LINEAR CLASSIFICATION

HTTP://CS231N.GITHUB.IO/LINEAR-CLASSIFY/

HAYEONG LEE

HY-KIERA

GITHUB.COM/HY-KIERA

Image Classification Problem

The task of assigning a single label to an image from a fixed set of categories.

e.g. KNN(K-Nearest Neighbor) classfier

: labels images by comparing them to (annotated) images from the training set.

-> must ’remember’ all of the training data and store it

-> classifying a test image is expensive

Parameterized mapping from images to label scores

Keyword

score function : maps the pixel values of an image to confidence scores for each class.

$$f : R^D \mapsto R^K$$

A training dataset of images $x_i \in R^D$

Each associated with a label $y_i \in 1\ldots K$

$\iota = 1\ldots N$

Parameterized mapping from images to label scores

Linear Classifier

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

x_i : [D x 1]

W : [K x D]

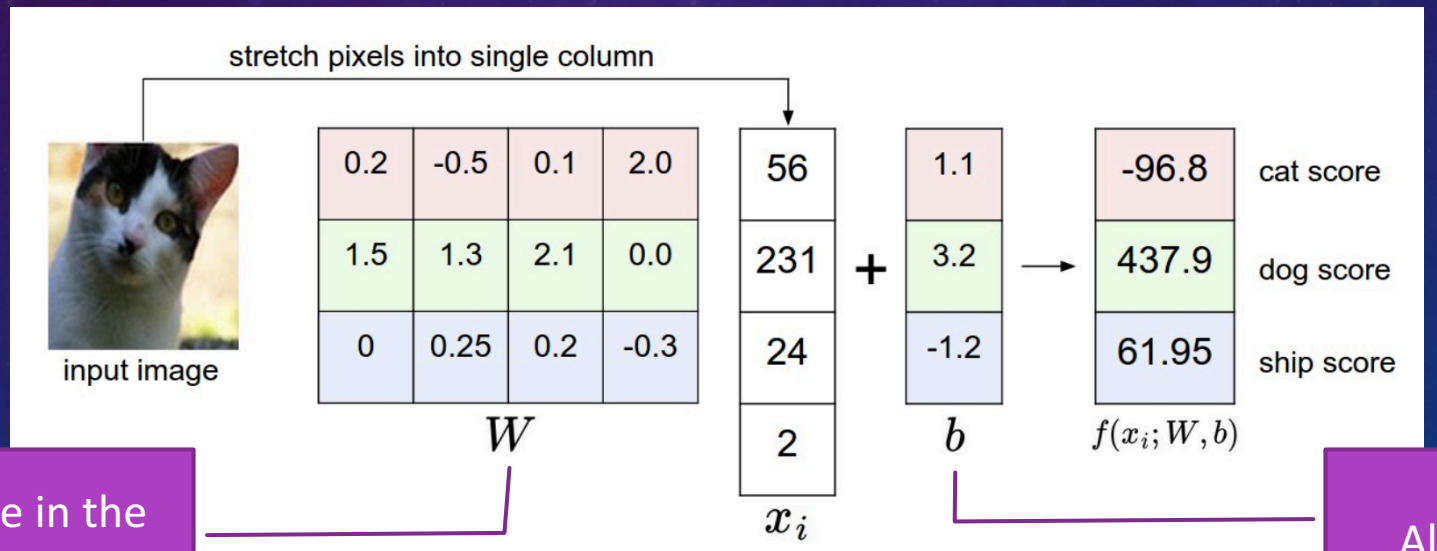
b : [K x 1]

Interpreting a linear classifier

Linear Classifier

Computes the score of a class as a weighted sum of all of its pixel values across all 3 of its color channels.

The function has the capacity to like or dislike certain colours at certain position in the images.



Corresponding line in the pixel space will rotate in different directions

Allow our classifiers to translate the line

Interpreting a linear classifier

Interpretation of linear classifiers as template matching

Each row of W corresponds to a template(prototype) for one of the classes.

