Learning from Examples & Deep Learning

Margrit Betke

CS 640

Fall 2020

1st 10 slides are based on Russell & Norvig, 3rd edition, Sections 1, 2, and 4.

Forms of Learning

AI system "learns" if it improves its performance *based on observations & feedback from its environment*

Unsupervised learning (=clustering):

Input: vector of attributes. No explicit feedback.

Supervised learning:

Input: vector of attributes. Feedback = output of continuous or discrete value(s) = labels of input examples.

Reinforcement Learning:

Actions are rewarded or punished.

In 640: Supervised Learning

Training set = N example input-output pairs

$(x_1,y_1),\ (x_2,y_2),\ ....,\ (x_N,y_N)$

where each y_j was generated by an unknown function f, such that $f(x) = y$. Function f needs to be learned.

- The AI system finds a function h that approximates f. For example, the AI system trains a neural net that computes $h(x_j)=y_j$ for all examples in the training set.
- There are no guarantees that new inputs $h(x_{new}) \approx f(x_{new})$.
- To measure accuracy (Is $h \approx f$?), we use a test set of labeled examples = input-output pairs ($\neq$ training set!):

A neural net is trained well if $h(x_{test}) \approx y_{test}$ for all test example pairs (x_{test}, y_{test}).

Classification versus Regression

Depending on the type of output, the learning problem is a

- Classification problem:

  Output values: number of classes (discrete, finite)

- Regression problem:

  Output values are numbers, e.g., tomorrow's temperature

Occham's Razor

= Law of succinctness

Which hypothesis among h_1, h_2, h_3 ... should the AI system choose?

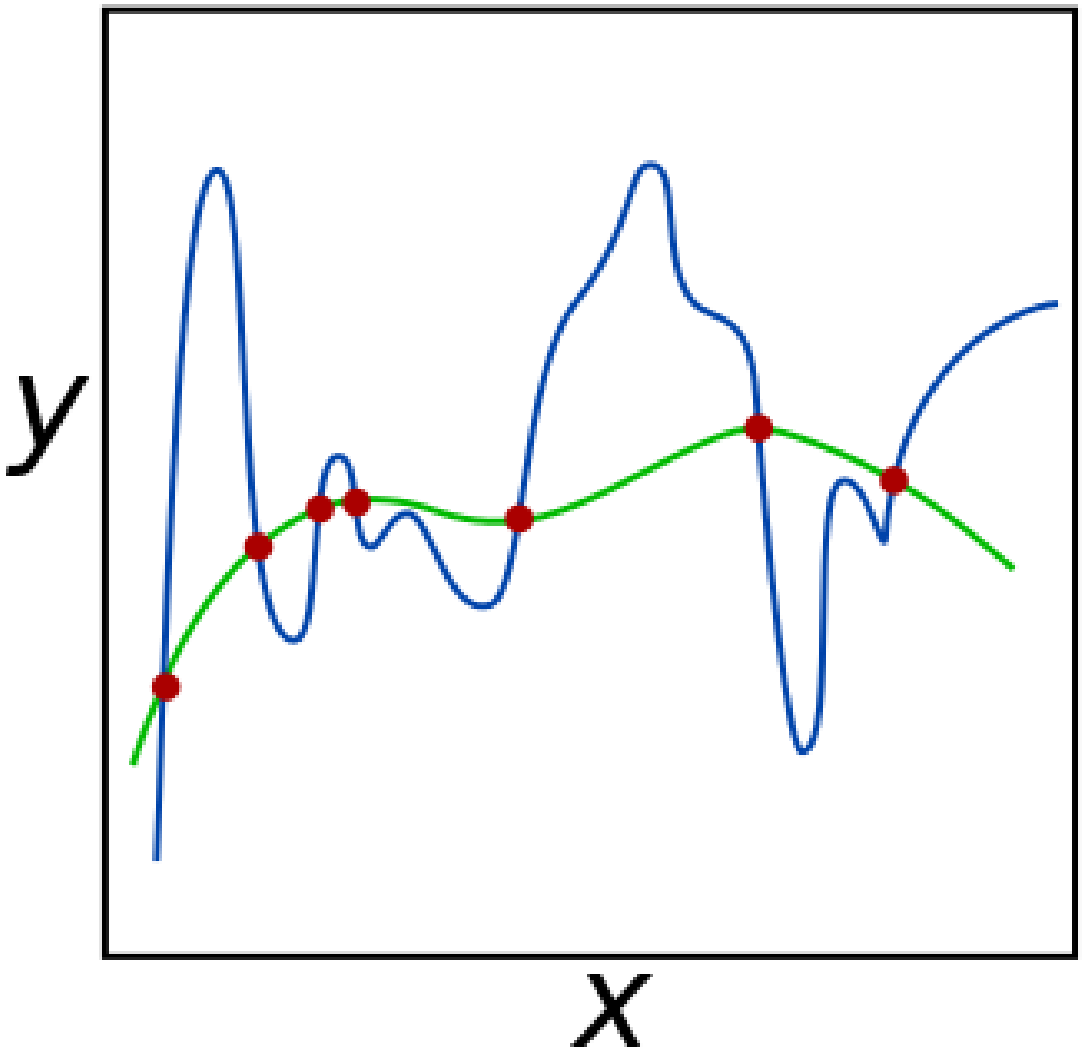
Choose the simplest hypothesis consistent with the data.

