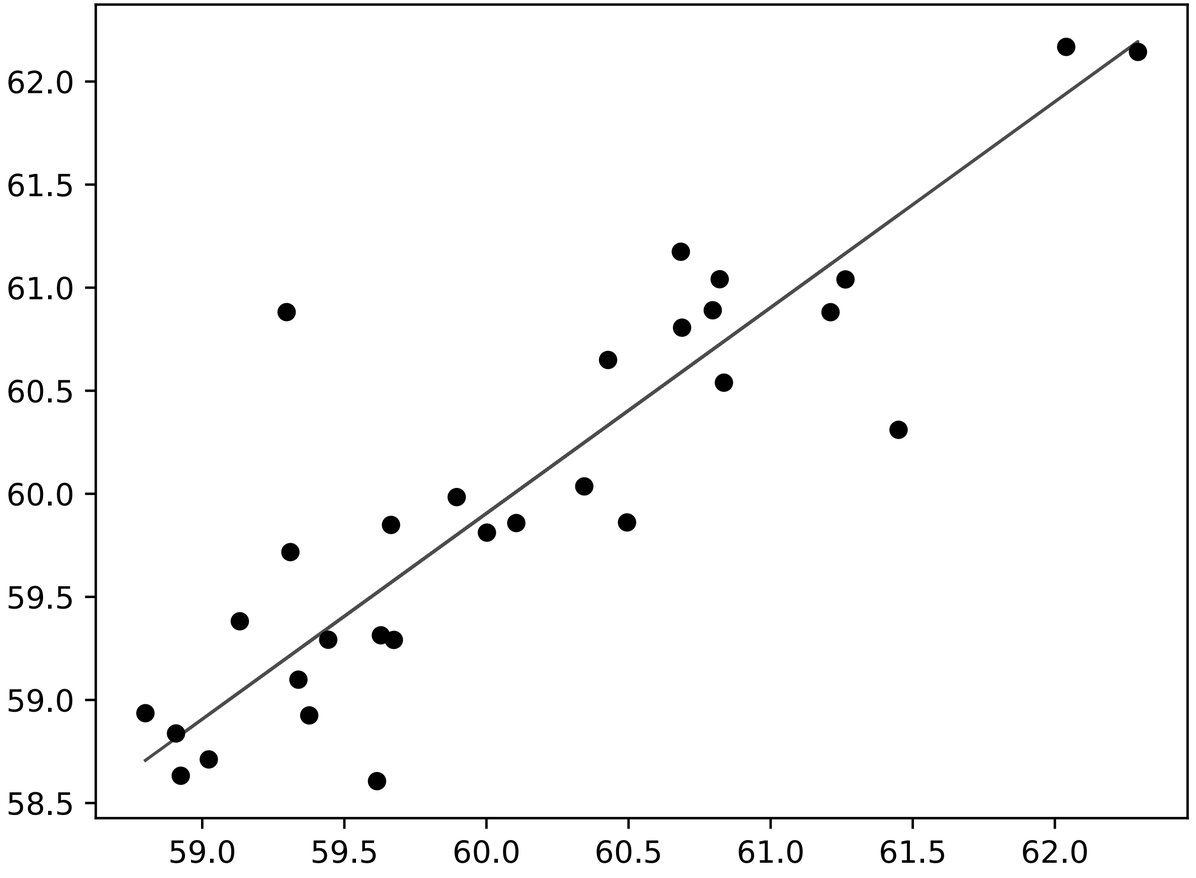
Lecture 6: Linear classifiers

Announcements

- Goodfellow book
- Szeliski 5.1-5.3 (can skip unsupervised learning)

Linear regression



Predict scalar **y** from vector **x**

Logistic regression

$\mathbf{x}$

$\hat{\mathbf{y}}$

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant

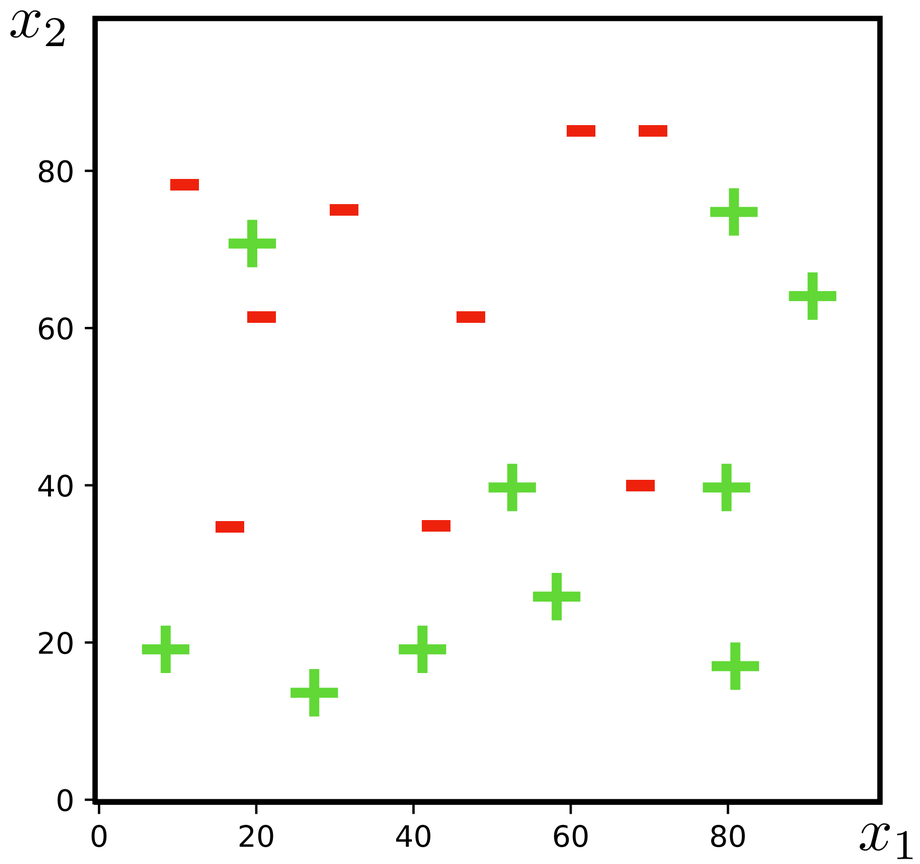
0 1

Predict probability of classes from vector **x**

How can we use a linear model for classification?

Linear decision boundaries

Consider a **binary** classification problem.

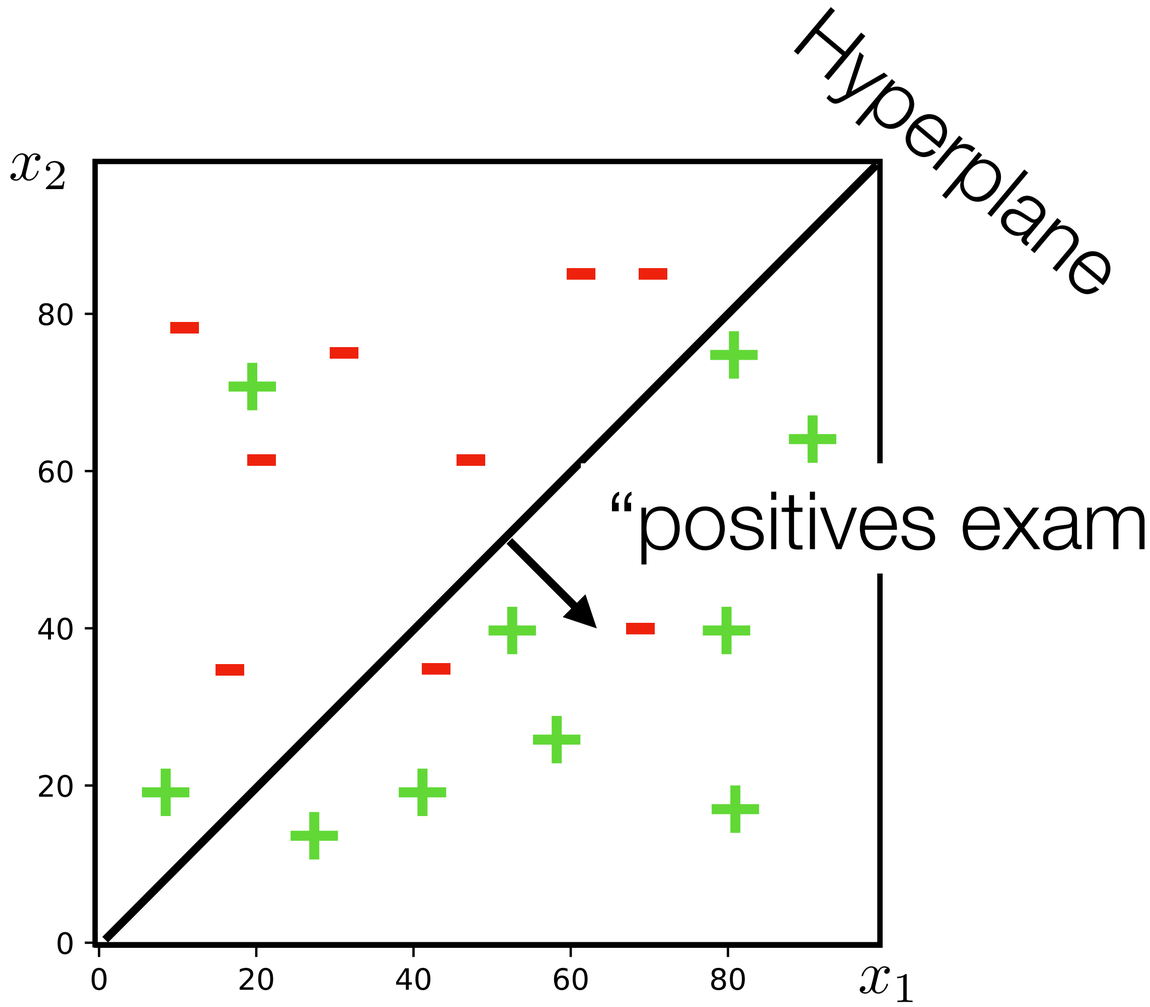


Training data

Example adapted from P. Isola

Linear decision boundaries

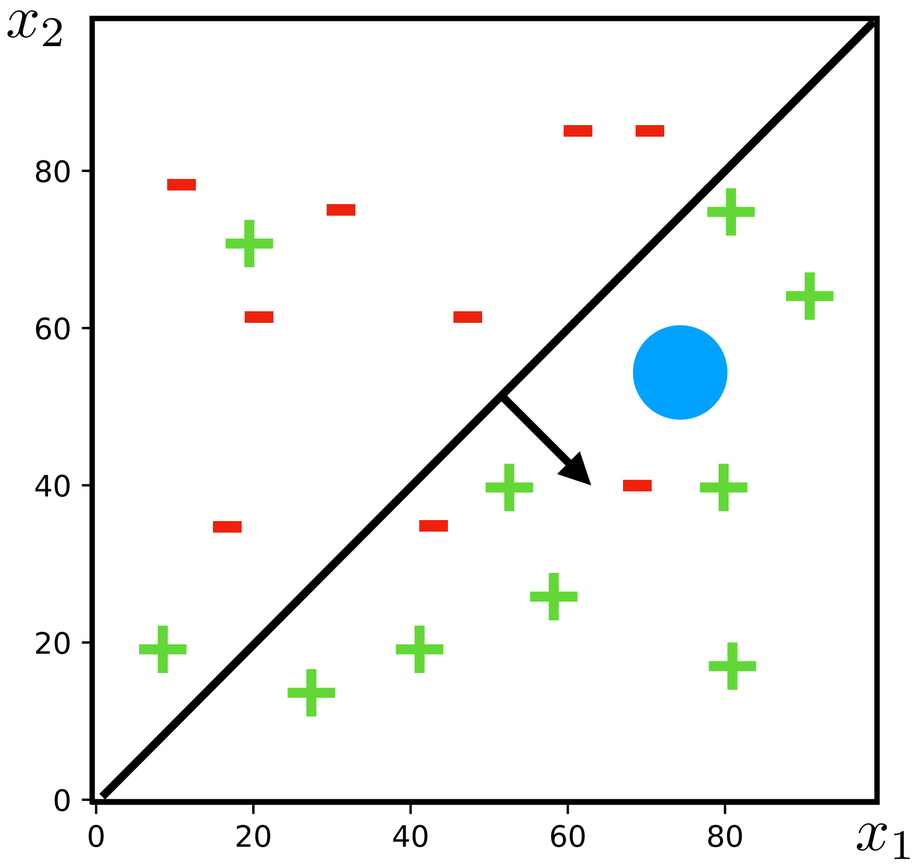
Consider a **binary** classification problem.



Training data

Example adapted from P. Isola

Linear decision boundaries



"What label is point?"

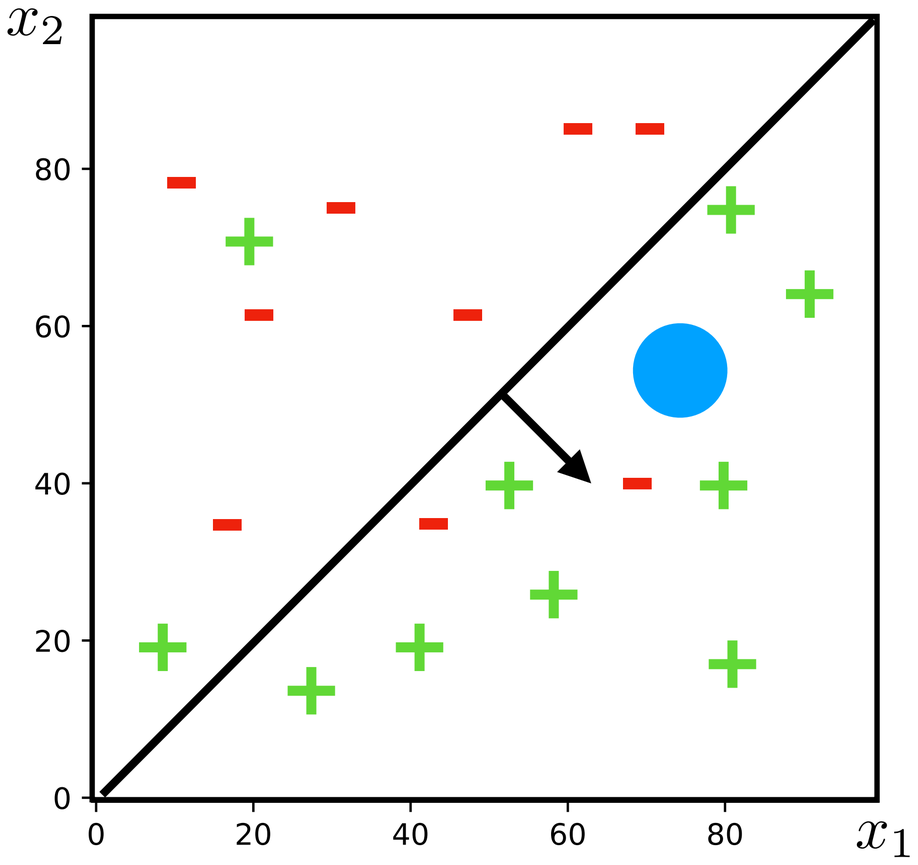
Which side of the hyperplane is **x** on?

$$\hat{y} = \mathbf{x}^T \mathbf{w} + b$$

$$g(\hat{y}) = \begin{cases} 1, & \text{if} \quad \hat{y} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Example adapted from P. Isola

Linear decision boundaries



"What label is point?"

