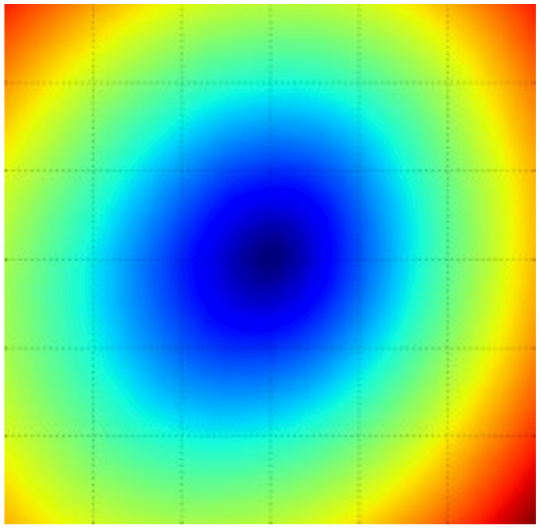
CS4670/5670: Computer Vision

Kavita Bala

Lecture 28: Recognition Basics 2



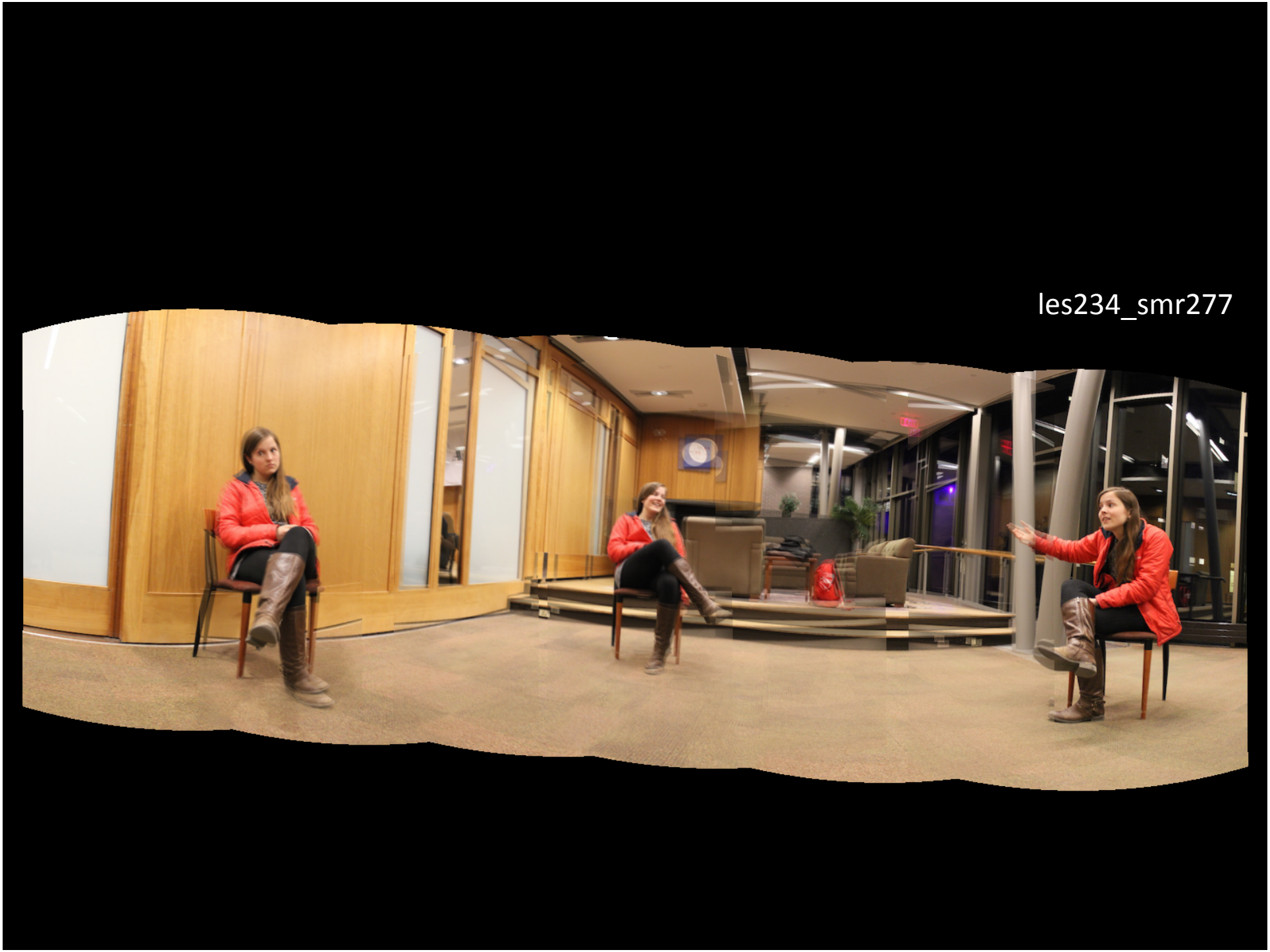
Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