The simplest explanation will be the most plausible until evidence is presented to prove it false.

Example: Prefer a degree-1 polynomial (line) over a degree-7 polynomial

Trade-off between complex hypothesis that fit training data well and simpler hypotheses that may generalize better (and can typically be computed faster)

Occam's Razor: Choose green over blue model for h



Source: Wikipedia

Overfitting

- Avoid choosing an excessively complex learning system= model= hypothesis=neural net h.
- h is too complex if it has too many parameters relative to the number of observations.
- A model which has been overfitted will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.
- Higher-degree polynomials or complicated neural nets with many hidden layers and nodes fit the data better but may lead to overfitting.

Overfitting

- Avoid choosing an excessively complex learning system= model= hypothesis=neural net h.
- h is too complex if it has too many parameters relative to the number of observations.
- A model which has been overfit will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.
- Higher-degree polynomials or complicated neural nets with many hidden layers and nodes fit the data better but may lead to overfitting.

Solutions:

- Use “wrapper” to enumerate models h according to model size (e.g., number of nodes in neural net h). Select model with smallest error.
- Feature selection: Simplify model by discarding irrelevant attributes (dimensionality reduction).
- Minimum description length: Select model with smallest number of bits required to encode program and data.

Loss Functions: SPAM Example

Loss value $L(y_{true}, y)$

= cost of misclassifying email:

A "false positive," e.g. hypothesize "non-spam" but it is truly "spam" L(spam,non-spam) = 1

Annoying but simply delete email.

A "false negative," e.g. hypothesize "spam" but it is truly "non-spam" L(non-spam, spam) = 10

Much worse, you may miss an important email.

Loss Functions

- Absolute value loss: $L_1(y_{true},y) = |y_{true}-y|$
- Squared error loss = Euclidean loss:
  $L_2(y_{true},y) = (y_{true}-y)^2$
- 0/1 loss: $L_{0/1}(y_{true},y) = 0$ if $y_{true} = y$, else 1
- Find h that minimizes the empirical loss
  $\text{EmpLoss}(h) = 1/N \sum L(y_{true}, h(x))$

  (sum over a set of N examples (x,y_{true}))
- Regularization: Find h that minimizes EmpLoss(h) + λ Complexity(h)

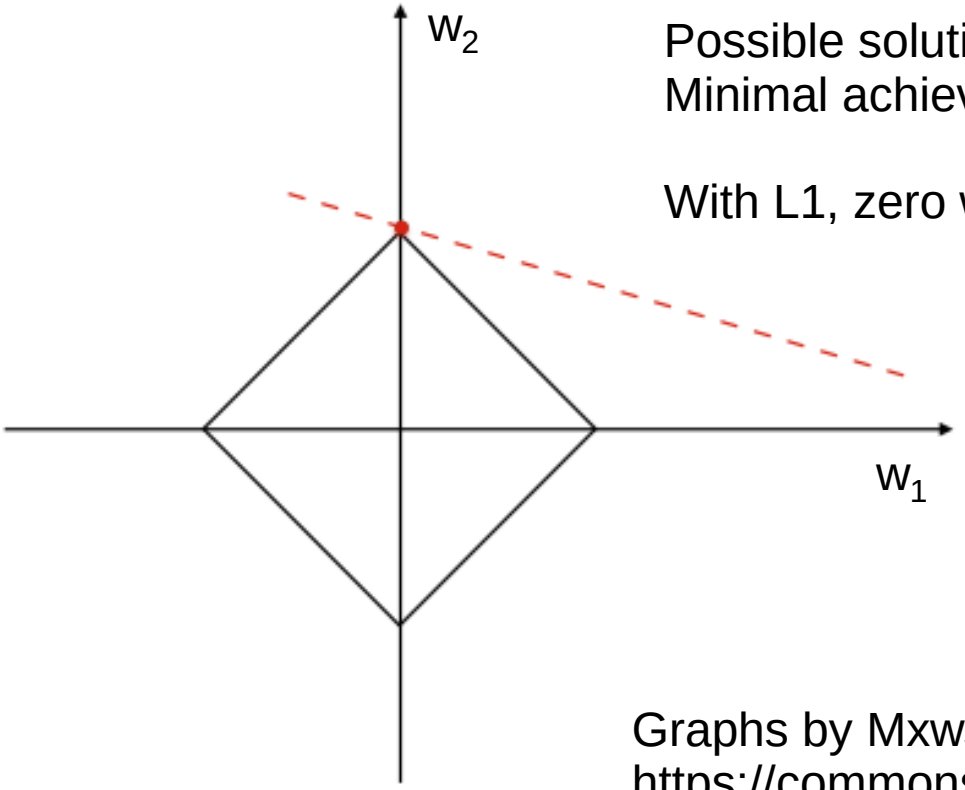
  e.g., $\text{Complexity}(h)=L_q(w) = \sum_i |w_i|^q$