Notational simplification:

Can get rid of b by "tacking on" a 1 to x

$$\hat{y} = \tilde{\mathbf{x}}^\top \tilde{\mathbf{w}}$$

$$\tilde{x} = \begin{bmatrix} x \\ 1 \end{bmatrix} \qquad \tilde{w} = \begin{bmatrix} w \\ b \end{bmatrix}$$

Multiway classification

For a **k**-class problem, we'll make a matrix "stacking" **k** hyperplanes as rows of a matrix $W \in \mathbb{R}^{k \times d}$:

$$W = \begin{bmatrix} \text{———} & w_1 & \text{———} \\ \text{———} & w_2 & \text{———} \\ & \dots & \\ \text{———} & w_k & \text{———} \end{bmatrix}$$

To classify an example: which row has the highest dot product with **x**?

$$z = Wx + b$$

$$g(\hat{y}) = \text{argmax}_k \, z_k$$

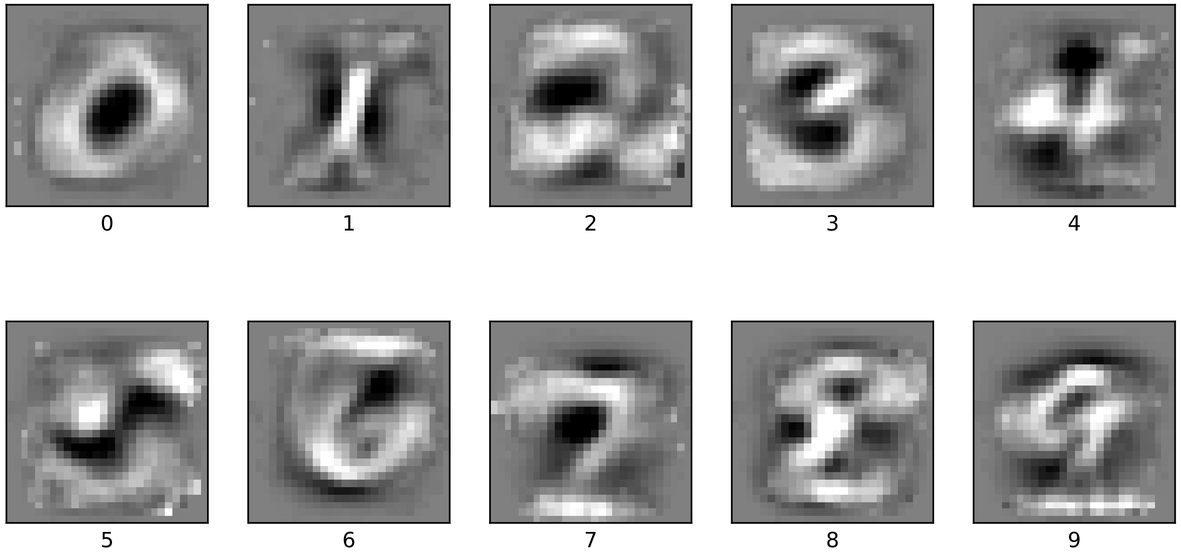
Example: handwritten digits

$$z = Wx + b$$

$$g(\hat{y}) = \text{argmax}_k \, z_k$$

$$W = \begin{bmatrix} \text{———} & w_1 & \text{———} \\ \text{———} & w_2 & \text{———} \\ & \cdots & \\ \text{———} & w_k & \text{———} \end{bmatrix}$$

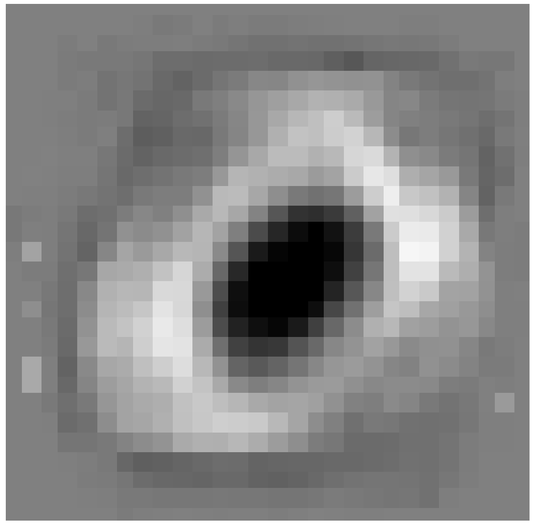
inputs



example of learned weights

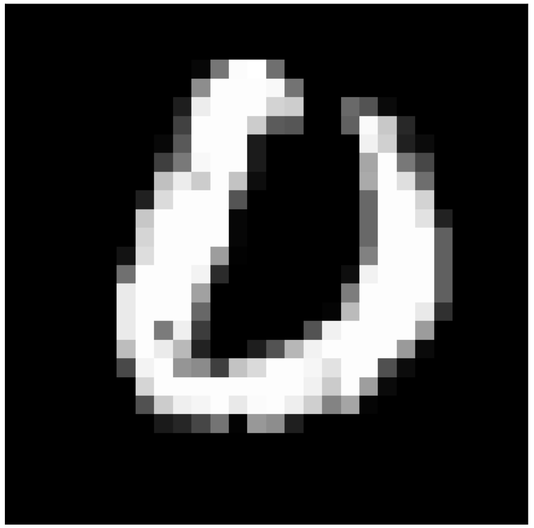
Visualization adapted from scikit-learn.

Example: handwritten digits



w_0

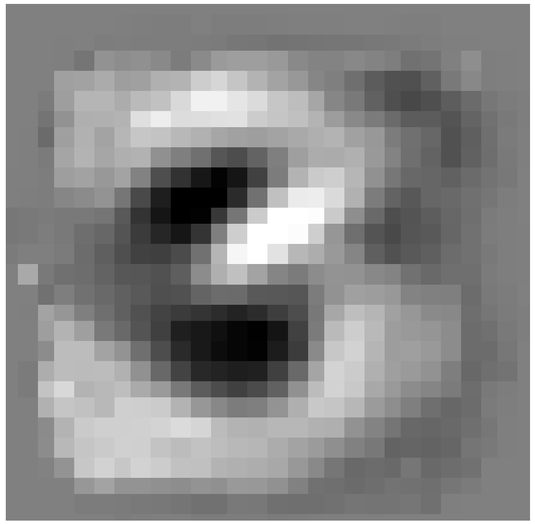
$\bullet$



x

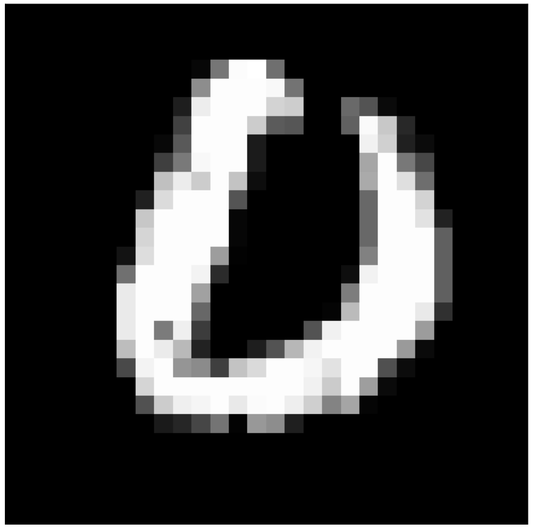
$=$ high number

Example: handwritten digits



w_3

•



x

= low number

Converting to probabilities

We want probabilistic predictions: $p(y_k \mid x)$

i.e. $\sum_k p(y_k \mid x) = 1$, and $p(y_k \mid x) \geq 0$

Solution: use the **softmax** function:

$$\hat{y} = \text{softmax}(Wx + b)$$

$$p(y_k \mid x) = \hat{y}_k$$

non-negative

sums-to-1

$$\text{softmax}(z)_k = \frac{\exp(z_k)}{\sum_i \exp(z_i)}$$

Recall: how to represent class labels?

One-hot vector

Training data

$\mathbf{x}$ y

{ , "Fish" }

{ , "Grizzly" }

{ , "Chameleon" }

,

⋮

Training data

$\mathbf{x}$ y

{ , 1 }

{ , 2 }

{ , 3 }

,

⋮

Training data

$\mathbf{x}$ $\mathbf{y}$

{ , [0,0,1] }

{ , [0,1,0] }

{ , [1,0,0] }

,

⋮

Source: Isola, Torralba, Freeman

Loss function

The loss we'd ideally want: **0-1 loss**

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = 1 - \mathbb{1}(\hat{\mathbf{y}}, \mathbf{y})$$ ← number of misclassifications

Least squares (predict 1 for true class, 0 for others)

$$\mathcal{L}(\hat{y}, y) = \sum_{k=1}^{K} (y_k - \hat{y}_k)^2$$

Cross entropy: a good surrogate that we'll be able to optimize:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

Adapted from P. Isola

Loss function

Simpler than it looks:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

Therefore:

One-hot vectors

Training data

$\mathbf{x}$	$\mathbf{y}$
	[0,0,1]
	[0,1,0]
	[1,0,0]

Adapted from P. Isola

Ground truth label $\mathbf{y}$

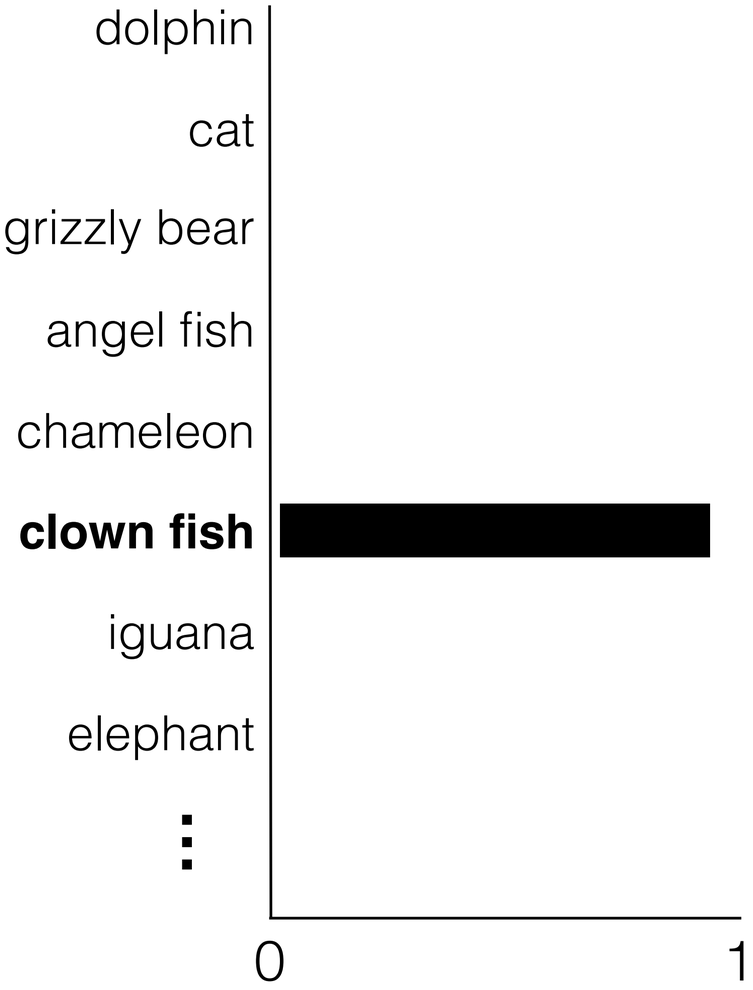
$\mathbf{x}$

[0,0,0,0,0,1,0,0,…]