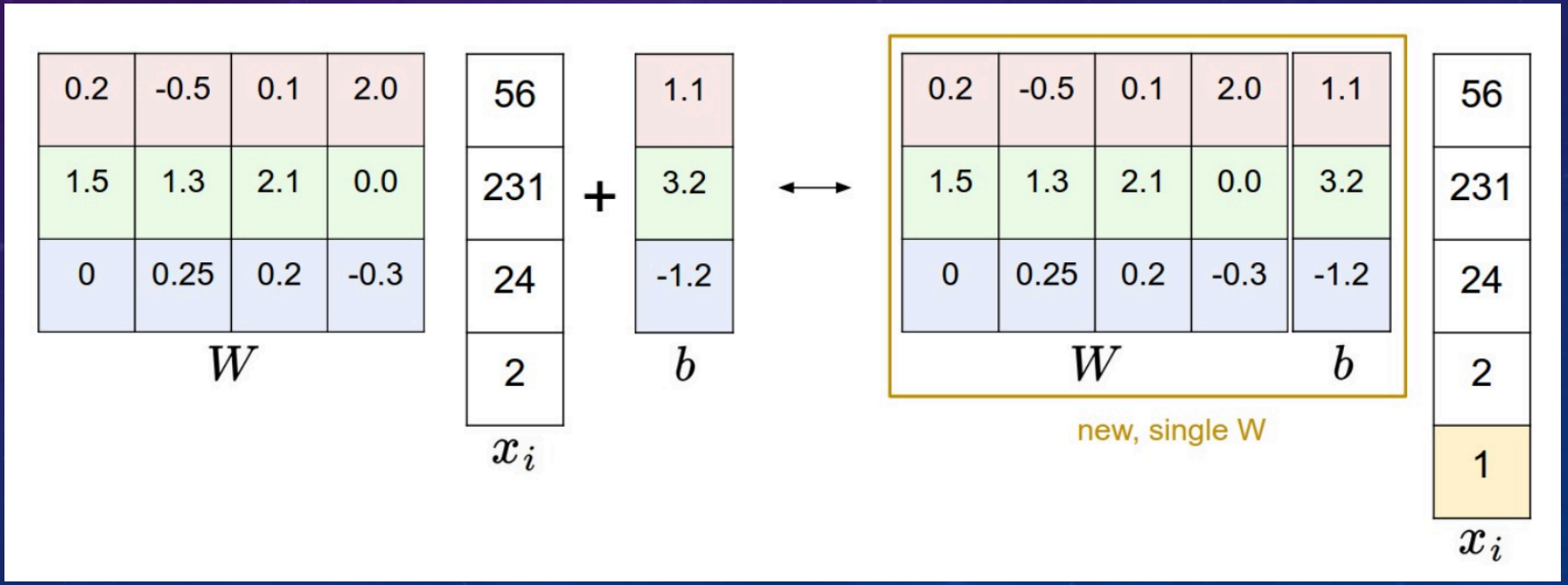
The score of each class for an image is then obtained by comparing each template with the image using an ‘inner product’ one by one to find the one that ‘fits’ best

->Linear classifier ’merges’

Interpreting a linear classifier

Bias trick

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



Loss function

Keyword

loss function : quantifies the agreement btw the predicted scores and the ground truth labels.

Multiclass Support Vector Machine loss (SVM loss)

Loss function

SVM loss

SVM ‘wants’ the correct class for each image to a have a score higher than the incorrect classes by some fixed margin Δ

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

The score for the j-th class is the j-th elements : $s_j = f(x_i, \mathrm{W})_j$

Loss function

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

e.g.