PA 3 Artifact Results

Runners Up

jam798_mhw66

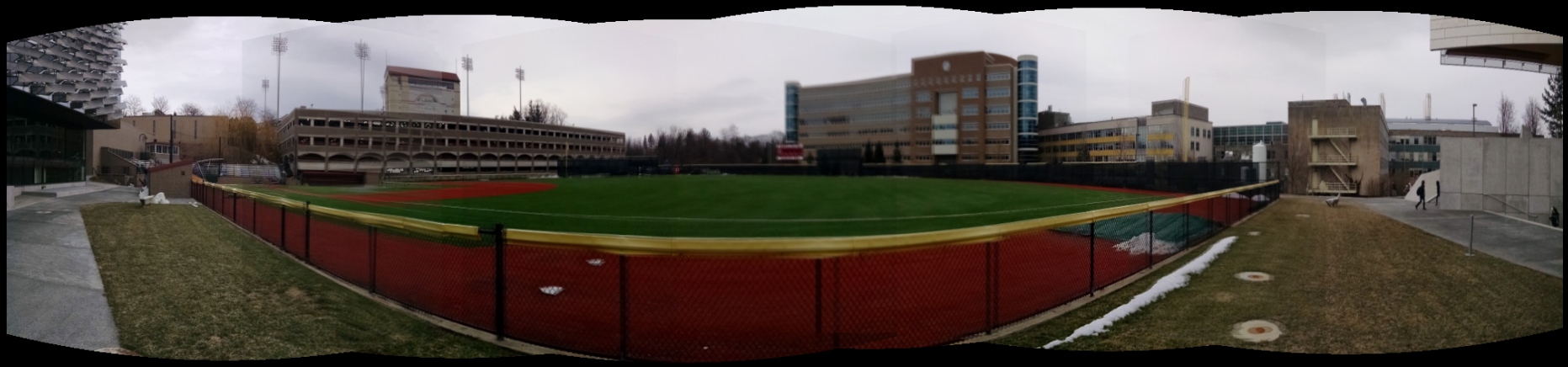



les234_smr277

lg399_zl256



djr244_gtw24



Third Place

gt224_gx33



Second Place

asn67_mu72



First Place

cl839_jm2232



Today

- Score function and loss function
- Optimization

- Next time: CNNs

Data-driven approach

- Collect a database of images with labels
- Use ML to train an image classifier
- Evaluate the classifier on test images



Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Score function





class scores

Slides from Andrej Karpathy and Fei-Fei Li
http://vision.stanford.edu/teaching/cs231n/

Linear Classifier

(assume CIFAR-10 example so
32 x 32 x 3 images, 10 classes)

data (image)
[3072 x 1]