Source: Isola, Torralba, Freeman

Ground truth label $\mathbf{y}$

$\mathbf{x}$



Source: Isola, Torralba, Freeman

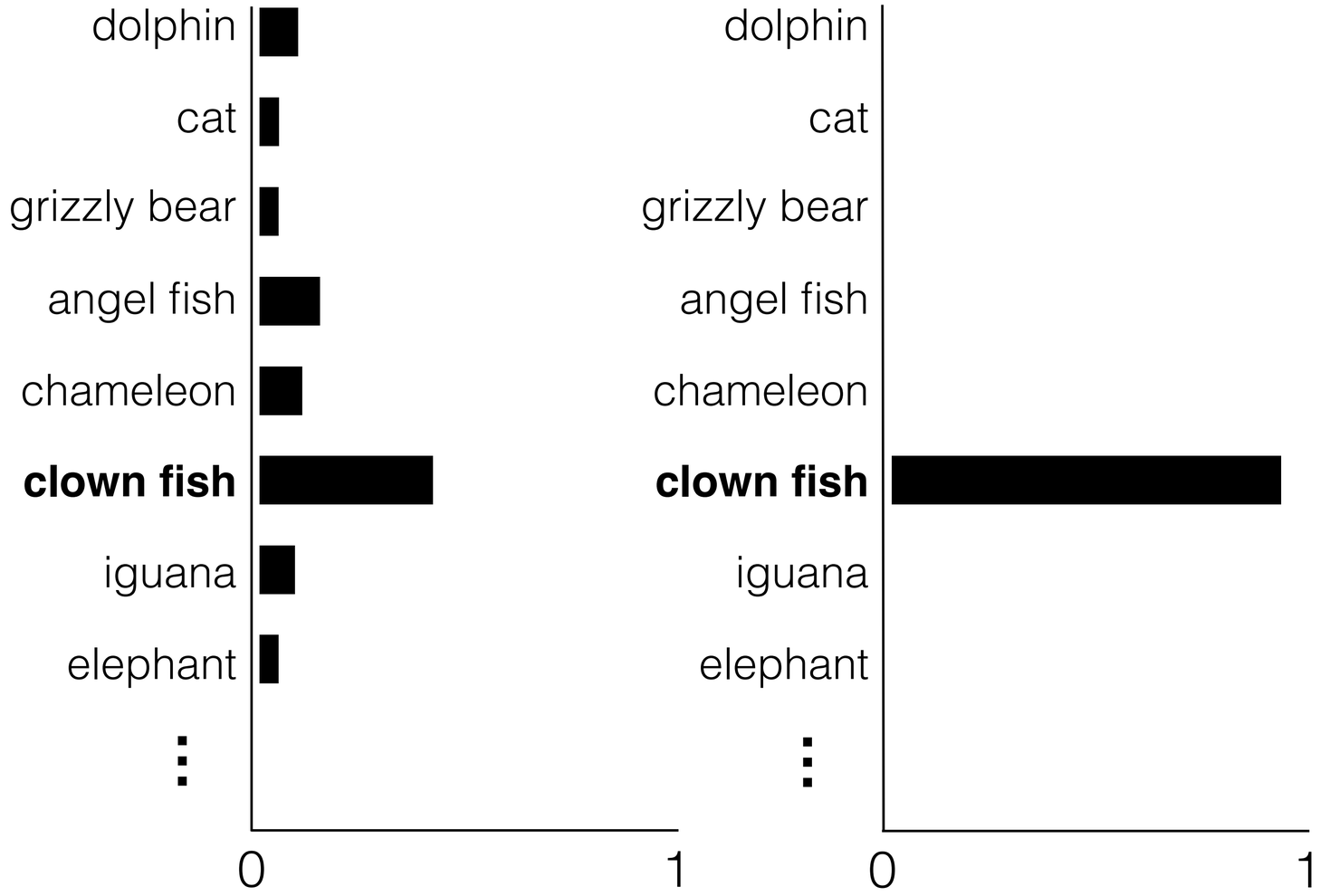
Prediction $\hat{\mathbf{y}}$

$f_\theta : X \rightarrow \mathbb{R}^K$

Ground truth label $\mathbf{y}$

$\mathbf{x}$

f



Source: Isola, Torralba, Freeman

Prediction $\hat{\mathbf{y}}$

$$f_\theta : X \to \mathbb{R}^K$$

Ground truth label $\mathbf{y}$

Loss

$$-\sum_{k=1}^{K} y_k \log \hat{y}_k$$

$\mathbf{x}$

f

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮

0 1

$\odot$

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮

0 1

0 1

Source: Isola, Torralba, Freeman

Prediction $\hat{\mathbf{y}}$

$$f_\theta : X \to \mathbb{R}^K$$

Ground truth label $\mathbf{y}$

Loss

$$-\sum_{k=1}^{K} y_k \log \hat{y}_k$$

$\mathbf{x}$

f

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮

0 1

$\odot$

dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮

0 1

0 1

Source: Isola, Torralba, Freeman

Hinge loss

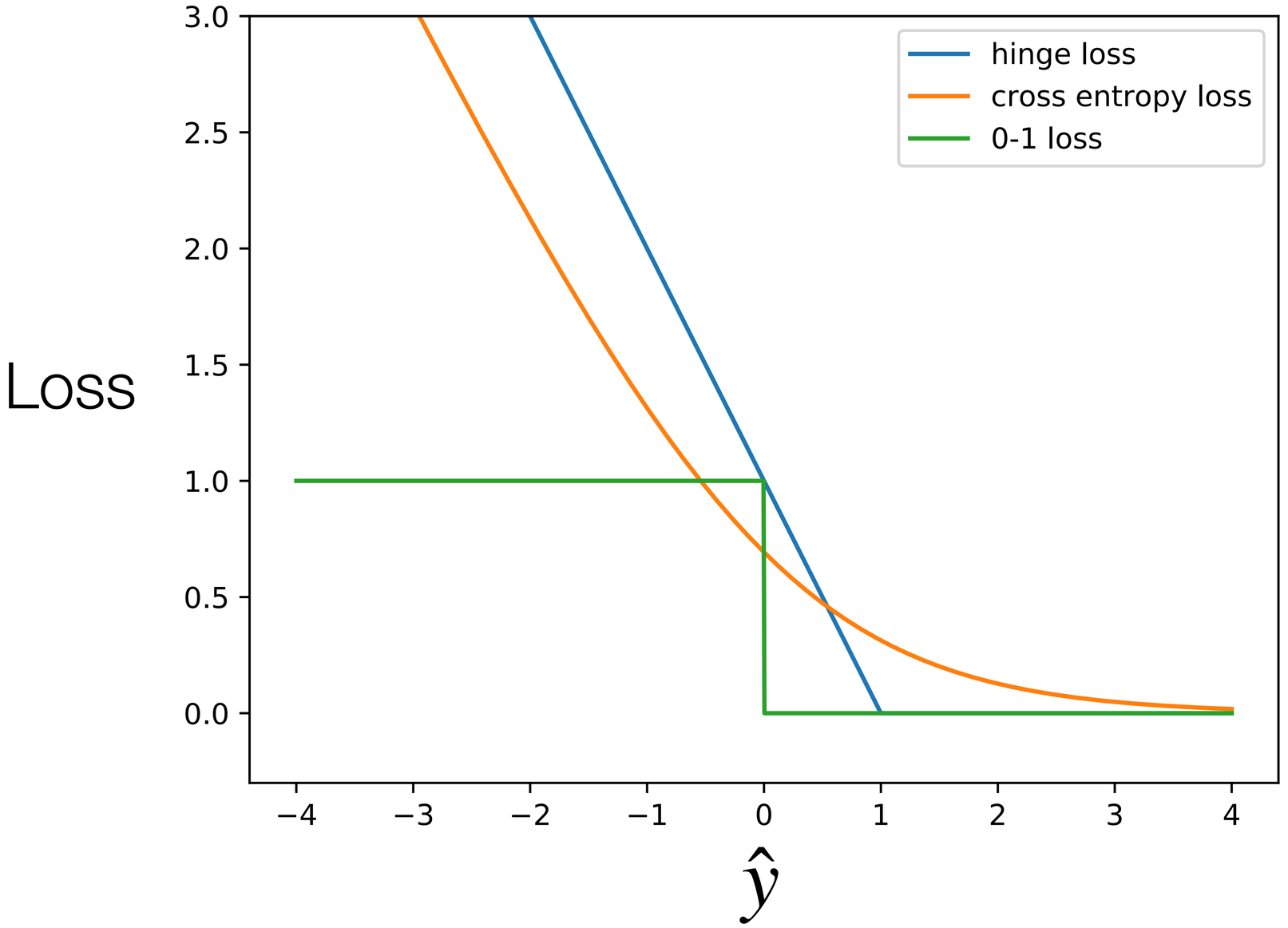
One more you might see: **hinge loss**

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \max(0, 1 - y\hat{y}, 0)$$ ← upper bound on 0-1 loss (true class **y** is {+1, -1} instead of {0,1})

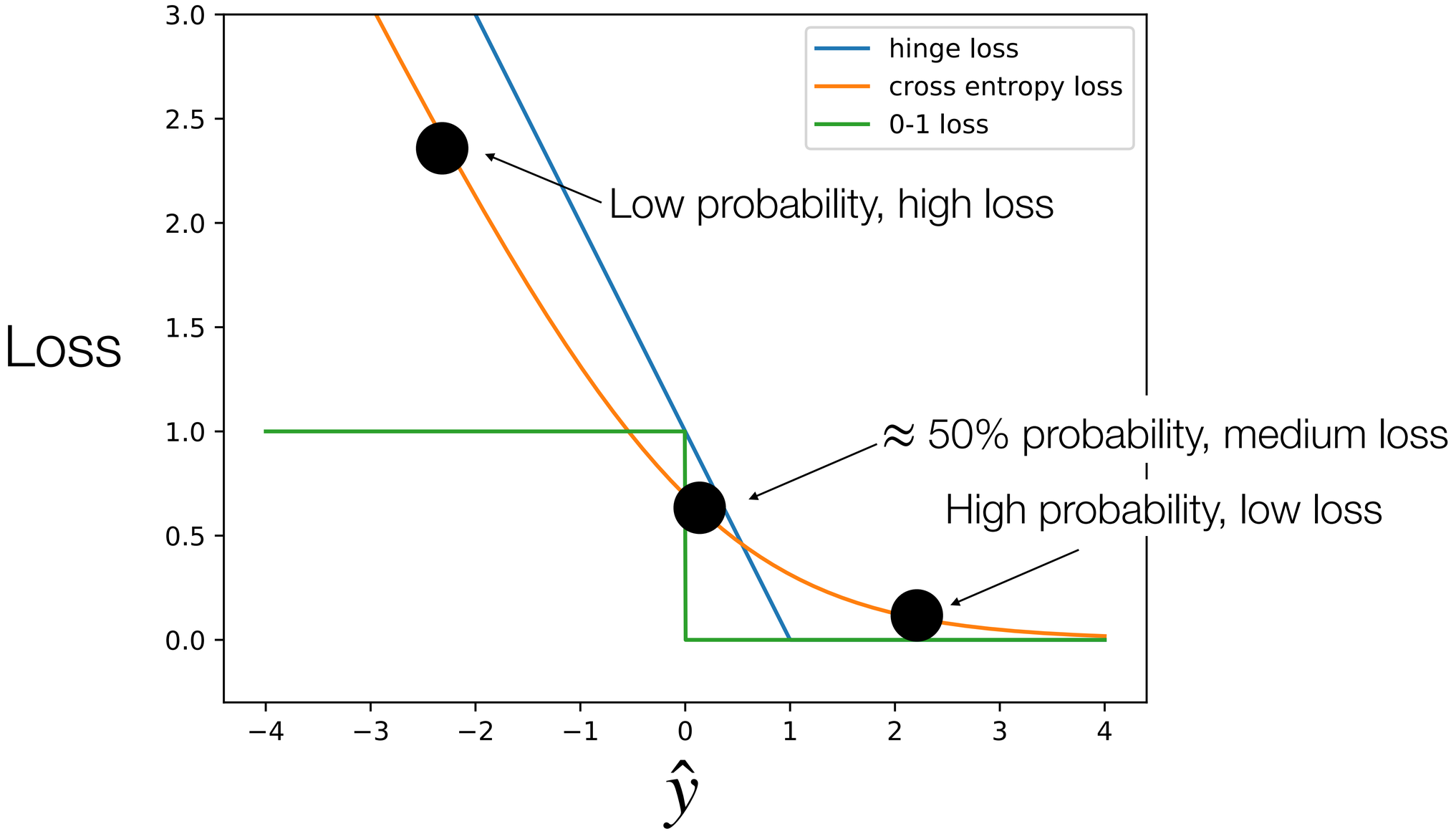
A linear classifier that uses a hinge loss is called a **support vector machine (SVM)**.

It has some very nice properties (they can be made nonlinear with "kernels"). Generally performs similar to logistic regression.

Loss functions

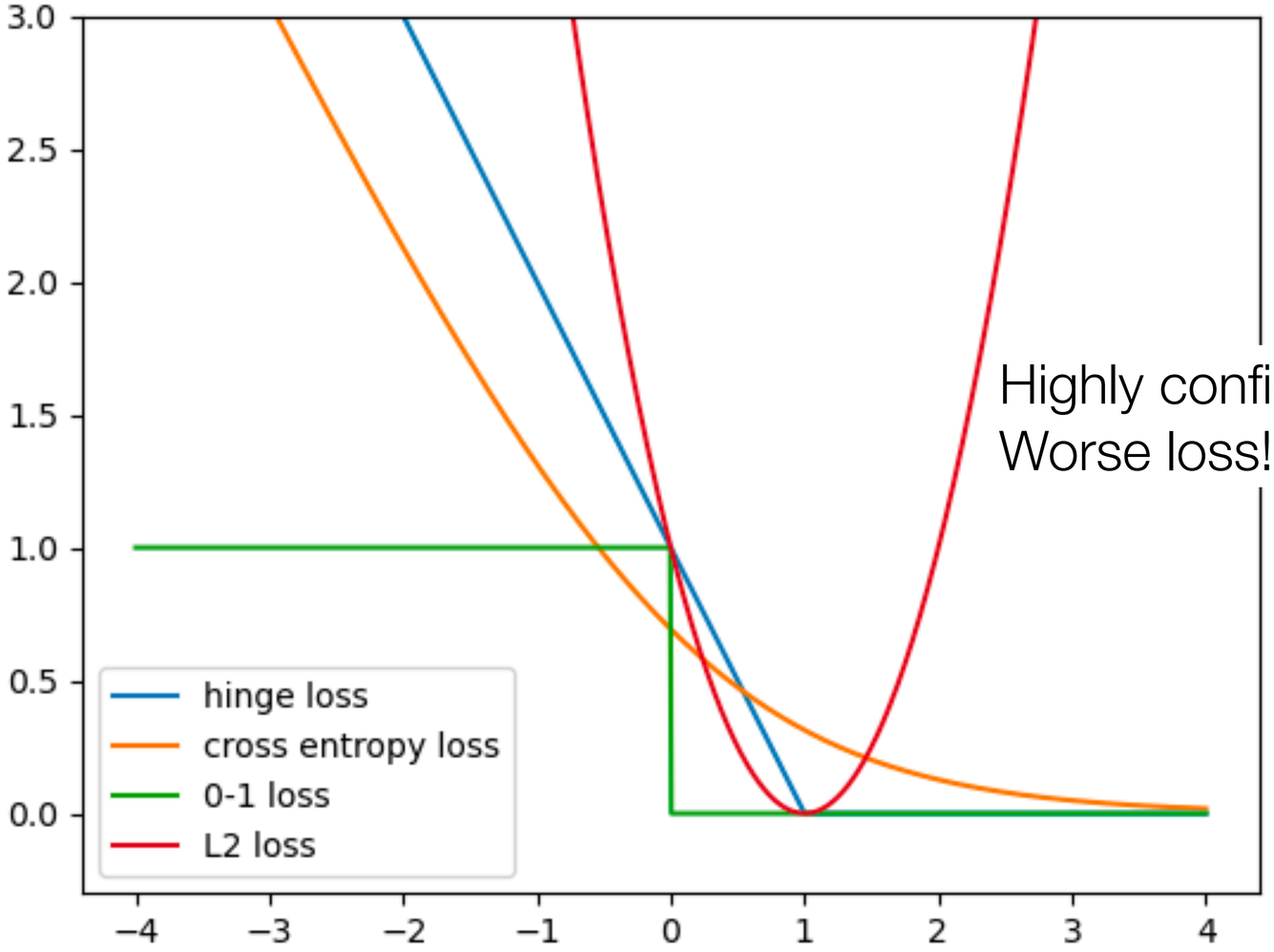


Loss functions



Why not squared loss?

Loss



$\hat{y}$

Highly confident prediction?
Worse loss!

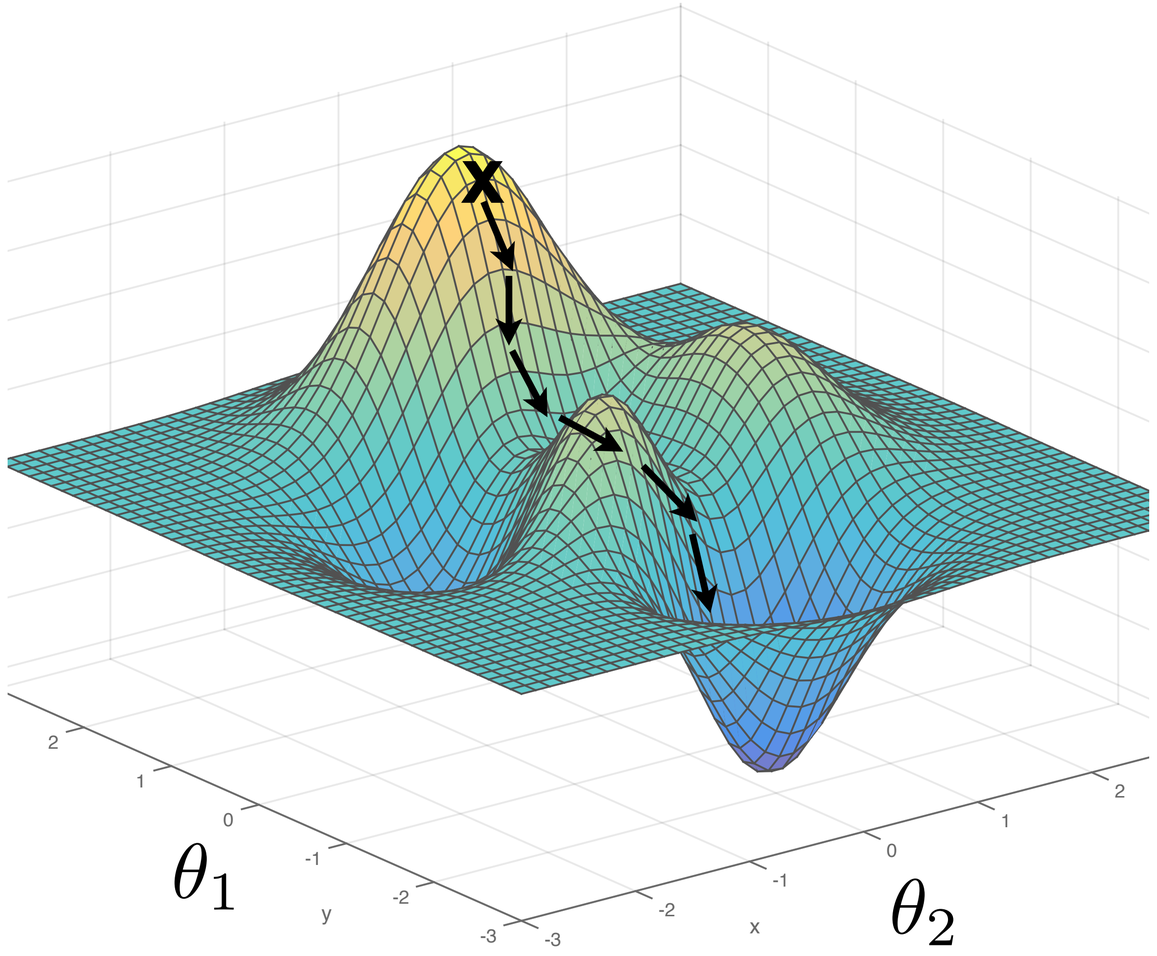
How do we learn a classifier?