3 classes receive the score s = [13, -7, 11]

First class is the true class (i.e. y_i = 0)

Δ = 10

$$L_i = \max(0, -7 - 13 + 10) + \max(0, 11 - 13 + 10)$$

⇒ SVM loss function 'wants' the score of the correct class y_i to be larger than the incorrect class scores by at least by Δ

If this is not the case, we will accumulate loss

How much higher the difference would have to be to meet the margin

Loss function

SVM loss

-> linear score functions

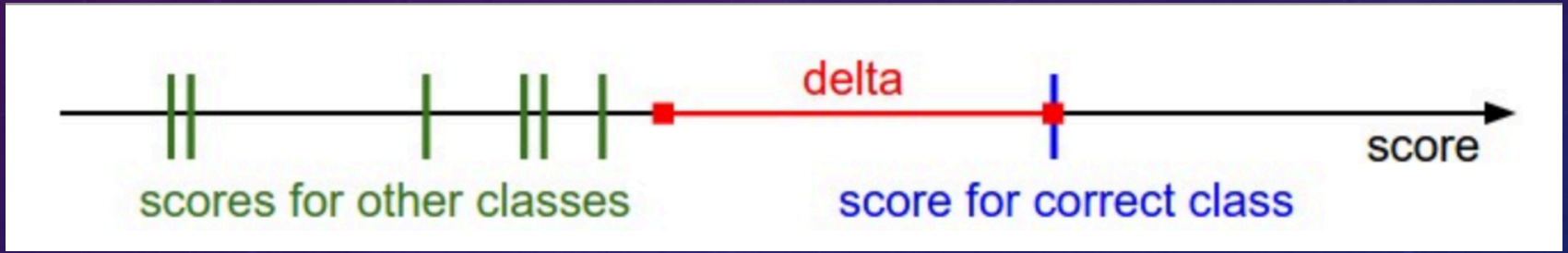
$$L_i = \sum_{j \neq y_i} \max(0, w_j^T x_i - w_{y_i}^T x_i + \Delta)$$

w_j : the j-th row of W reshaped as a column

hinge loss : the threshold at zero max(0, -) function

Loss function

SVM loss



The Multiclass Support Vector Machine ‘wants’ the score of the correct class to be higher than all other scores by at least a margin of delta.

Loss function

bug

Suppose that we have a dataset and a set of parameters W that correctly every example.

All scores are so that all the margins are met, and L_i = 0 for all i.

However, set of W is ‘not’ necessarily unique.

(i.e. there might be many similar W that correctly classify the examples)

-> If some parameters W correctly classify all examples (L_i = 0 for all i)

Then any multiple of these parameters λW (λ > 1) will also give zero

Loss function

Regularization – loss function with a regularization penalty R(W)

L2 norm : discourage large weights through an elementwise quadratic penalty over all parameters

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

$$L = \frac{1}{N} \sum_i L_i + \lambda R(W)$$

N : the # of training examples

λ : hyper-parameter
(usually determined by cross-validation)

Data loss
(the average loss L_i over all examples)

Regularization loss
(regularization function is not a function of data, it is only based on the weights.)

Loss function

Regularization $$L = \frac{1}{N}\sum_{i}\sum_{j \neq y_i}[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta] + \lambda \sum_{k}\sum_{l} W_{k,l}^2$$

Penalizing large weights tends to improve generalization

∵ no input dimension can gave a very large influence on the scores all by itself.

Input vector $x = [1, 1, 1, 1]$

Two weight vector $W_1 = [1, 0, 0, 0]$, $W_2 = [0.25, 0.25, 0.25, 0.25]$

$W_1^T x = W_2^T x = 1$

But, L2 penalty of W_1 is 1.0, W_2 is 0.25

(W_2 is lower regularization loss)

=> Take into account all input dimensions to small amounts rather than a few input dimensions and very strongly.

∵ W_2 is smaller and more diffuse

cf) bias do not control the strength of influence of an input dimension

Delta(Δ)

Set Δ = 1.0 in all cases -> safely

Hyper-parameter Δ, λ

-> control the same tradeoff (btw the data loss and the regularization loss in the objs).

The magnitude of the weights W

-> direct effect on the scores(the difference)

W↓ -> the score difference ↓

W↑ -> the score difference ↑

∴ the exact value of the margin btw the scores (e.g. Δ = 1, or Δ = 100) is in some sense meaningless because the weights can shrink or stretch the differences arbitrarily.