Avoid Overfitting with Regularization

- Find h that minimizes EmpLoss(h) + λ Complexity(h)
- Complexity(h_w) = $L_q(w) = \sum_i |w_i|^q$

L_1

$|w_1|+|w_2| <=$ constant

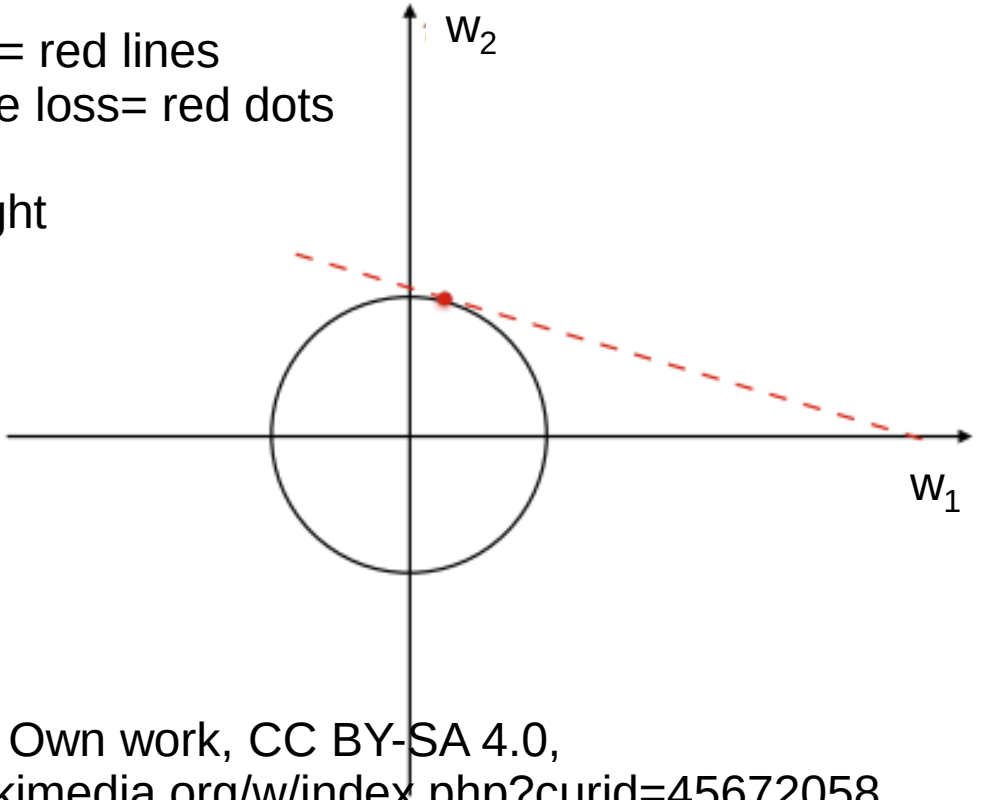


Possible solutions= red lines
Minimal achievable loss= red dots

With L1, zero weight

L_2

$w_1^2+w_2^2 <=$ constant



Cross-Validation

Holdout cross-validation =

Randomly split available (input,output) pairs into a training set to learn h and a test set to test the learned h.