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}$$

Source: Isola, Torralba, Freeman

Gradient descent

$J(\theta)$



$$\theta^* = \arg\min_{\theta} J(\theta)$$

Source: Isola, Torralba, Freeman

Gradient descent

$$\theta^* = \arg\min_\theta \underbrace{\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}$$

One iteration of gradient descent:

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta)\Big|_{\theta=\theta^t}$$

learning rate

Source: Isola, Torralba, Freeman

Gradient descent

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta) \Big|_{\theta=\theta^t}$$

What's this again?

1. Gradient of the loss w.r.t. the classifier's parameters
2. Direction of steepest descent
3. "Local" linear approximation to the function

For a refresher on gradients and partial derivatives:
https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives#partial-derivatives

Gradient descent

Take gradient at current θ

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta) \Big|_{\theta=\theta^t}$$

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J}{\theta_1} \\ \frac{\partial J}{\theta_2} \\ \cdots \\ \frac{\partial J}{\theta_d} \end{bmatrix}$$

Vector of partial derivatives for loss

Estimating the gradient

Idea #1: finite differences $\dfrac{f(x+\epsilon) - f(x)}{\epsilon}$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\theta_1} \\ \frac{\partial J}{\theta_2} \\ \cdots \\ \frac{\partial J}{\theta_d} \end{bmatrix} \approx \frac{J(\theta + \epsilon e_2) - J(\theta - \epsilon e_2)}{2\epsilon}$$

where e_2 is vector of zeros except for a 1 in 2nd component

Estimating the gradient

Idea #1: finite differences $\frac{f(x+\epsilon) - f(x)}{\epsilon}$

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\theta_1} \\ \frac{\partial J}{\theta_2} \\ \cdots \\ \frac{\partial J}{\theta_d} \end{bmatrix}$$

- Easy to compute but **slow**.
- Inaccurate in some cases, unless careful
- Useful for **checking** other methods

Estimating the gradient

Idea #2: compute it analytically using calculus.

Simple example: binary labels, squared loss, and regularization on θ

$$J(\theta) = \lambda||\theta||^2 + \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta^\top x_i)^2$$

$$\nabla_\theta$$

$$\nabla_\theta J(\theta) = 2\lambda\theta - \frac{1}{N}\sum_{i=1}^{N} 2(y_i - \theta^\top x_i)x_i$$

Example from D. Fouhey

Analyzing the gradient descent update

Recall update rule: $\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta)\Big|_{\theta=\theta^t}$

How does this change θ ?

"Decay" toward 0 — Scalar α

$$-\nabla_\theta J(\theta) = -2\lambda\theta - \frac{1}{N}\sum_{i=1}^{N} 2(y_i - \theta^\top x_i)x_i$$

Example from D. Fouhey

Analyzing the gradient descent update

What happens at each example?

Scalar α

$$-\nabla_\theta J(\theta) = -2\lambda\theta - \frac{1}{N}\sum_{i=1}^{N} 2(y_i - \theta^\top x_i)x_i$$

If $\theta^\top x_i < y$ (too *low*): then $\theta_{t+1} = \theta + \alpha x_i$ for some $\alpha > 0$

Dot product before: $\theta^\top x_i$

Dot product after: $(\theta + \alpha x_i)^\top x_i = \theta^\top x_i + \alpha x_i^\top x_i$

Example from D. Fouhey

Computational issues

Batch gradient descent

Loss function:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} L(x_i, y_i, \theta)$$

Its gradient is the **sum of gradients** for each example:

$$\nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla L(x_i, y_i, \theta)$$

Requires iterating over every training example each gradient step!

Can we speed this up?

Stochastic gradient descent

This is just an **average**:

$$\nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla L(x_i, y_i, \theta)$$

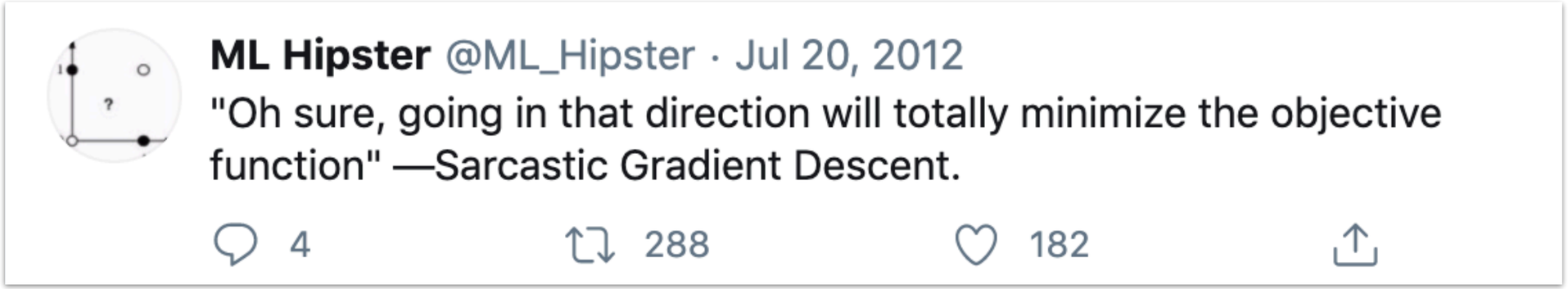
We know from statistics that we can estimate the average of a full "population" from a **sample.**

$$\nabla J(\theta) \approx \frac{1}{|B|} \sum_{i \in B} \nabla L(x_i, y_i, \theta)$$

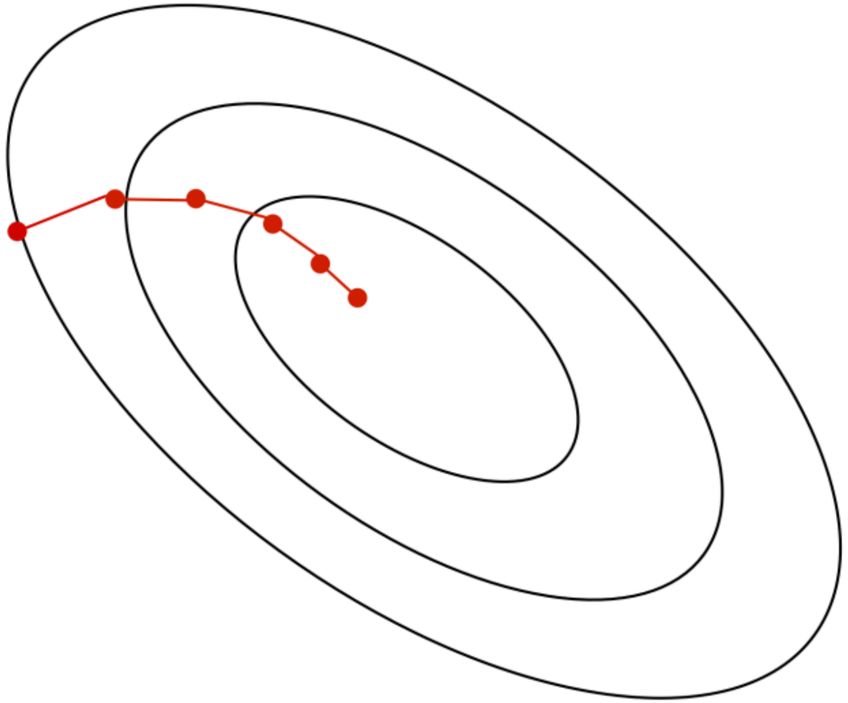
where B is a **minibatch:** a random subset of examples.

This is called **stochastic gradient descent (SGD)**.

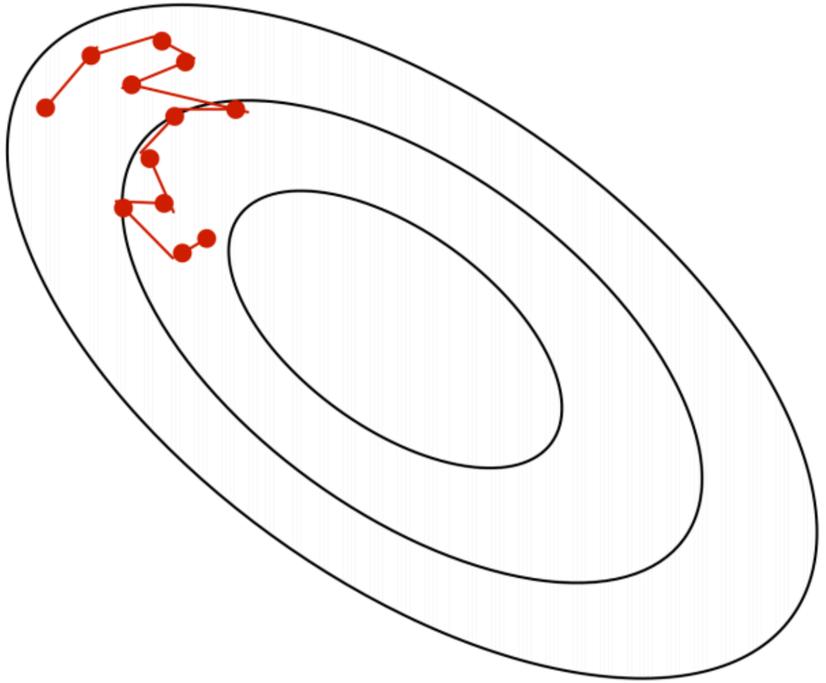
Stochastic gradient descent



Stochastic gradient descent



Batch gradient descent



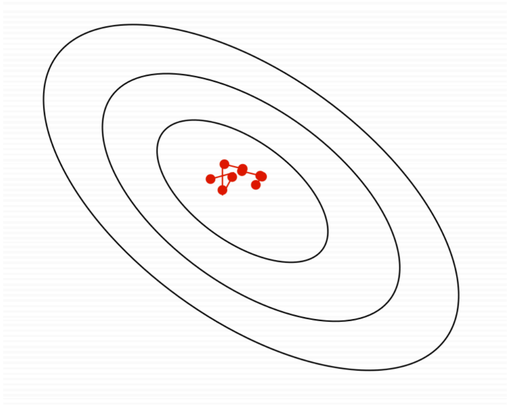
Stochastic gradient descent

Source: R. Grosse

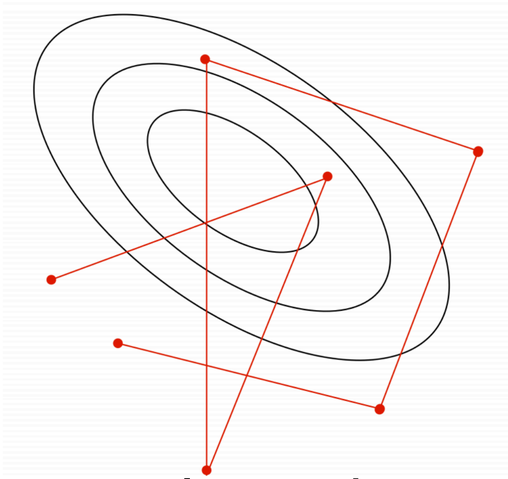
Learning rate

- Sensitive to the learning rate:

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta J(\theta)\Big|_{\theta=\theta^t}$$