Binary Support Vector Machine

$$L_i = \mathrm{C}\max(0, 1 - y_i W^T x_i) + R(W)$$

C : hyper-parameter
$y_i \in \{-1, 1\}$

- The binary SVM as a special case when there are only two classes.
- C in this formulation and λ in our formulation control the same tradeoff (C $\propto \frac{1}{\lambda}$)

Softmax classifier

Softmax classifier : its generalization to multiple classes

- More intuitive output(normalized class probabilities)
- A probabilistic interpretation

cf) SVM - $f(x_i, W)$: output

(uncalibrated and
possibly difficult to interpret)

$$f(x_i; W) = Wx_i$$

-> interpret as the unnormalized log probabilities for each class

-> replace the hinge loss with a *cross-entropy-loss*

Softmax classifier

Cross-entropy-loss

$$L_i = -\log\left(\frac{e^{f}_{y_i}}{\sum_j e^{f_j}}\right) \qquad \left(L_i = -f_{y_i} + \log \sum_j e^{f_j}\right)$$

f_j : j-th element of the vector of class scores f

The full loss for the dataset : the mean of L_i over all training examples together with a regularization term $R(W)$

Softmax classifier

Softmax function

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Softmax function

: A vector of arbitrary real-valued scores (in z) and

squashes it to a vector of value btw zero and one that sum of one.

Softmax classifier

Cross-entropy btw a “true” distribution p, an estimated distribution q

$$H(p,q) = -\sum_x p(x) \log q(x) \qquad q = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

The Softmax classifier is hence minimizing the cross-entropy btw the estimated class prob and the “true” distribution, which in this interpretation is the distribution where all prob mass is on the correct class(i.e. $p = [0, \dots 1, \dots, 0]$ contains a single 1 at the y_i-th position)

Softmax classifier

Probabilistic interpretation

e^x -> (unnormalized) prob

$$P(y_i \mid x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

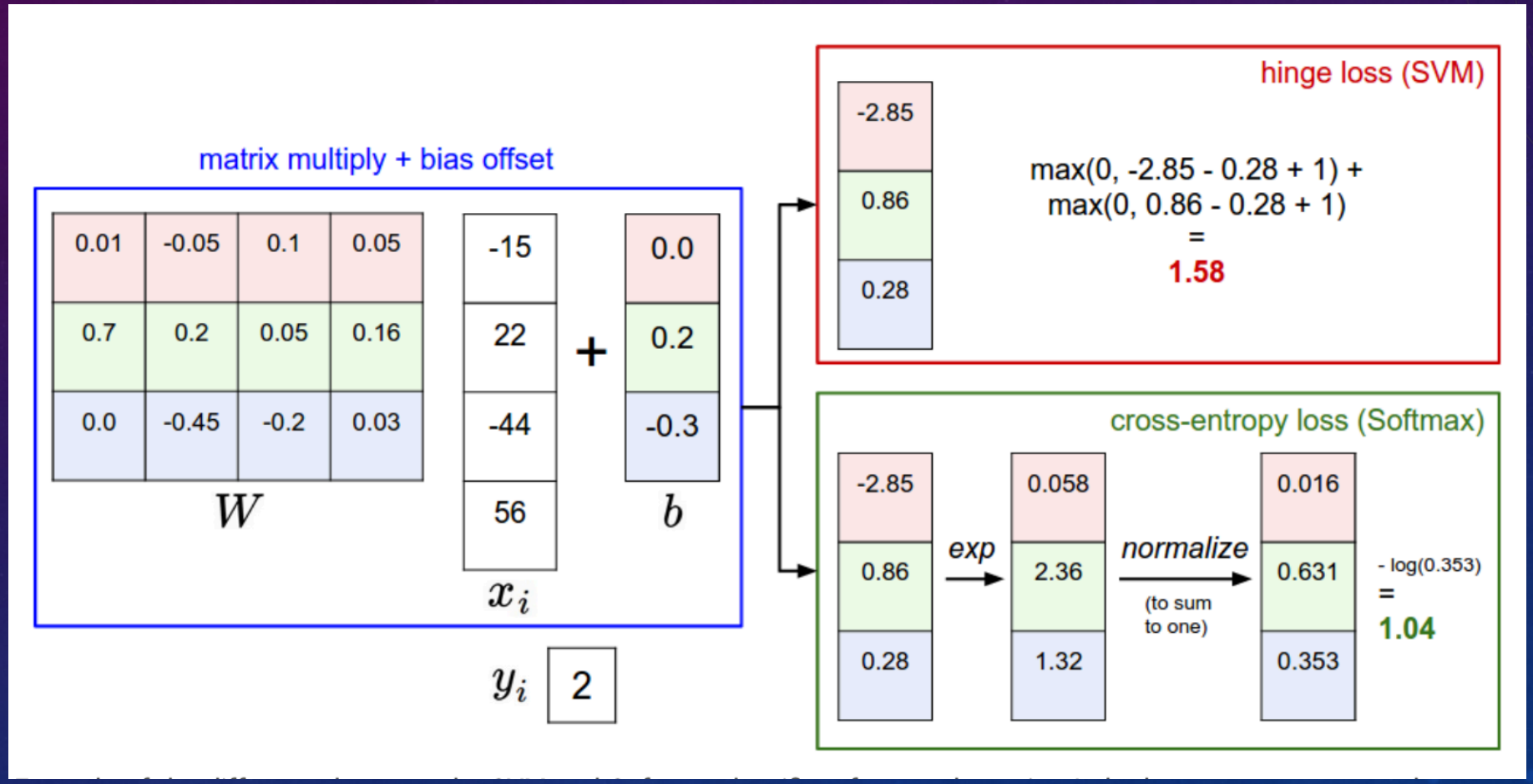
-> normalization so that the prob sum to one