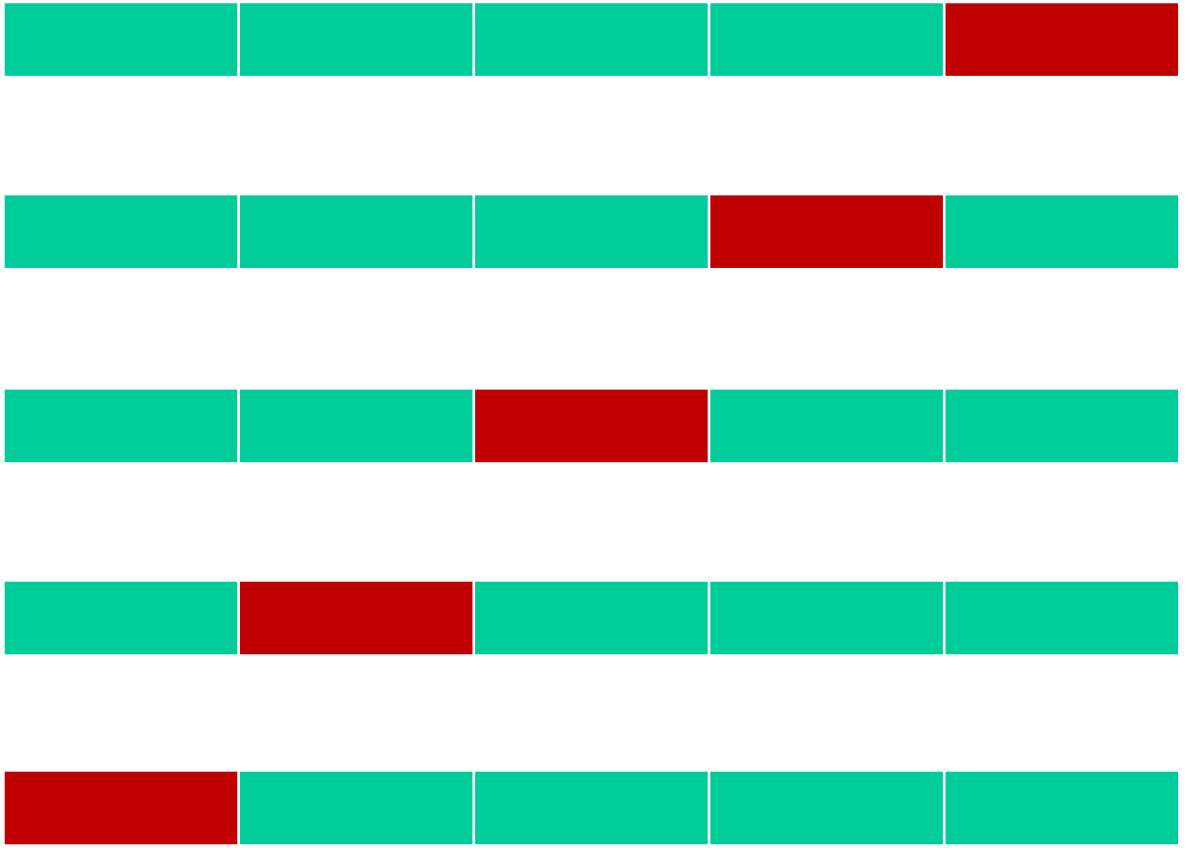
k-fold cross-validation =

- Split data into k equal subsets.
- Perform k rounds of learning. Each round leaves 1/k examples out of the training set that can then be used as the test set.
- The average test set score should be a better estimate than a single score (need to keep k h's around for prediction). Typically, k=5 or 10.

Leave-one-out cross validation: k=N.

