s = f(x_i, W)$

the SVM loss has the form:

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

Q: What happens to loss if car scores change a bit?

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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: $s = f(x_i, W)$

the SVM loss has the form:

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

Q2: what is the min/max possible loss?

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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: $s = f(x_i, W)$

the SVM loss has the form:

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

Q3: At initialization W is small so all s ≈ 0. What is the loss?

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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,

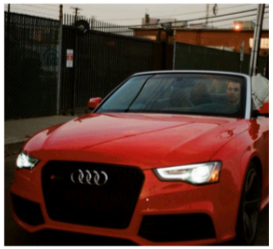
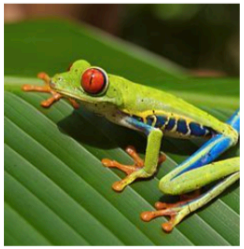
and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

Q4: What if the sum was over all classes? (including j = y_i)

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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,

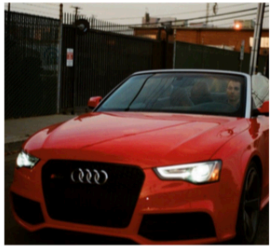
and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

Q5: What if we used mean instead of sum?

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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: $s = f(x_i, W)$

the SVM loss has the form:

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

Q6: What if we used

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

Multiclass SVM Loss: Example code

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

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

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

Score 함수

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

N 장의 이미지에 대한 Loss

이제 score를 만드는 법을 배웠고, 계산된 score를 통해
잘못 분류된 값들을 이용해서 Loss를 정의했다.

이제 Loss를 줄이는 W는 찾으면 된다

$$f(x, W) = Wx$$ Score 함수

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

N 장의 이미지에 대한 Loss

이제 score를 만드는 법을 배웠고, 계산된 score를 통해
잘못 분류된 값들을 이용해서 Loss를 정의했다.

이제 Loss를 줄이는 W는 찾으면 된다

그러나,

***L* = 0 이 되는 *W*를 찾았다면 이 값이 *W*가 고유한 값인가? 여러개가 될 수도 있지 않나?**

L = 0 인 W값은 무수하군요 ....

모두의연구소

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ 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	

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

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

그럼 값에 제한을 둡시다

너무 멀리서 찾지마 작은데서 찾아...
너무 커지지마 ...

\lambda = regularization strength
(hyperparameter)

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

In common use:

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)

Weight Regularization

L2 regularization: motivation

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

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

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

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

- Same Score
  - But L2 penalty?
    - w_1 : 1
    - w_2 : 0.25

Less Loss

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

L2 regularization: small and widely distributed W

→ **Use all features with small weights**

그런데 저 1은 어디서 나온건가요?

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

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$

1을 일단 델타로 두고 버그를 쓸어 담아 봅시다

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$

Do we have to cross-validate both Δ and λ ?

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W)$$

모두의연구소

$$\begin{aligned}
&\max(0, \delta f + \Delta) \\
\rightarrow &\max(0, \delta f + \Delta_2) && \text{if } \Delta \rightarrow \Delta_2 \\
\rightarrow &\max(0, \frac{\Delta_2}{\Delta} \delta f + \Delta_2) && \text{then scale the weights by } \frac{\Delta_2}{\Delta} \\
= &\max(0, \frac{\Delta_2}{\Delta} (\delta f + \Delta)) \\
= &\frac{\Delta_2}{\Delta} \max(0, \delta f + \Delta)
\end{aligned}$$

Softmax Loss

모두의연구소

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

where $s = f(x_i; W)$

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

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

강아지 점수 2.3

고양이 점수 -1.2

---

in summary: $L_i = -\log(\frac{e^{sy_i}}{\sum_j e^{s_j}})$

Softmax Loss

모두의연구소

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

강아지 점수	2.3	exp →	9.97
고양이 점수	-1.2		0.3

전부다 양수가 됨

Softmax Loss

모두의연구소

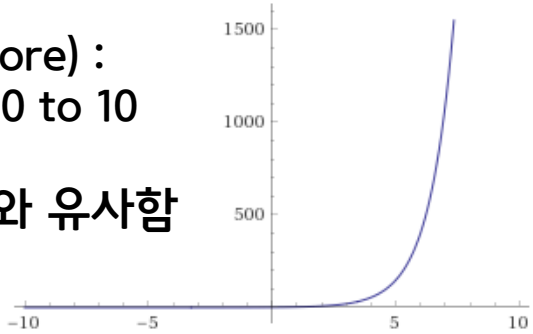
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