Small learning rate

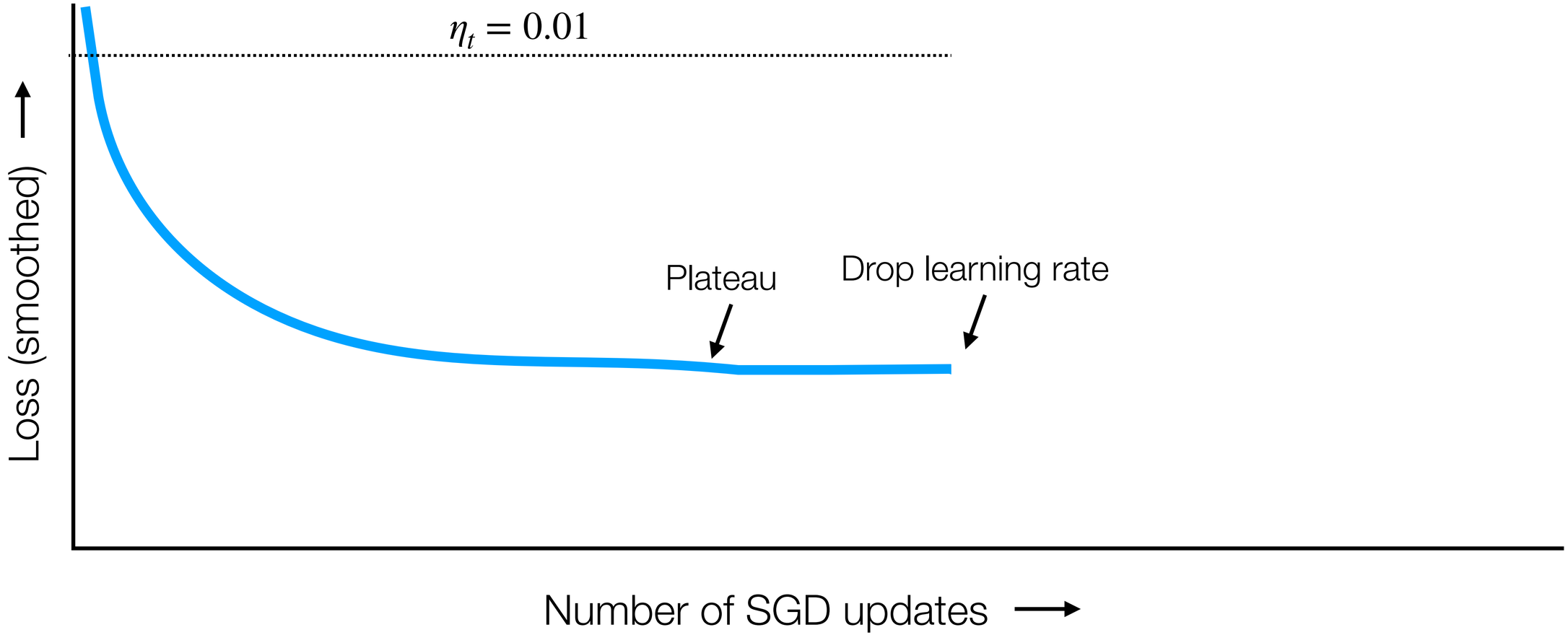


Large learning rate

Figure source: R. Grosse

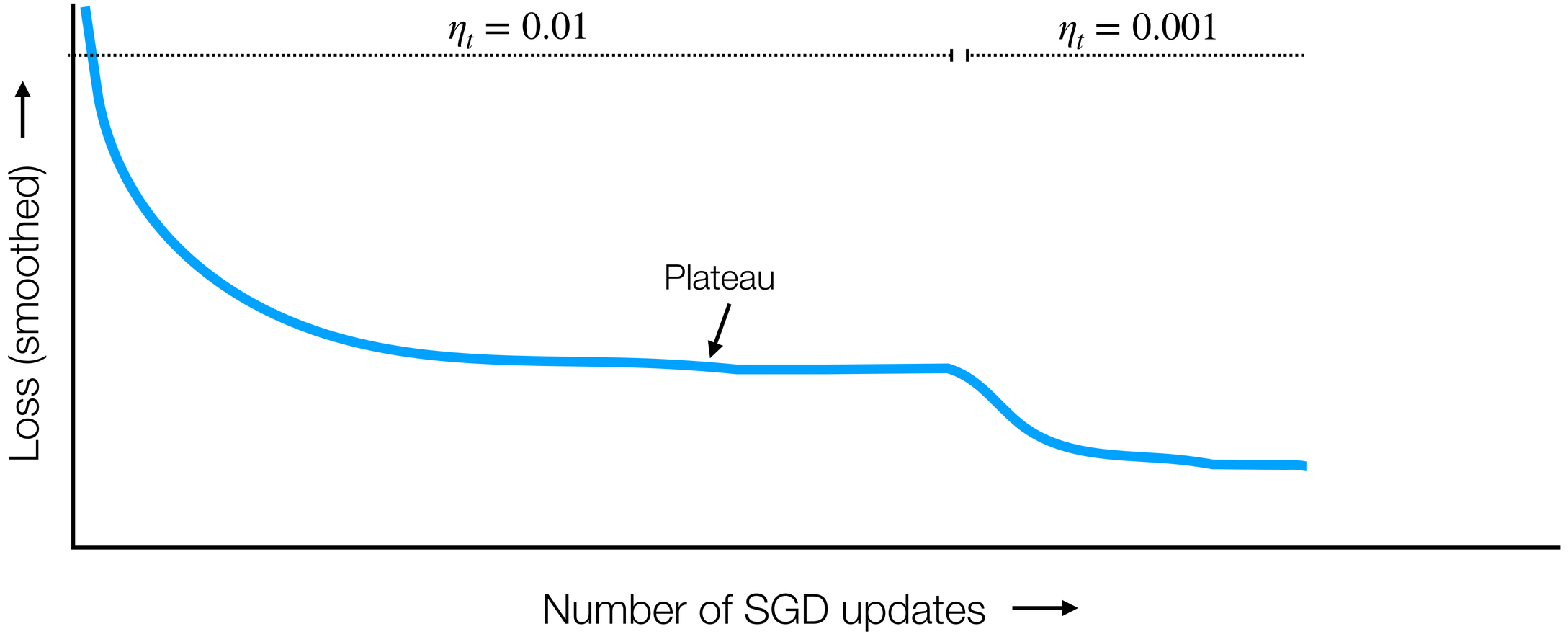
Learning rate schedules

- Start with high learning rate, and decrease over time.



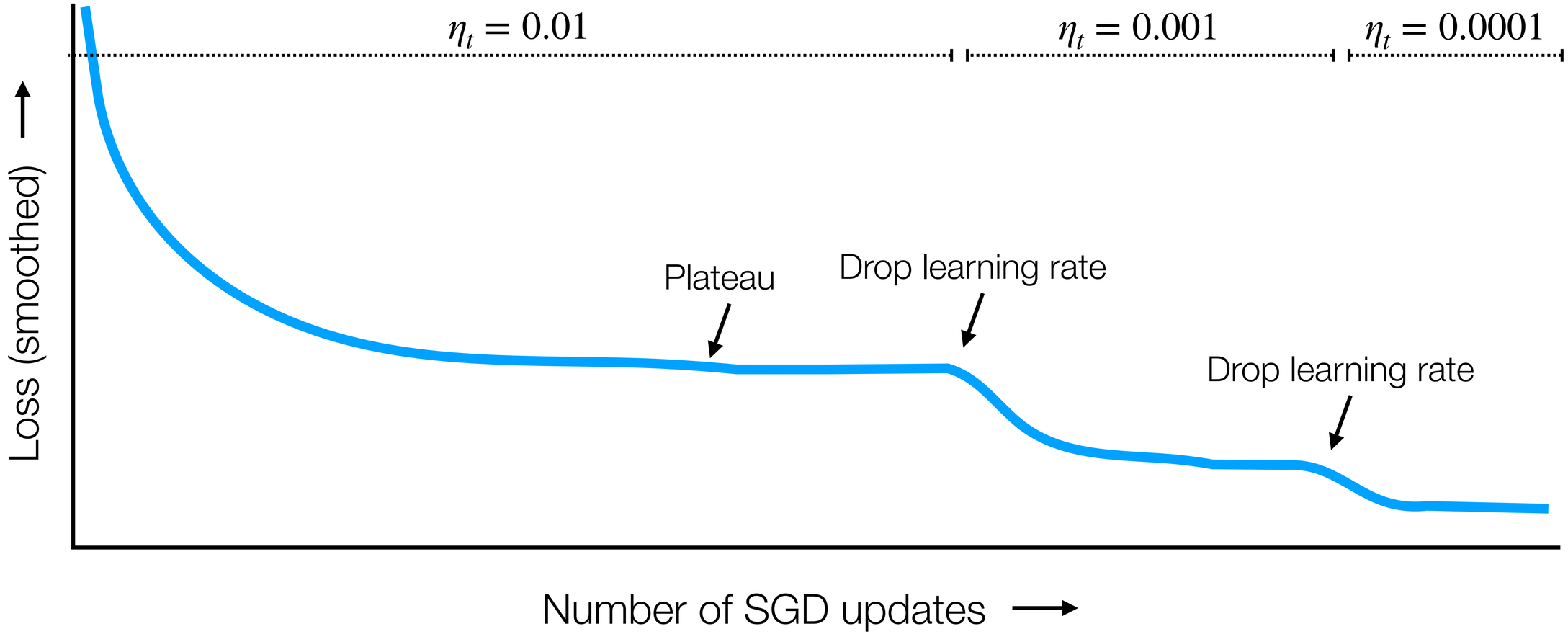
Learning rate schedules

- Start with high learning rate, and decrease over time.



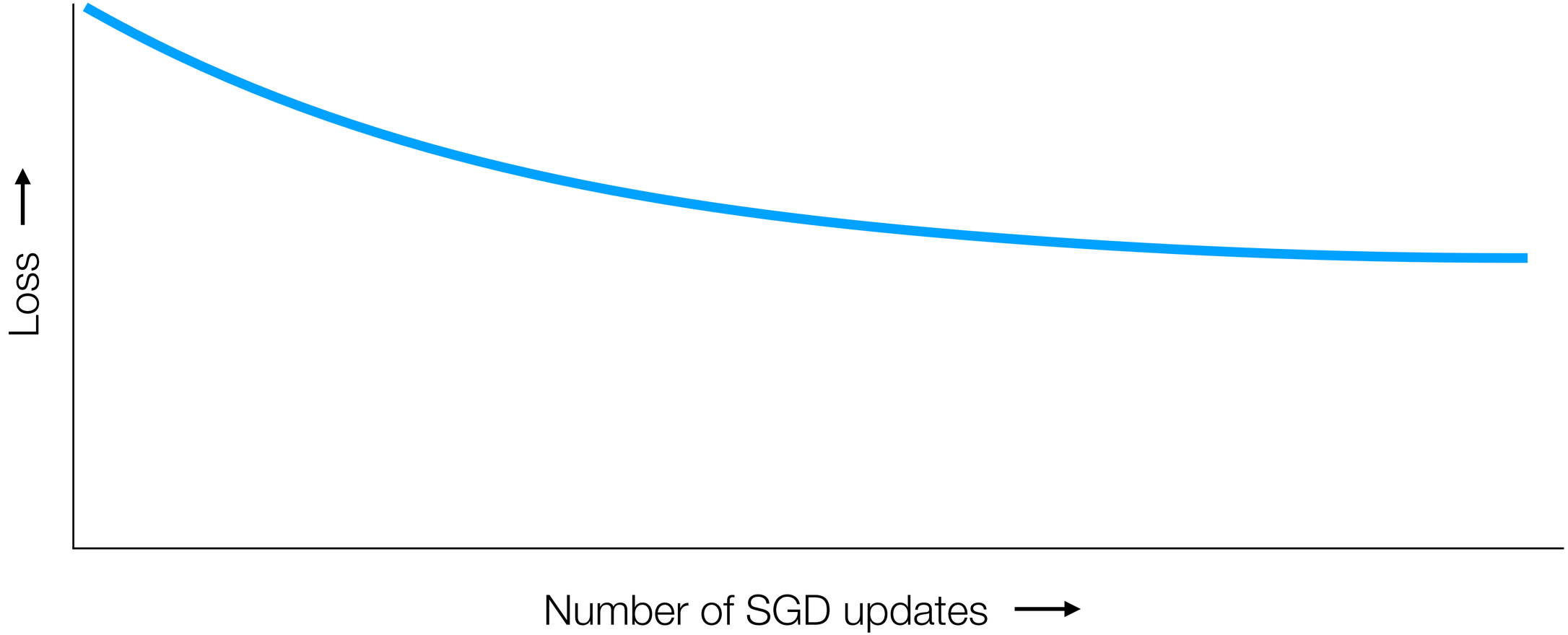
Learning rate schedules

- Start with high learning rate, and decrease over time.



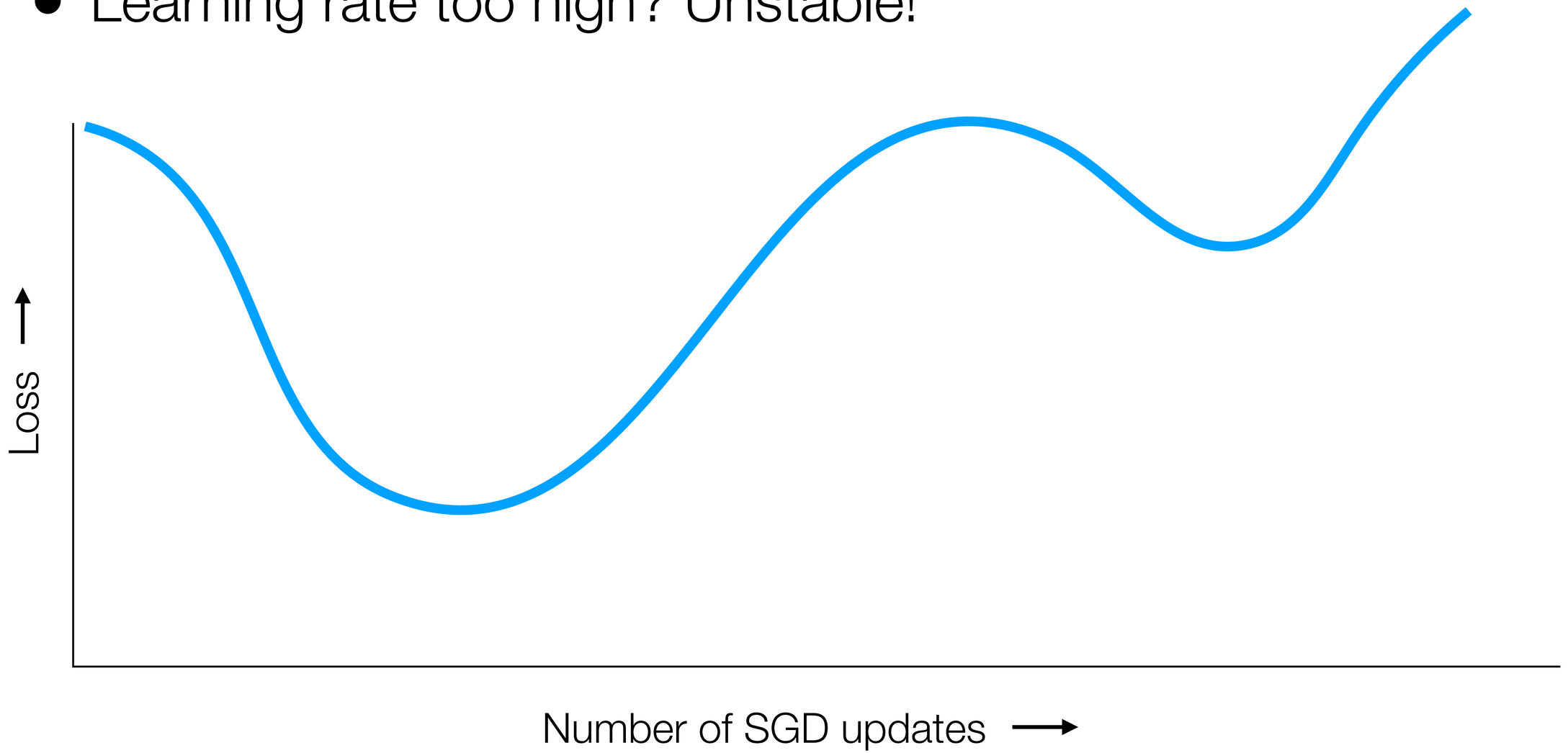
Learning rate schedules

- What if you always use a small learning rate?
- Loss goes down extremely slowly



Learning rate schedules

- Learning rate too high? Unstable!



Learning rate schedules

- Choosing the initial learning rate is surprisingly hard. Often requires grid search, i.e., trying many rates and choosing the best.
- When do you drop the learning rate? Some strategies:
  - Wait until validation or training loss plateaus, then drop it (e.g., by a factor of 10).
  - Smoothly drop the learning rate over time. Requires choosing the rate of dropping.
- Another option: decrease *and* increase the learning rate using a periodic function (e.g., cosine) [Loshchilov & Hutter, 2017]. Perhaps surprisingly, this works well.

Regularization

- As before, add L_2 regularization $R(\theta) = ||\theta||^2$ model parameters.
- Early stopping: stop training when validation loss increases, and revert to previous checkpoint.