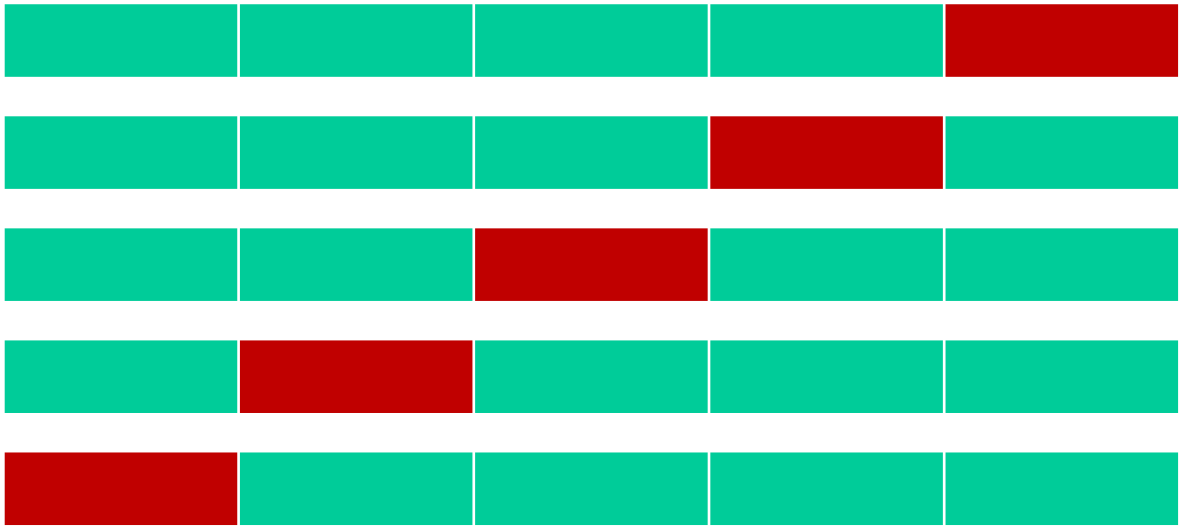
Example: 5 Fold Cross-Validation

5 Folds of Labeled Dataset: **Training** & **Testing**



Example: 5 Fold Cross-Validation

1. Create 5 folds of the labeled dataset for **training** & **testing**



2. Train 5 AI models
3. Conduct ROC analysis for each model
4. Report average performance (accuracy etc.)

Example: 5 Fold Cross-Validation

What if you were simply lucky in your **training** & **testing** folds?

2. Randomly create a different set of 5 folds & report average performance

3. Do this n times & report 1/n of average performance

Convolutional Neural Nets

Multi-layer neural networks that recognize visual (or other 2D) patterns:

- directly from pixel images with minimal preprocessing
- with robustness to distortions and simple geometric transformations

Network architecture:

Convolutional layers with local connections (e.g., for small sub-image processing), optionally followed by fully connected layers.

Famous examples from the 1990s:

- LeNet-5 for handwritten and machine-printed character recognition
- ALVINN and MANIAC for autonomous driving
- Also discussed in CS 640: SexNet, CommodityNet, NetTalk

LeNet5

Handwritten Digit Recognition (= ZIP codes)

Yann LeCun 1990, now Director of AI Research, Facebook

28x28 image input,

5-layer network

4 hidden layers with 4, 4, 12, and 12 nodes

See online demo

Autonomous Vehicles

ALVINN 1992

960 inputs, 3 layers, 4 hidden units, 50 outputs

MANIAC 1993

2 ALVINN nets, 8 hidden units

See papers online

SexNet

Gender Recognition from Facial Images, 1991

Input: Aligned face: Size 30x30 pixels, 12 pixels between eyes, face is strictly vertical, normalized greyscale pixels [0:1]

1 hidden layer with 40 nodes (they tried 2, 5, 10, 20 also)

Output: One node: >0.5 = “male” <0.5 = “female”

Training: 2,000 epochs, 90 images (80+10)

Performance: 93% (human 88%)

1 male image classified as female, turned out wrong label

See paper online

Commodity Trader

Trained on 1 year past data: Commodities open, close, high, low, and volume trade data plus weather and season info

Predicts market positions

$1,000 investment → $10,301 outcome

Various networks tested

16 hidden nodes

See paper online.

Pronunciation of English

NetTalk 1987 – Listen to recording

Network with 80 hidden units

Input: Sequence of 5 characters in sliding window, e.g.

hello world

29 input nodes (26 letters, plus blank, period, etc)

Output layer: Nodes for pronunciation (un)voiced

Why difficult?

- “c” = [k] like cat or [s] like cent
- “lead” includes [e] as bed and [i:] as bead

NetTalk Training

1024 word text, 95% accuracy

50 passes: Initially babbling speech

High-pitched speech generator = child like

100 passes: Separate words

500 passes: Distinction consonants/vowels

Improving until understandable

1500 passes: Nearly perfect

Critics

Face Recognition

Turk and Pentland 1991

1 hidden layer

Input: each face pixel (normalized, cropped)