강아지 점수	2.3	exp	9.97
고양이 점수	-1.2		0.3

전부다 양수가 됨

exp(score) :
from -10 to 10

Max함수와 유사함



Softmax Loss

모두의연구소

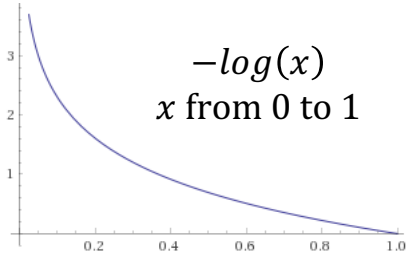
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

강아지 점수	2.3	exp →	9.97	normalize →	0.97
고양이 점수	-1.2		0.3		0.03
			전부다 양수가 됨		**확률**

Softmax Loss

모두의연구소

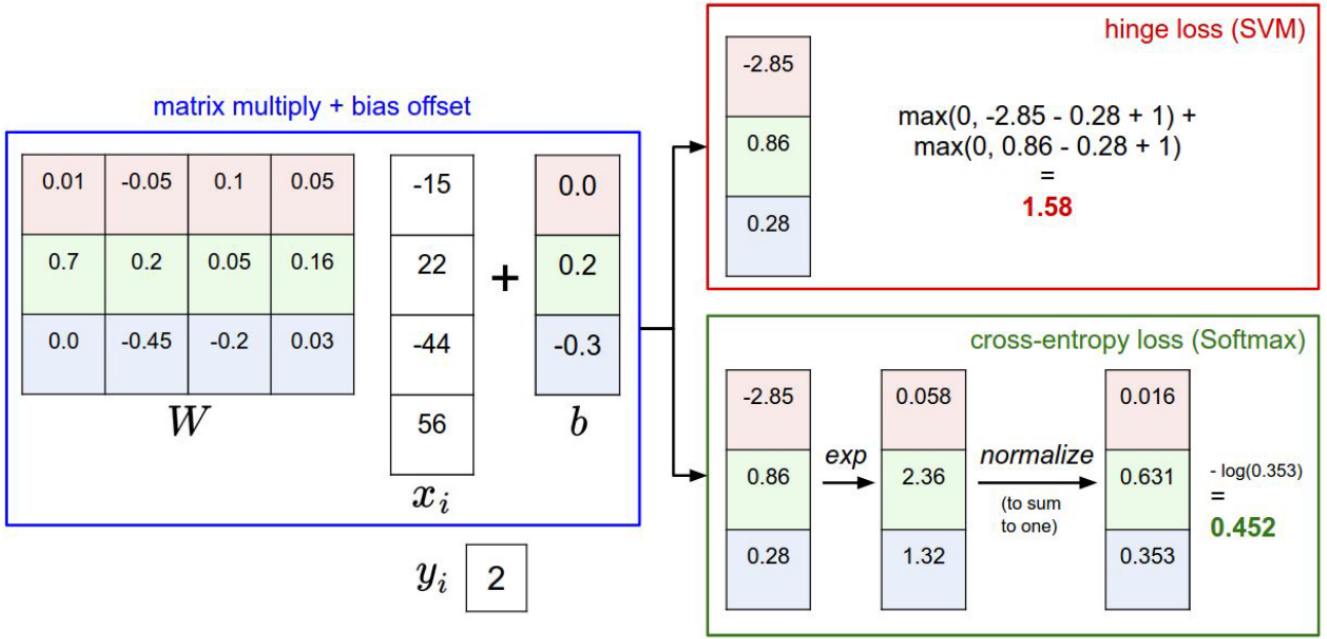
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$



강아지 점수	2.3	exp →	9.97	normalize →	0.97
고양이 점수	-1.2		0.3		**0.03**
			전부다 양수가 됨		**확률**