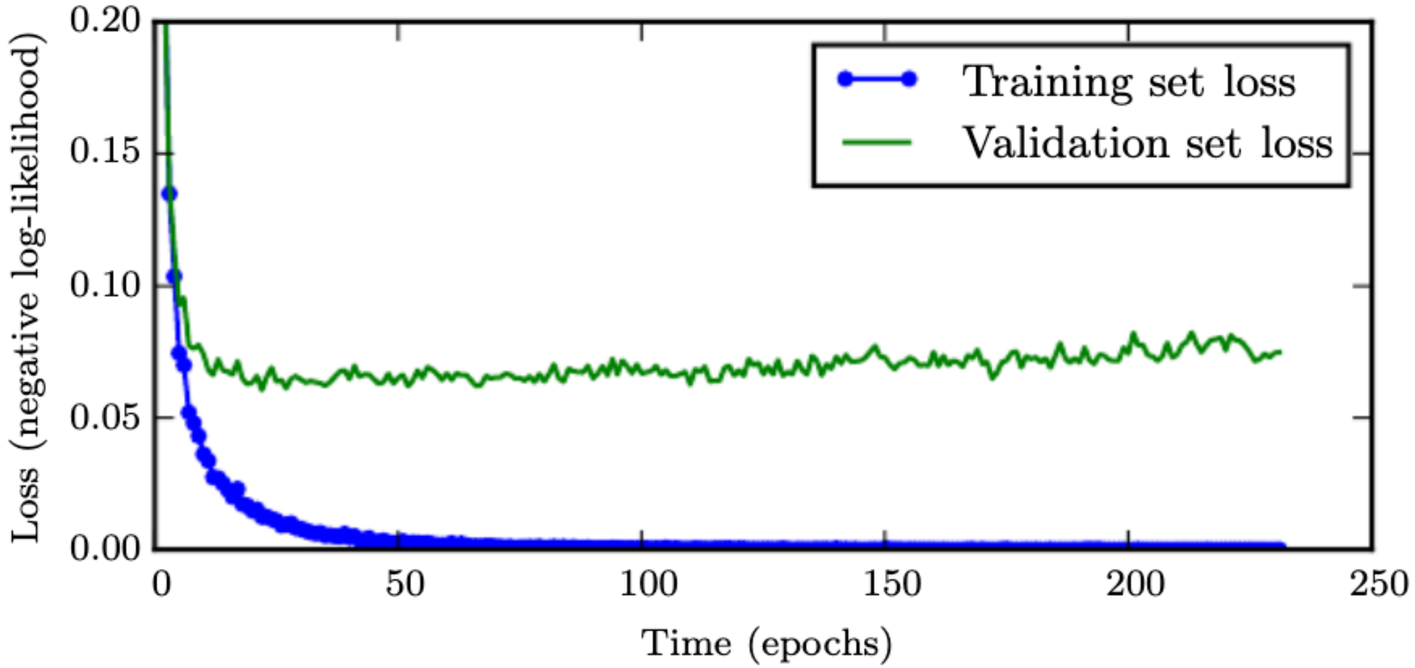
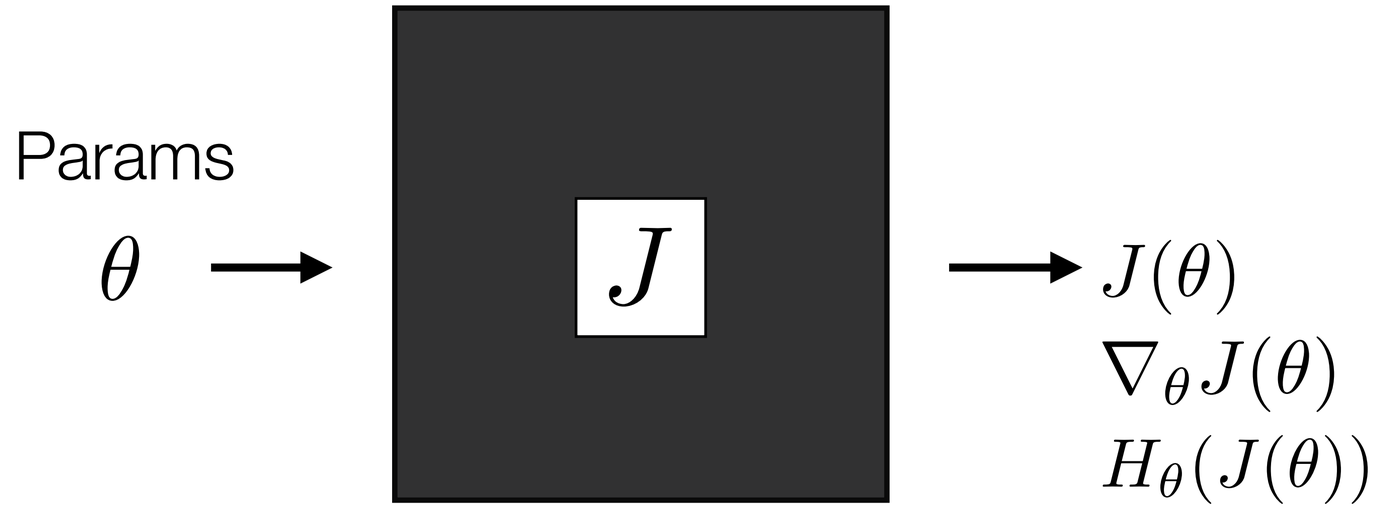


Figure source: [Goodfellow et al., 2016]

Optimization beyond gradient descent



$$\theta^* = \arg\min_\theta J(\theta)$$

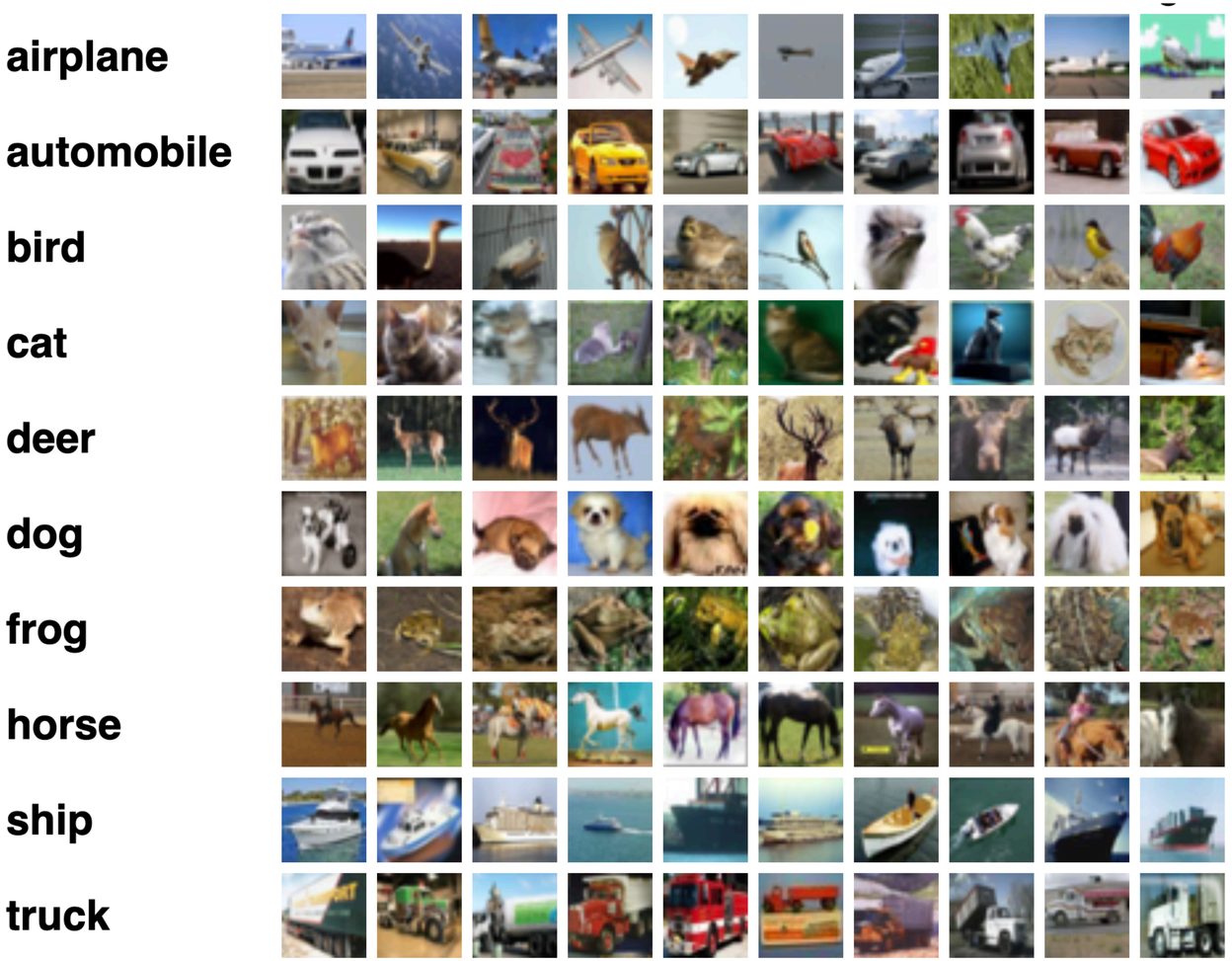
- What's the knowledge we have about J?
  - We can evaluate $J(\theta)$ ← Black box optimization
  - We can evaluate $J(\theta)$ and $\nabla_\theta J(\theta)$ ← First order optimization
  - We can evaluate $J(\theta)$, $\nabla_\theta J(\theta)$, and $H_\theta(J(\theta))$ ← Second order optimization

Gradient

Hessian

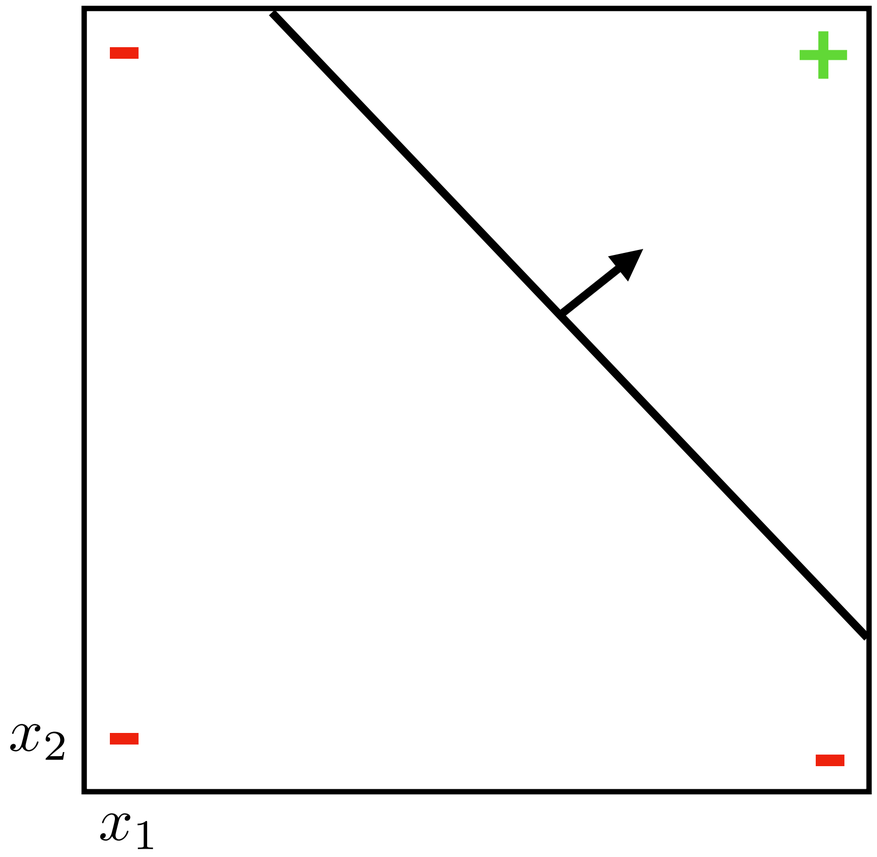
Source: Isola, Torralba, Freeman

PS3: intro to machine learning



- Object recognition with small images
- Hand-crafted image features
- K-nearest neighbor
- Logistic regression with SGD
- Cross-validation