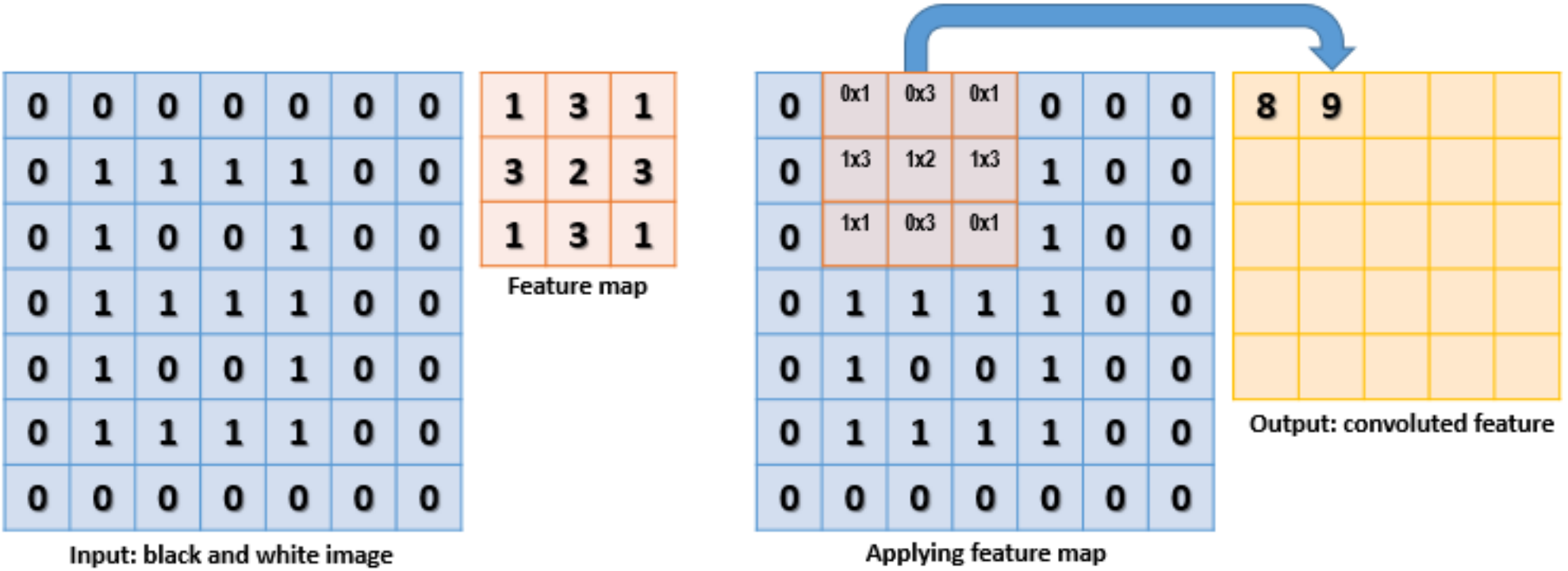
Output: “Eigenspace projection”

1D Discrete Convolution

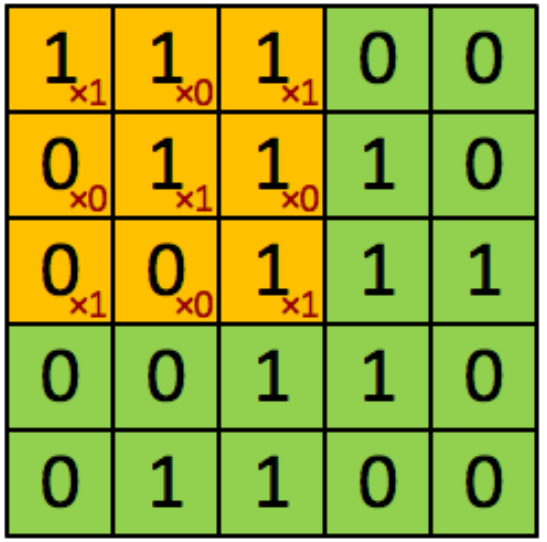
$$(f * g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$
$$= \sum_{m=-\infty}^{\infty} f[n-m]\, g[m].$$

2D generalization:
f = input image, g = template image (or CNN function)