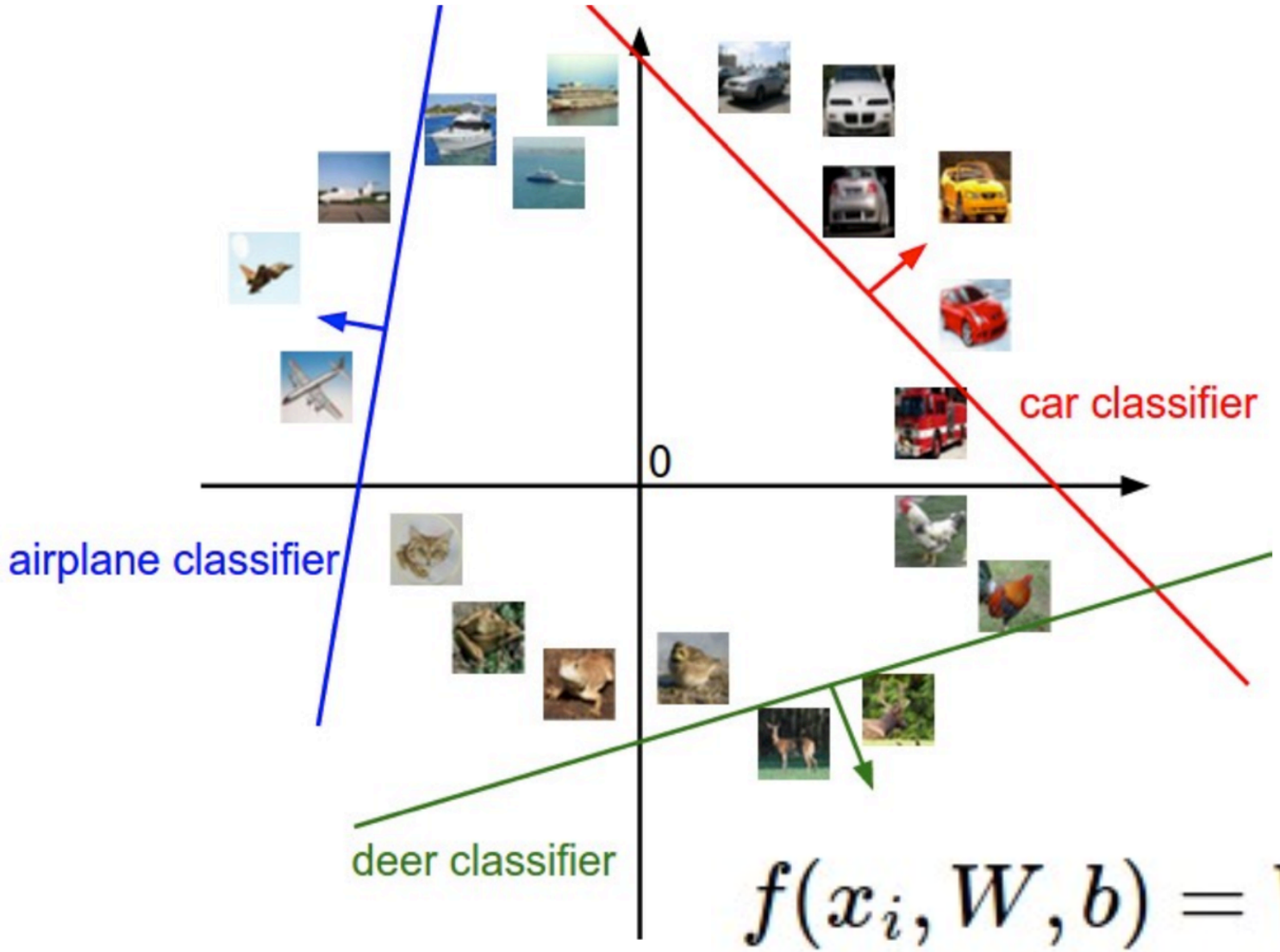
$$f(x_i, W, b) = Wx_i + b$$

class scores
[10 x 1]

weights
[10 x 3072]

bias vector
[10 x 1]

Geometric Interpretation



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

Loss function, cost/objective function

- Given ground truth labels (yi), scores f (xi, W)
  - how unhappy are you with the scores?

- Loss function or objective/cost function

- Want to minimize the loss function

Loss function, cost/objective function

- Given ground truth labels (yi) and scores f(xi, W)
  - how unhappy are you with the scores?

$$f(x_i, W) = [13, -7, 11]$$

$$y_i = 0$$



Intuition


delta
score
scores for other classes
score for correct class

Loss fn: Multi-class SVM loss

- Given ground truth labels (yi) and scores f(xi, W)
  - how unhappy are you with the scores?

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

(One possible generalization of Binary Support Vector Machine to multiple classes)

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

Loss fn: interpretation

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

loss due to example i

sum over all incorrect labels

difference between the correct class score and incorrect class score

Example: loss

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

e.g. 10

$f(x_i, W) = [13, -7, 11]$

$y_i = 0$

---

loss = ?

Example: loss

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

e.g. 10

$f(x_i, W) = [13, -7, 11]$

$y_i = 0$

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

Need more..