$L_i = -log(0.03) = 3.5$

Loss Function : Softmax Loss

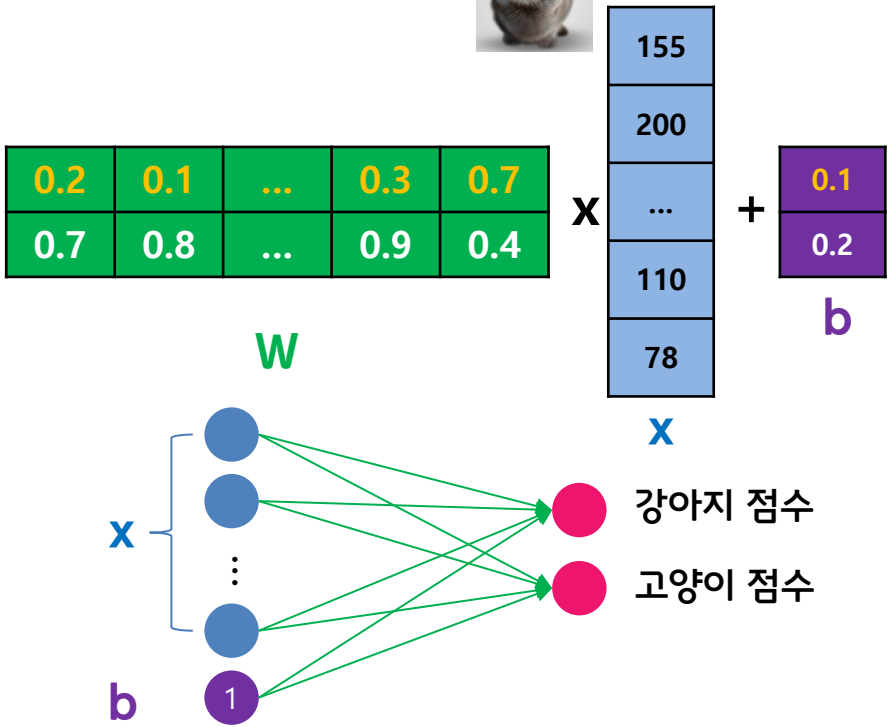


강아지와 고양이 분류해보기

모두의연구소

- 분류기의 구성
  - Score function
  - Loss function

고양이가 입력이면 고양이 점수가 높아야 함



강아지와 고양이 분류해보기

모두의연구소

- 분류기의 구성
  - Score function
  - Loss function

0.2	0.1	...	0.3	0.7
0.7	0.8	...	0.9	0.4

W

x

155
200
...
110
78

x

+

0.1
0.2

b

=

Cross-entropy loss (Softmax)

강아지 점수	2.3	exp	9.97	normalize	0.97	강아지 확률
고양이 점수	-1.2		0.3		0.03	고양이 확률

$-log(0.03) = 3.5$

현재의 분류기는 3.5만큼 안 좋음. 이 loss 값을 줄이는게 목표

Loss Function : Softmax Loss

의연구소

Softmax vs. SVM

- Interpreting the probabilities from the Softmax

$$[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]$$

suppose the weights W were only half as large:

$$[0.5, -1, 0] \rightarrow [e^{0.5}, e^{-1}, e^0] = [1.65, 0.37, 1] \rightarrow [0.55, 0.12, 0.33]$$

Loss Function : Softmax Loss

모두의연구소

Softmax vs. SVM

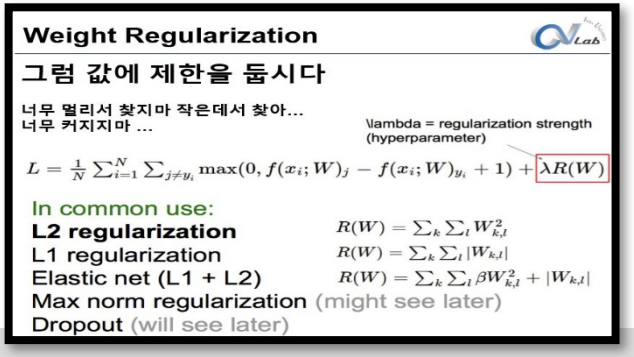
- Interpreting the probabilities from the Softmax

$$[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]$$

suppose the weights W were only half as large:

$$[0.5, -1, 0] \rightarrow [e^{0.5}, e^{-1}, e^0] = [1.65, 0.37, 1] \rightarrow [0.55, 0.12, 0.33]$$

질문 : 만약 Softmax에서 Regularization 파라미터가 너무 너무 크면 어떻게 될 것 같나요?



Softmax vs. SVM

Softmax vs. SVM

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

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

---

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

질문 : Score값에 작은 노이즈가 발생했습니다. 그래서 각 Score값에 작은 변화들이 생겼습니다. 이때 Softmax 와 SVM Loss는 각각 어떨까요?

요약

- 지금까지 W라는 파라미터를 이용한 Parametric 접근의 선형 분류를 살펴보았습니다
- Score 함수를 선형으로 정의해서 살펴보았습니다
- Loss 함수를 살펴보았습니다 (SVM / Softmax)

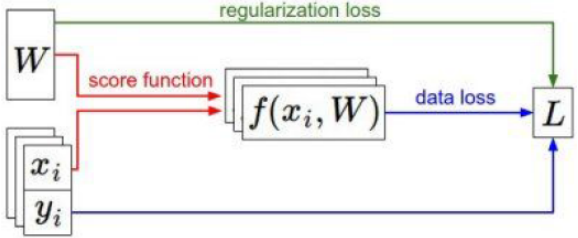
Softmax

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

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



이제 Loss를 최소로 하는 W를 찾는 방법만 남았습니다 :

요약

모두의연구소

- 분류기의 구성
  - Score function
  - Loss function
  - Optimization

이제 Loss를 최소로 하는 W를 찾는 방법만 남았습니다 :
다음 시간에

박은수 책임연구원

E-mail : es.park@modulabs.co.kr