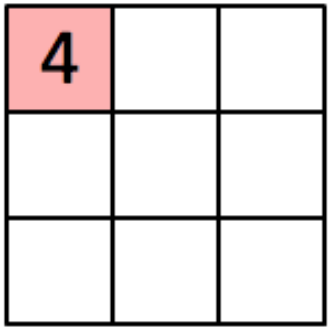
2D Convolution Example



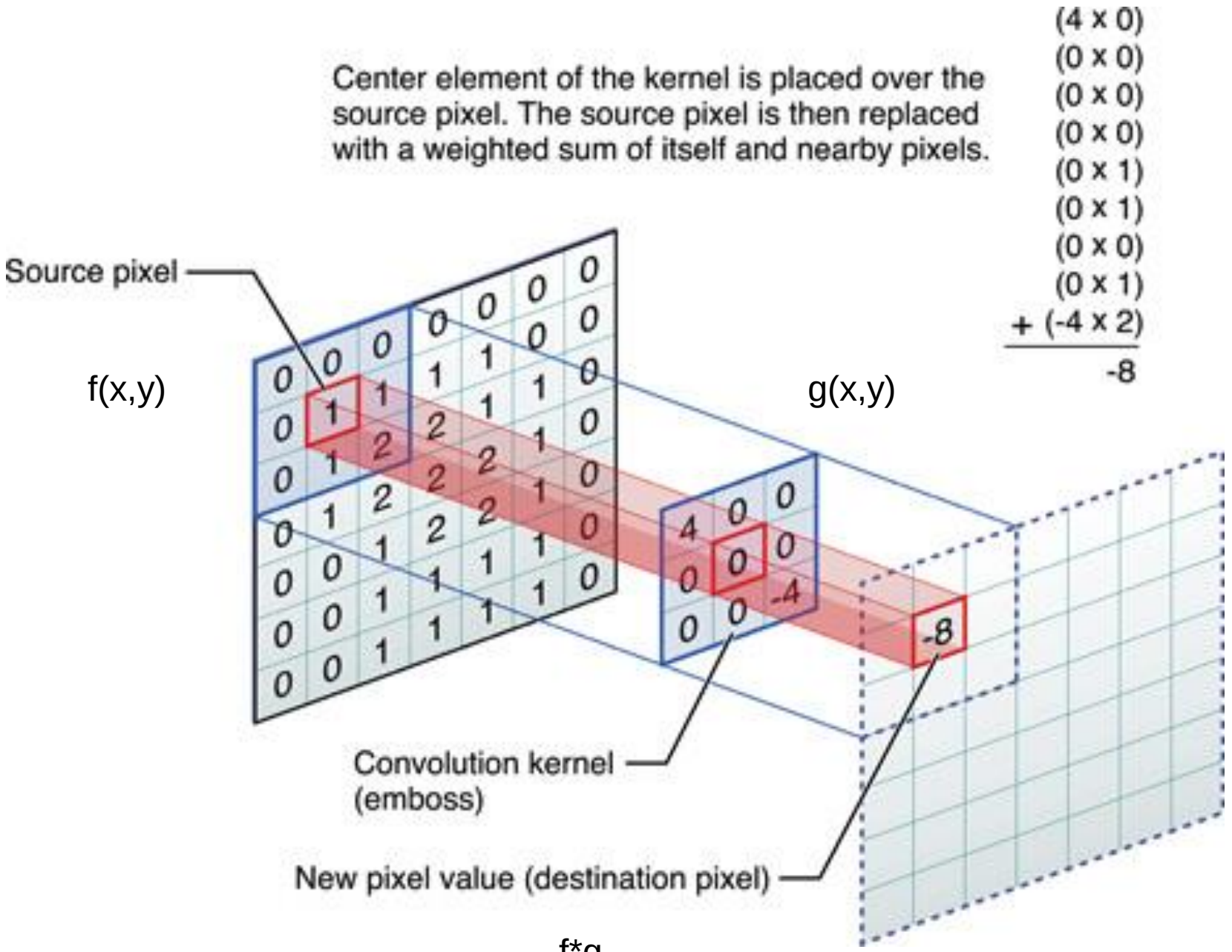
Convolution example



Image



Convolved Feature


Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
f(x,y)
g(x,y)
Convolution kernel (emboss)
New pixel value (destination pixel)
f*g

2012 Revolution: Deep Learning

Learning Algorithm that attempts to model high-level abstractions in data by using model architectures composed of multiple non-linear transformations, e.g., neural networks.

Initial successes:

- Winner of traffic sign competition, 2011: 6 layers
- Winner of the "segmenting neuronal structures" competition, 2012: 10 layers
- Winner of ImageNet competition 2012: 7 layers

Used in industry for many applications, e.g., Facebook automatically tags pictures with people's names

Deep NN Training (vs. Traditional)

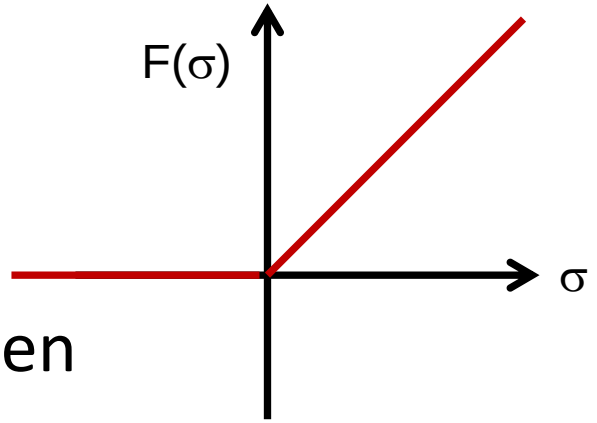
Stochastic Gradient Descent (SGD)

Each parameter update is computed with respect to a few training examples (a “mini batch”) as opposed to a single group of all training examples

Rectified Linear Units (RLUs)

$F(\sigma)=\max(0,\sigma)$

Used as a threshold function in each hidden node instead of a sigmoid function



Many other Activation Functions

Dropout Trick

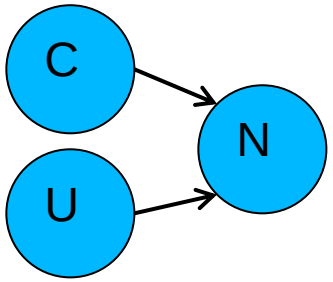
Dropout Trick for Training Deep Nets

What is it?