- Regularization
  - Ambiguity: W is not unique
  - If loss is 0, k W also has 0 loss
  - Add a regularization penalty
    - Try to keep the weights low

Important: regularization

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

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

Regularization strength

Determine by cross-validation

Example: regularization

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

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

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

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

Final loss function

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

Can set delta to 1

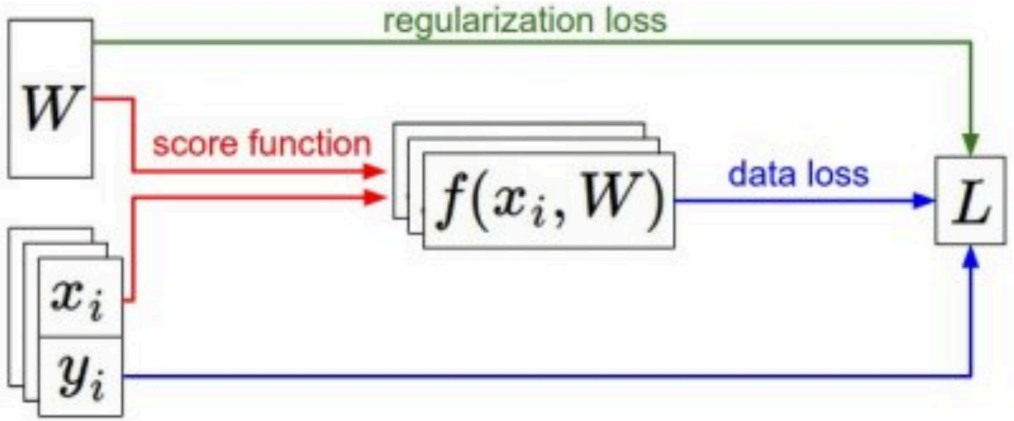
Summary

1. Score function

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

2. Loss function

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



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

Summary

- Have score function and loss function
  - Will generalize the score function

- Find W and b to minimize loss

Minimizing loss

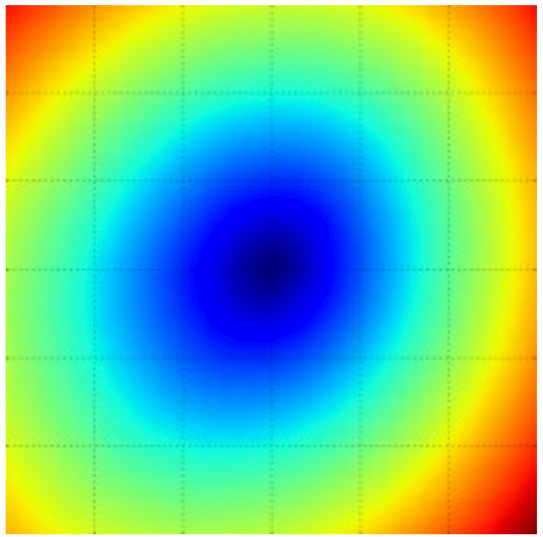
- Find W and b to minimize loss
  - Very high dimensional space

Visualizing the (SVM) loss function

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

the full data loss:

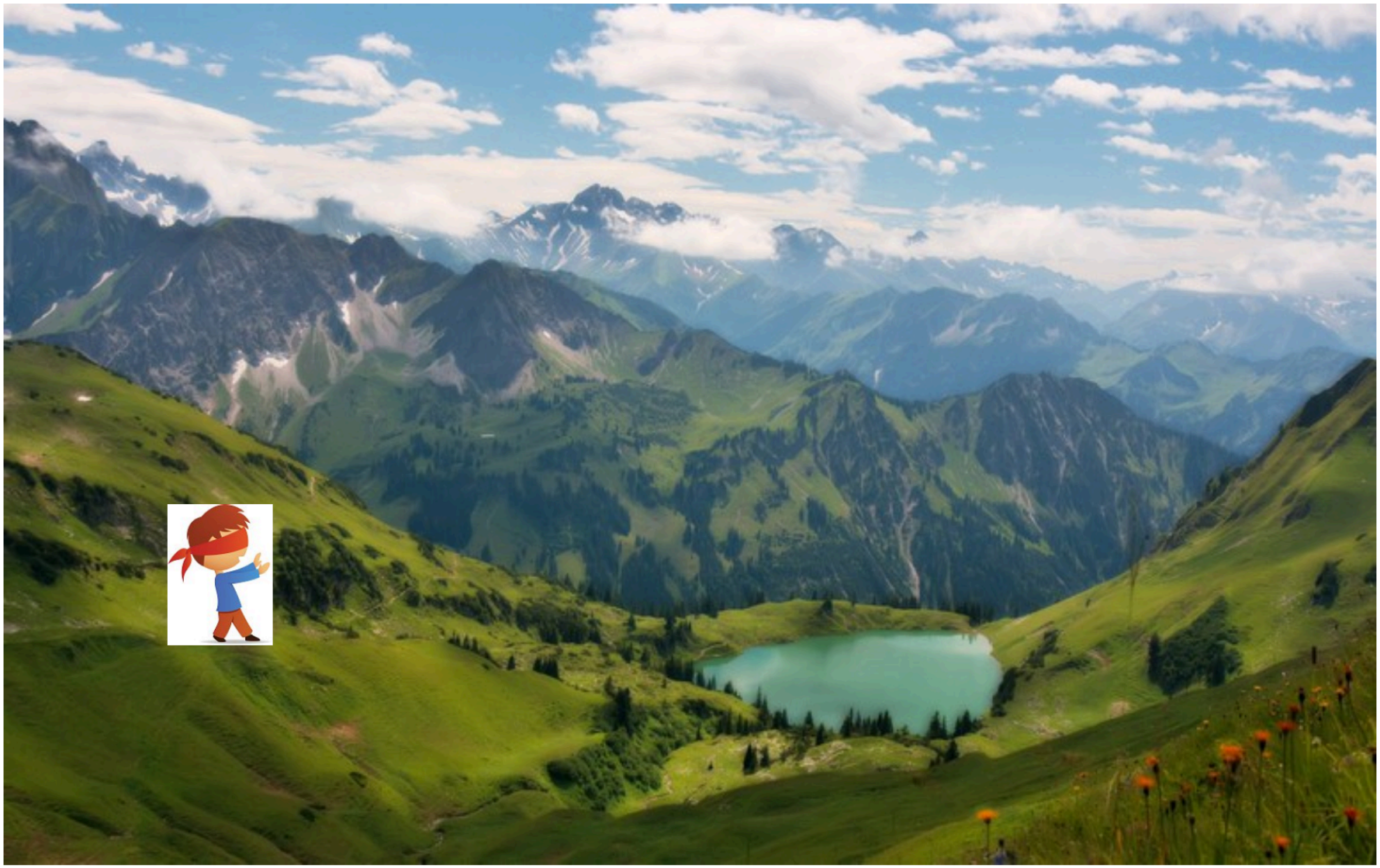
$$L_i(W + aW_1 + bW_2)$$

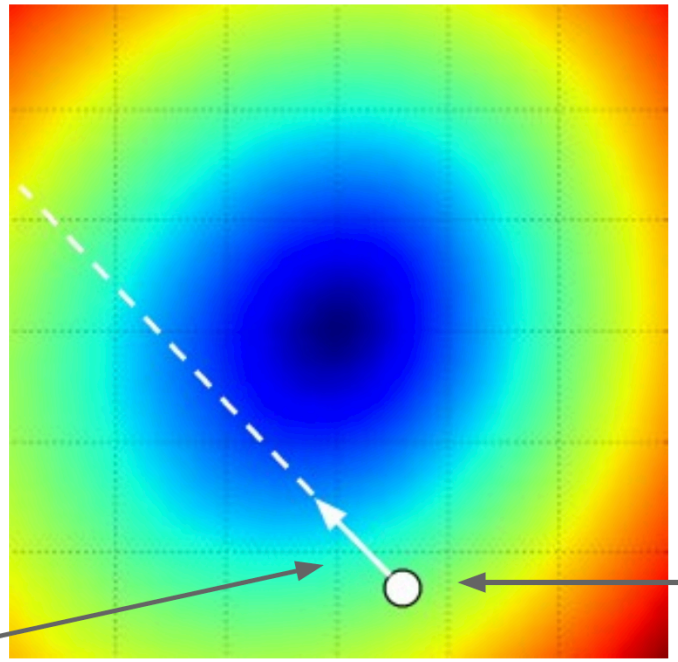


b

a

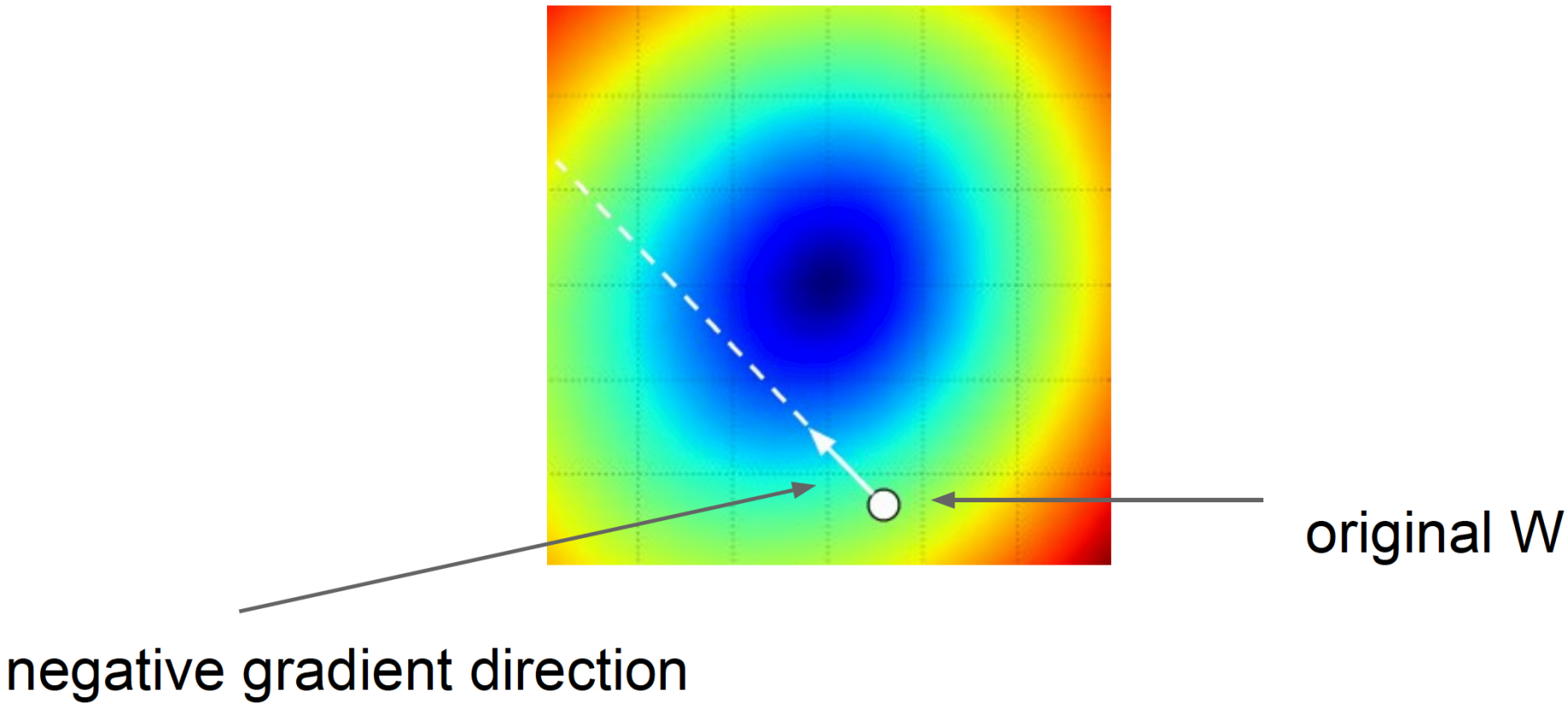
Gradient Descent





original W

negative gradient direction



Step size: learning rate
Too big: will miss the minimum
Too small: slow convergence

Gradient descent

In 1-dimension, the derivative of a function:

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

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

Evaluate the gradient numerically

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

in practice:

$$[f(x+h) - f(x-h)]/2h$$

“centered difference formula”

Approximate
Slow to evaluate

Analytic Gradient

$$L_i = \sum_{j \neq y_i} \left[\max(0, w_j^T x_i - w_{y_i}^T x_i + 1) \right]$$

$$\nabla_{w_j} L_i = 1(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0) x_i$$

$$\nabla_{w_{y_i}} L_i = -\left(\sum_{j \neq y_i} 1(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0) \right) x_i$$

In summary:

- Numerical gradient: approximate, slow, easy to write
- Analytic gradient: exact, fast, error-prone

=>

In practice: Always use analytic gradient, but check implementation with numerical gradient. This is called a **gradient check.**