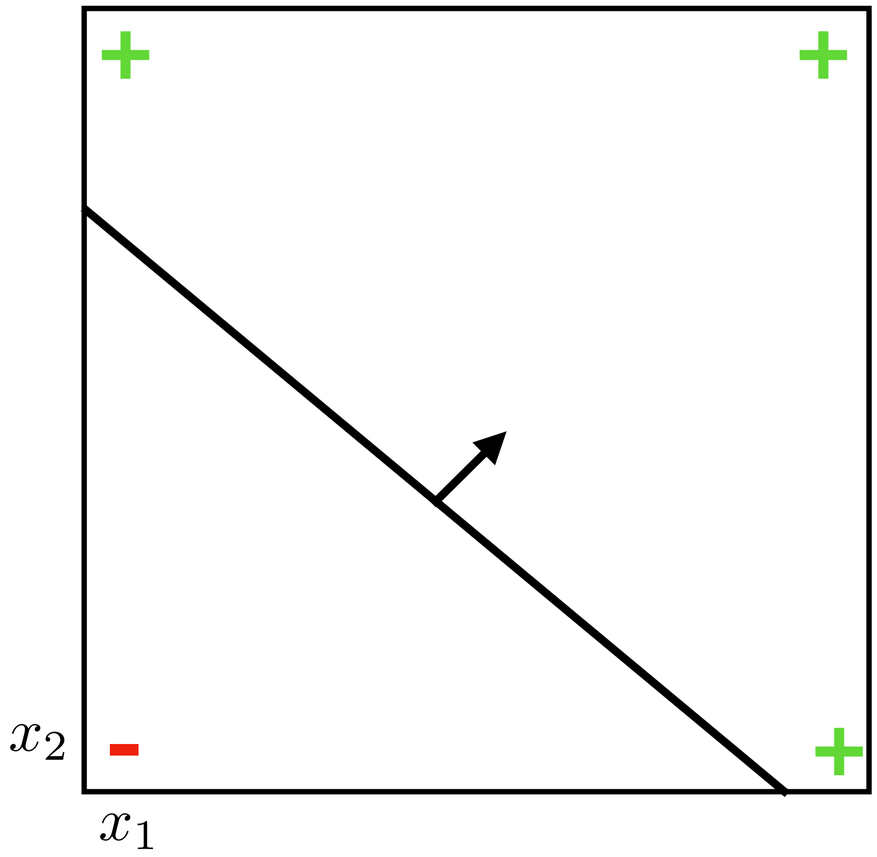
Limitations to linear classifiers



x_1 \ x_2	0	1
0	0	0
1	0	1

AND

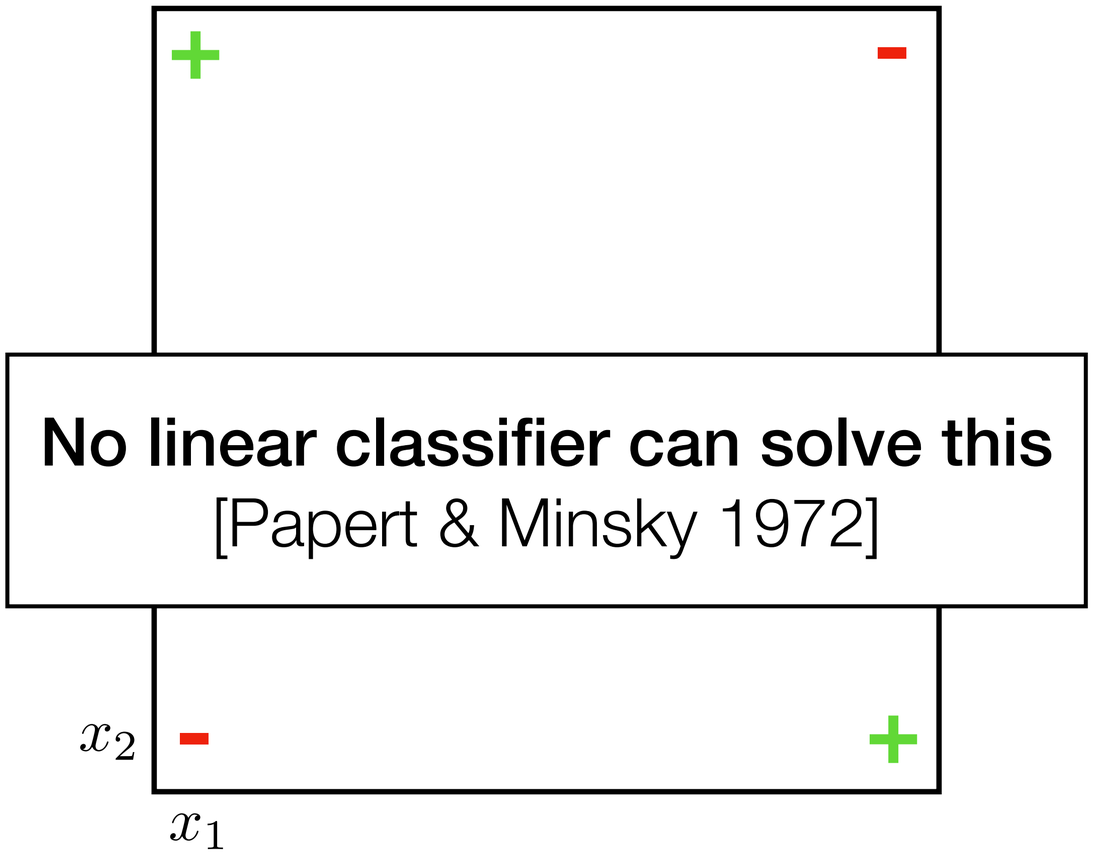
Limitations to linear classifiers



x_1 \ x_2	0	1
0	0	1
1	1	1

OR

Limitations to linear classifiers

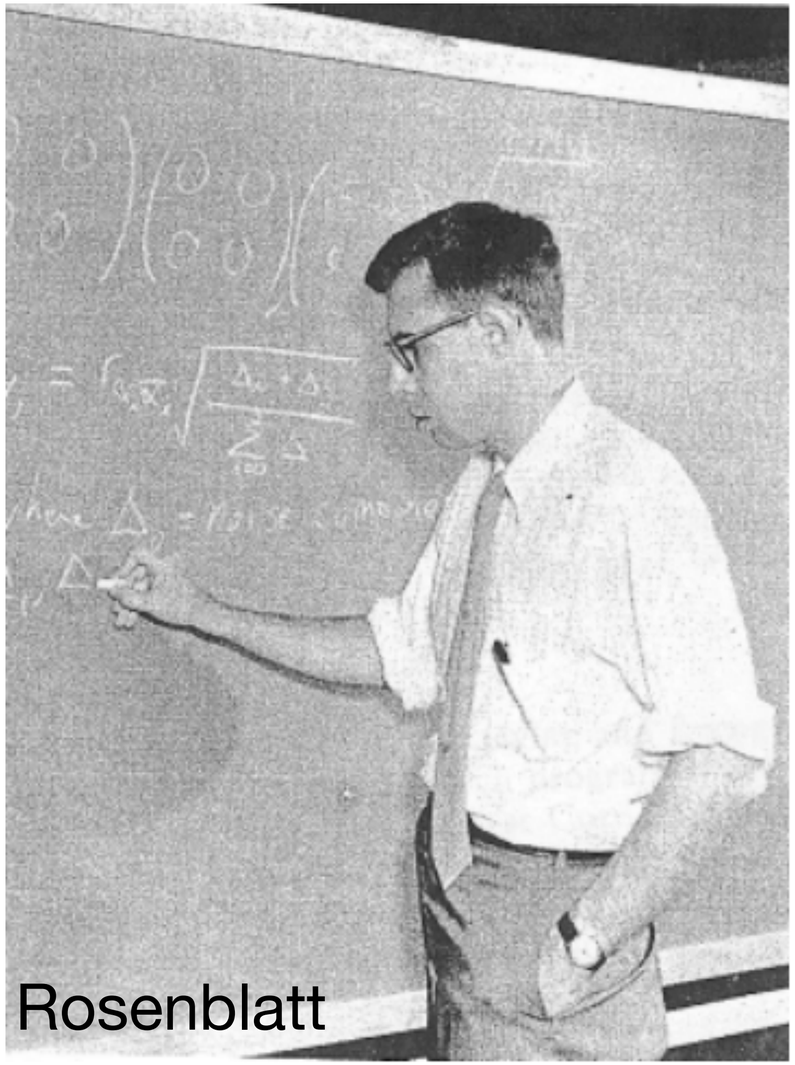


x_1 \ x_2	0	1
0	1	0
1	0	1

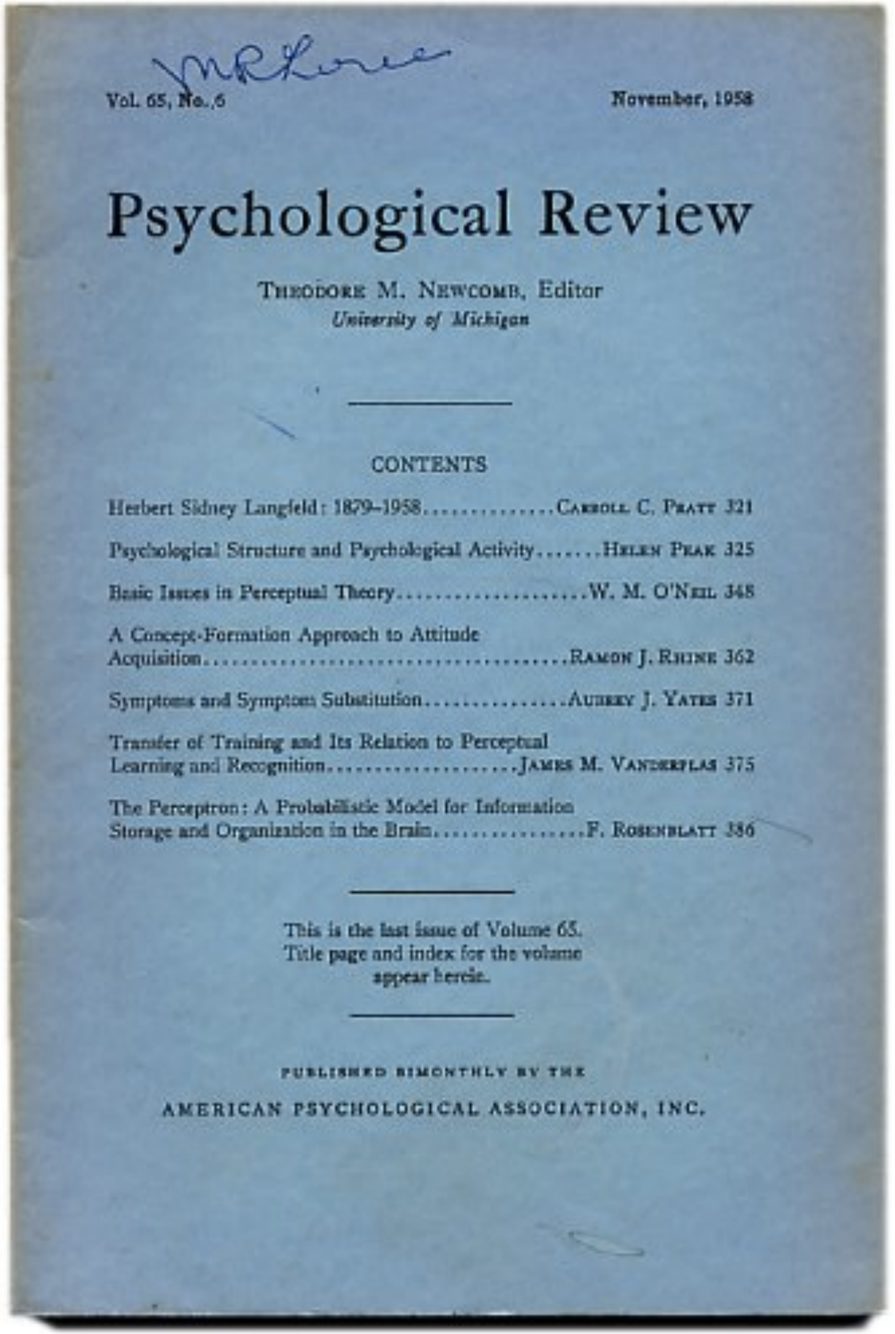
XOR

Neural networks

Perceptrons, 1958



http://www.ecse.rpi.edu/homepages/nagy/PDF_chrono/2011_Nagy_Pace_FR.pdf. Photo by George Nagy



Vol. 65, No. 6 November, 1958

Psychological Review

THEODORE M. NEWCOMB, Editor
University of Michigan

CONTENTS

Herbert Sidney Langfeld: 1879–1958 ..... CARROLL C. PRATT 321

Psychological Structure and Psychological Activity ..... HELEN PEAK 325

Basic Issues in Perceptual Theory ..... W. M. O'NEIL 348

A Concept-Formation Approach to Attitude Acquisition ..... RAMON J. RHINE 362

Symptoms and Symptom Substitution ..... AUBREY J. YATES 371

Transfer of Training and Its Relation to Perceptual Learning and Recognition ..... JAMES M. VANDERPLAS 375

The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain ..... F. ROSENBLATT 386

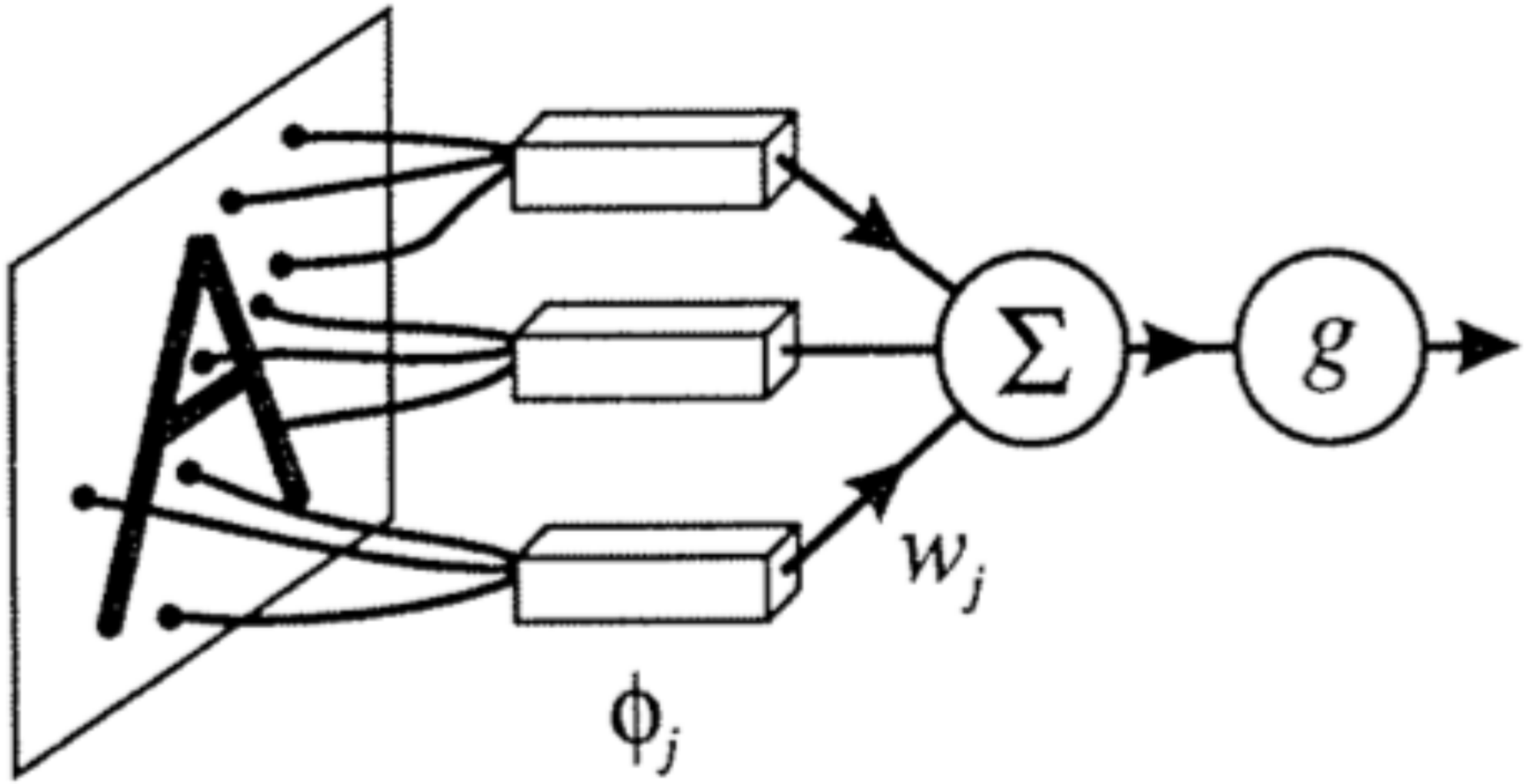
This is the last issue of Volume 65.
Title page and index for the volume appear herein.

PUBLISHED BIMONTHLY BY THE
AMERICAN PSYCHOLOGICAL ASSOCIATION, INC.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.3398&rep=rep1&type=

Source: Isola, Torralba, Freeman

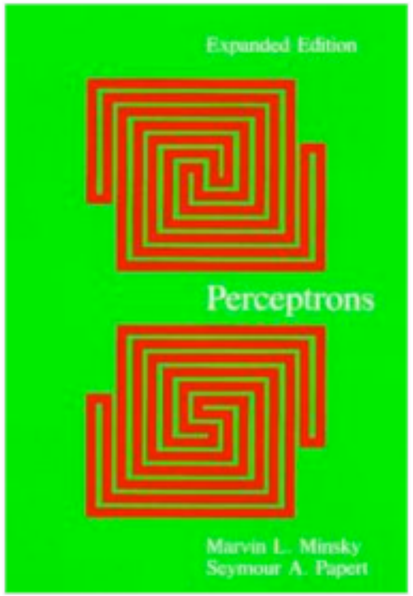
Perceptrons, 1958



Very similar to the linear models we've seen.

Source: Isola, Torralba, Freeman

Minsky and Papert, Perceptrons, 1972



Perceptrons, expanded edition

An Introduction to Computational Geometry

By Marvin Minsky and Seymour A. Papert

Overview

Perceptrons - the first systematic study of parallelism in computation - has remained a classical work on threshold automata networks for nearly two decades. It marked a historical turn in artificial intelligence, and it is required reading for anyone who wants to understand the connectionist counterrevolution that is going on today.

Artificial-intelligence research, which for a time concentrated on the programming of ton Neumann computers, is swinging back to the idea that intelligence might emerge from the activity of networks of neuronlike entities. Minsky and Papert's book was the first example of a mathematical analysis carried far enough to show the exact limitations of a class of computing machines that could seriously be considered as models of the brain. Now the new developments in mathematical tools, the recent interest of physicists in the theory of disordered matter, the new insights into and psychological models of how the brain works, and the evolution of fast computers that can simulate networks of automata have given *Perceptrons* new importance.

Witnessing the swing of the intellectual pendulum, Minsky and Papert have added a new chapter in which they discuss the current state of parallel computers, review developments since the appearance of the 1972 edition, and identify new research directions related to connectionism. They note a central theoretical challenge facing connectionism: the challenge to reach a deeper understanding of how "objects" or "agents" with individuality can emerge in a network. Progress in this area would link connectionism with what the authors have called "society theories of mind."