Randomly remove nodes ("let them drop out") during training

Why?

Enables remaining "underperforming nodes" to be retrained



Initial situation: Node C = 90% correct on training data, Node U = 10% correct (underperforming), Node N uses C's output only (weight $w_{U->N}=0$)

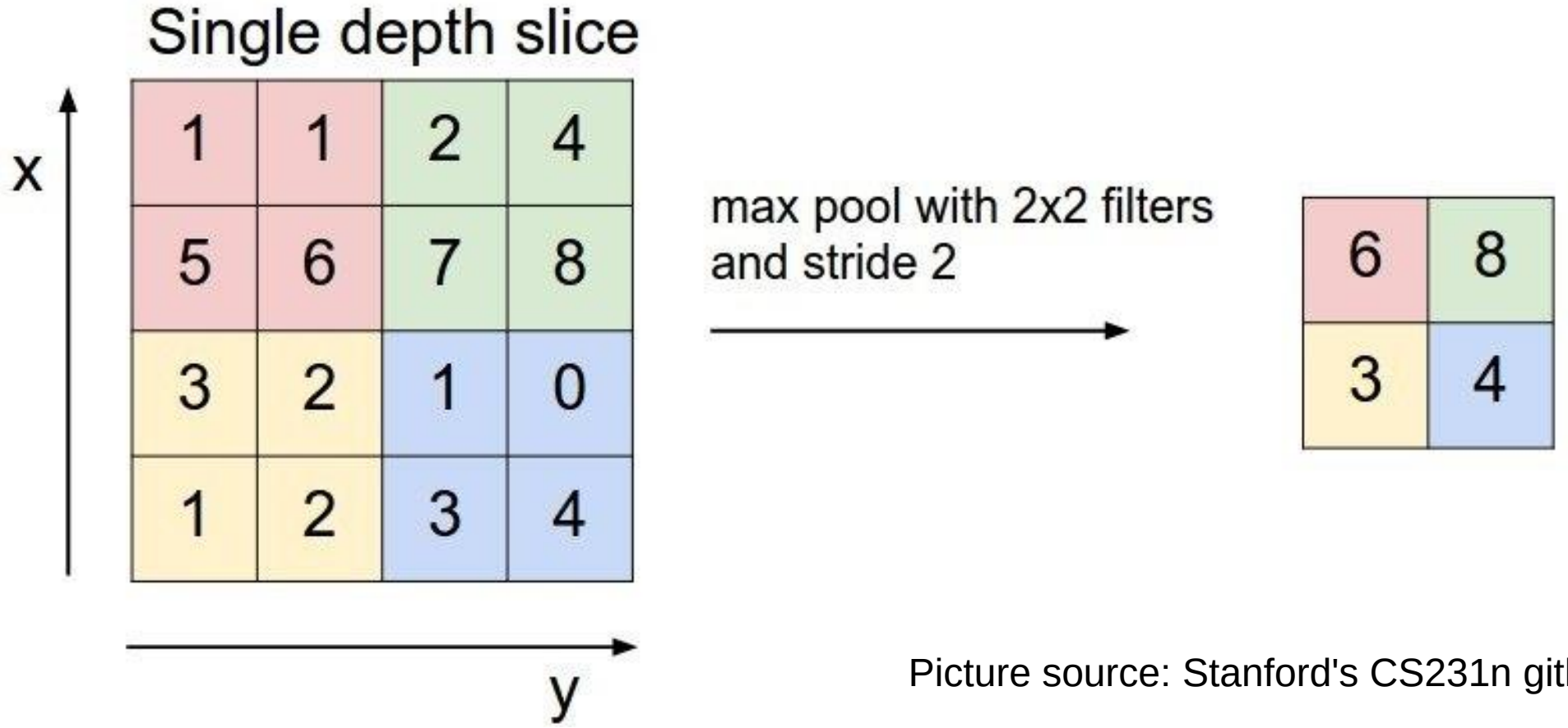
Dropout of Node C forces Node N to use Node U's results, which in turn forces Node U's weights to change and Node U to perform better.

Tricks used in Deep NN Architectures

Max Pooling

What? Why? Downsample representation

(e.g., pixels to objects)



Picture source: Stanford's CS231n github

Tricks used in Deep NN Architectures

Softmax function is applied to each node in the last layer

What?

$$F_j(\mathbf{x}) = \exp(x_j) / \sum_{k=1}^{K} \exp(x_k) \quad \text{where } j=1,.., K$$

Why?

Converts the output values to [0:1] so that they can be interpreted as probabilities.