P : the (normalized) prob assigned to the correct label y_i given the image x_i and parameterized by W

In probabilistic interpretation,

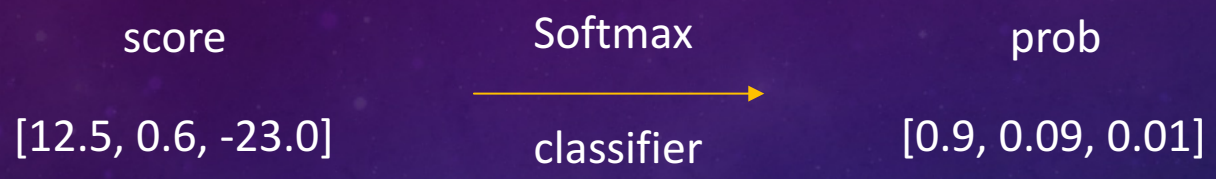
Minimize the negative log likelihood of the correct class, which can be interpreted as performing MLE

SVM vs Softmax



SVM vs Softmax

- Softmax classifier provides “probabilities” for each class



probabilities

How peaky or diffuse these prob are depends directly on the regularization strength λ

If λ was higher, W would be penalized more and would lead to smaller weights

e.g)

Weights : [0.5, -1, 0] -> $[e^{0.5}, e^{-1}, e^{0}]$ = [1.65, 0.37, 1] -> [0.55, 0.12, 0.33]

SVM vs Softmax

- SVM and Softmax are usually comparable in practice

SVM -> more “local” objective (dose not care about the details of the individual scores)

e.g) [10, -100, -100] or [10, 9, 9] -> margin of 1 is satisfied
-> loss of zero

cf) Softmax classifier -> accumulate a much higher loss for the scores [10, 9, 9] than for [10, -100, -100]

SVM vs Softmax

- SVM and Softmax are usually comparable in practice

Softmax classifier

: the correct class could always have a higher prob and the incorrect classes always a lower prob and the loss would always get better.

SVM

: it is happy once the margins are satisfied and it does not micromanage the exact scores beyond this constraint.