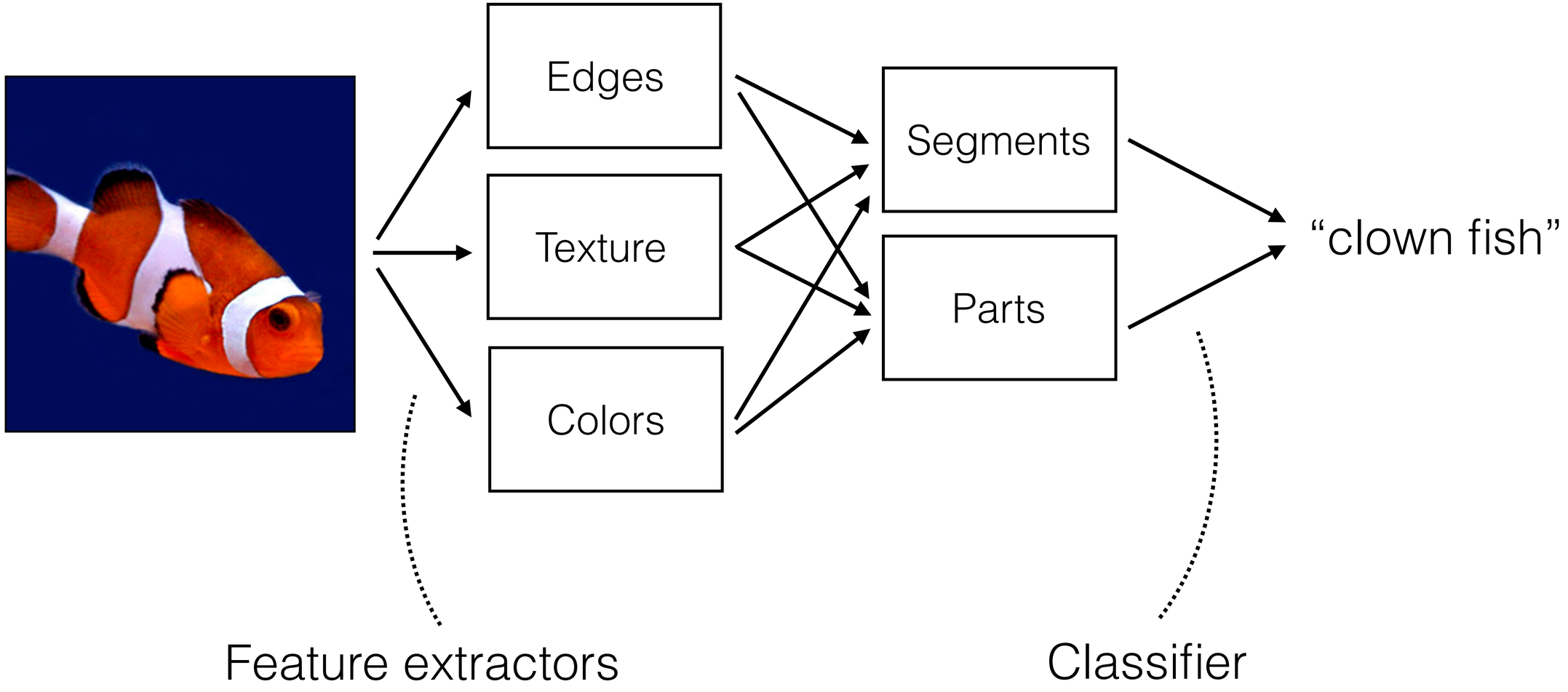
FOR BUYING OPTIONS, START HERE

Select Shipping Destination

Paperback | $35.00 Short | £24.95 |
ISBN: 9780262631112 | 308 pp. | 6 x 8.9 in | December 1987

Source: Isola, Torralba, Freeman

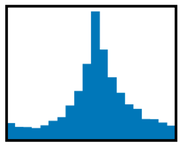
“Classic” recognition **without** neural nets



Source: Isola, Torralba, Freeman

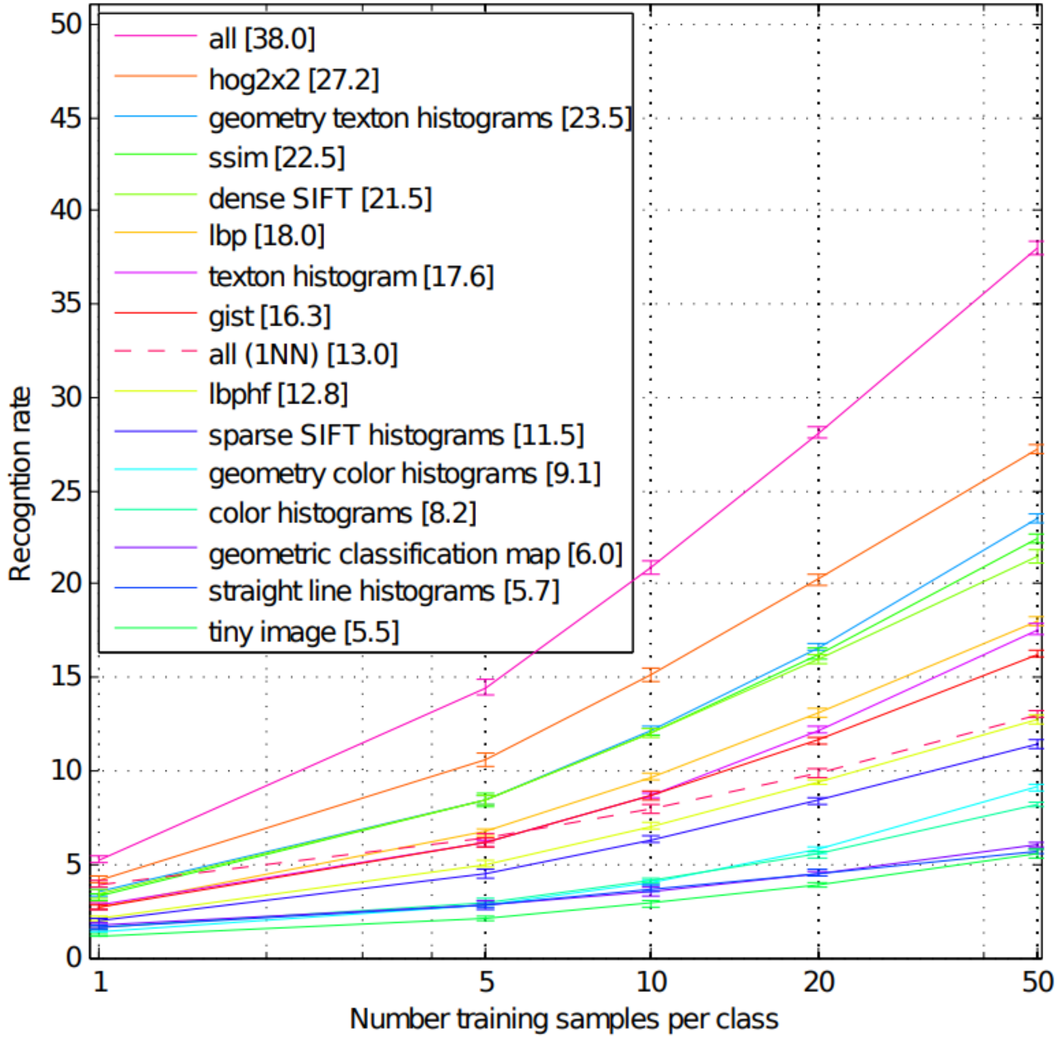
“Classic” recognition **without** neural nets

Hand-crafted features



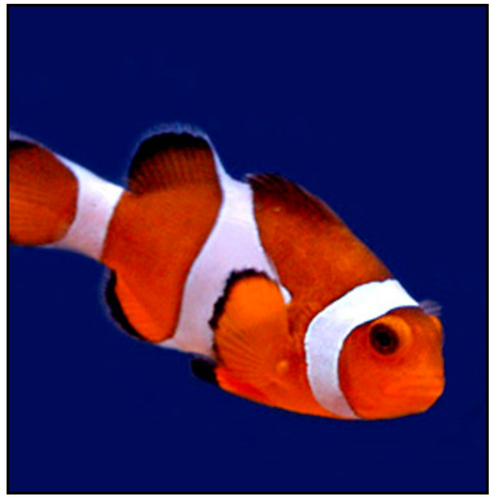
θ

edge orientation histogram



[Xiao et al., “SUN database”, 2010]

“Classic” object recognition **without** neural nets



Edges

Texture

Colors

Segments

Parts

Learned

“clown fish”

$\phi_k(x)$

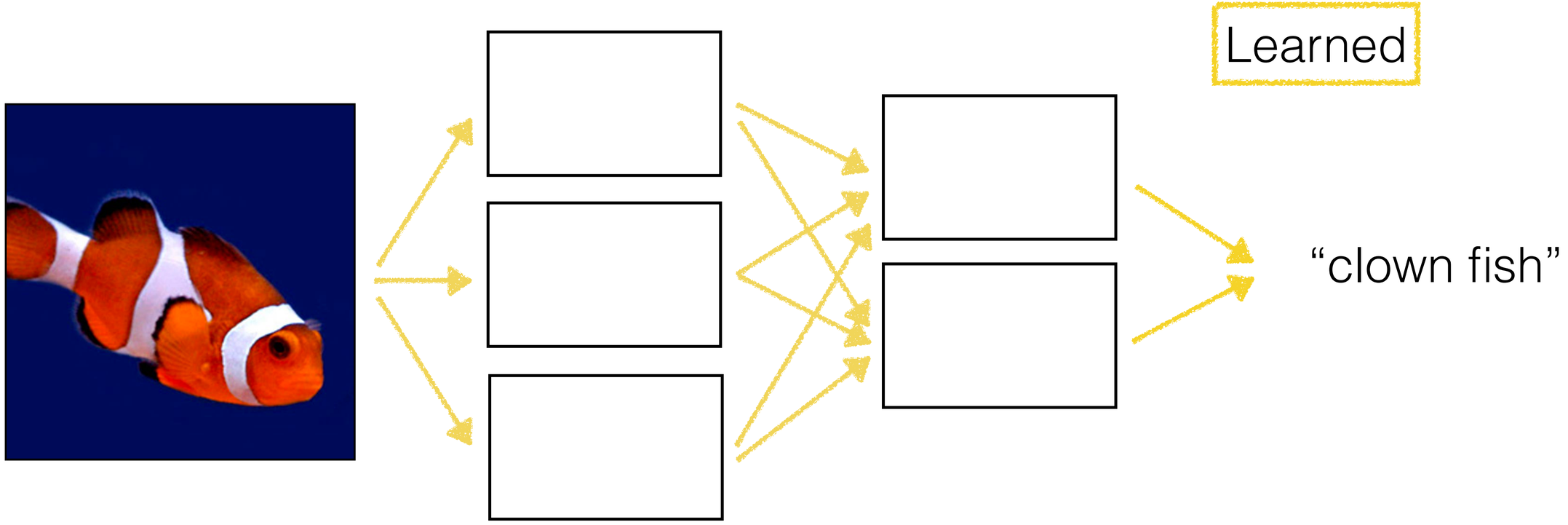
$$f_\theta(x) = \sum_{k=1}^{K} \theta_k \phi_k(x)$$

Feature extractors

Classifier

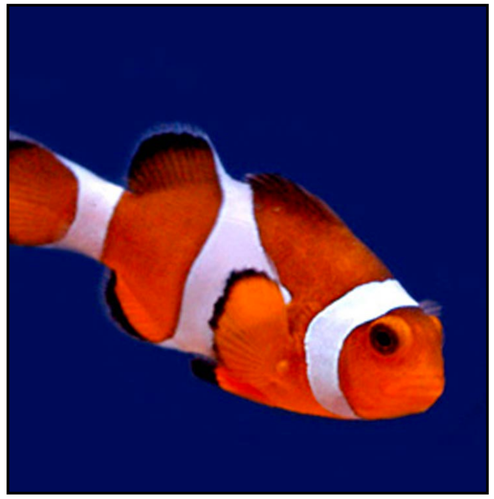
Source: Isola, Torralba, Freeman

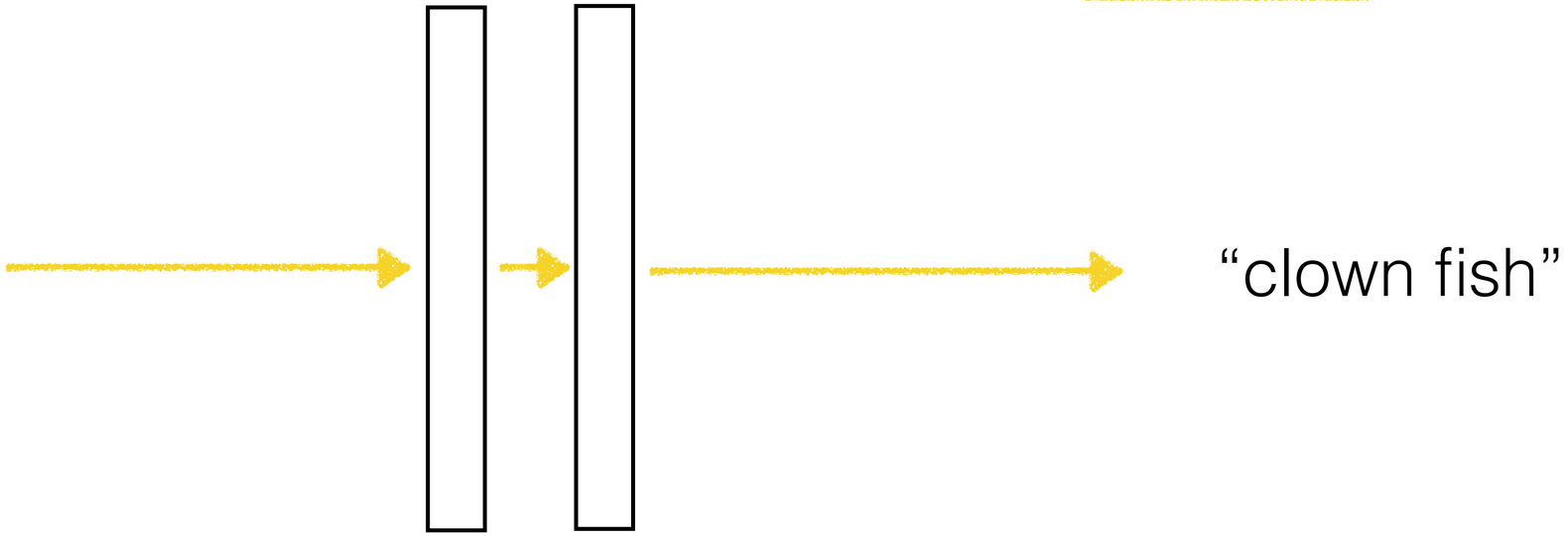
Object recognition



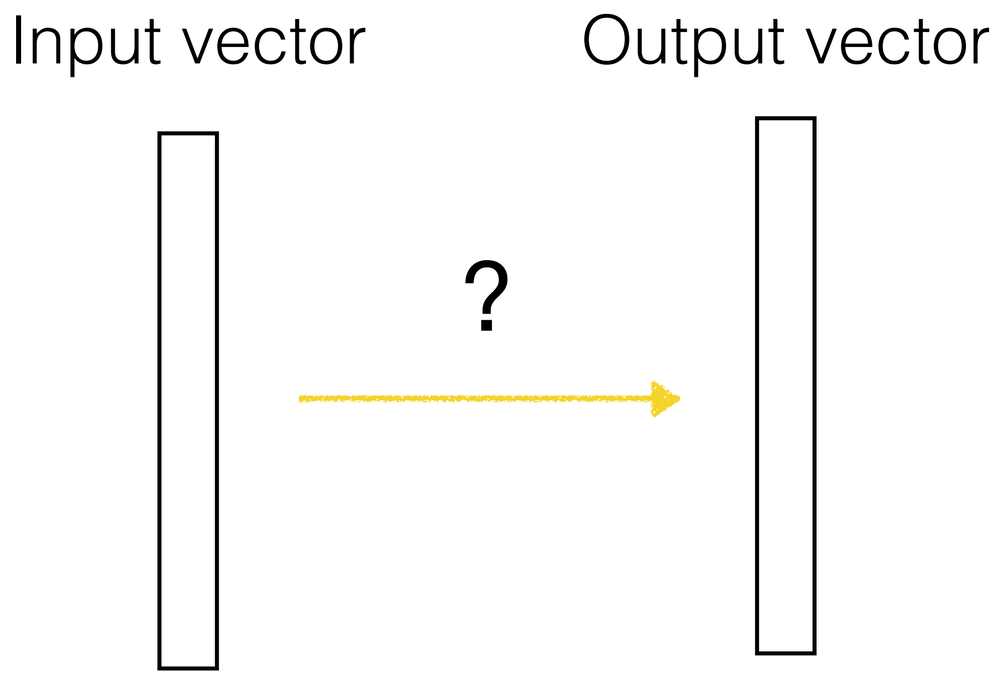
Source: Isola, Torralba, Freeman

Object recognition





Neural net

Source: Isola, Torralba, Freeman

Computation in a neural net