Lesson 1

Overview of Computer Vision and the History of Computer Vision

-Computer Vision has been a topic(research subject) for several years.

Recent decade’s progress in computation and dataset boosts the development

Yet many achievements in this field have been impressive, but a whole lot bunch of questions is yet to be solved.(And as a student who is determined to focus on this area, I should feel that this is cool because I get to work on some of these problems and enjoy the good feeling of solving problems in Computer Vision

The aim is to tell a machine to see things the way human eyes do.

-Computer Vision will better human lives

It’s a marginal area and relies on study in related areas like cs, math, psychology, physis and so on.

It will help reduce the workload of human beings, especially things related to vision.

-This course mainly focuses on neural networks(Deep Learning)

Since the AlexNet was put forward, CNN has been doing so well in the task of image classification

In ILSVRC2017, the accuracy of someNet passed the accuracy of human eyes.

This is exactly what I need!

Lesson2

-How do computers see an image

Pixels represented by three numbers

A huge gap between the numbers and the label of a image

Problems: viewpoint , illumination, deform, pose, occlusion

def classify\_image(image):

# magics

return label

One attempt: looking at the edges of a cat—not a general approach

So, data-driven approach. Find the features of cat in many cat images

def train(images, labels):

# Machine Learning!

return model

def predict(model, test\_images):

# Use model to predict labels

return test\_labels

-Nearest Neighbors

Dumb

On CIFAR-10

Ways to compare images:

L1 distance

Efficiency: Slow at training time, fast at testing time

K nearest Neighbor

Hyperparameters: K, L1/L2, etc

Split the dataset: The good performance lies in the performance on unseen data

Split it into three parts: Train: actually train, Validation: determine the best performance, Test: use the best hyperparameters

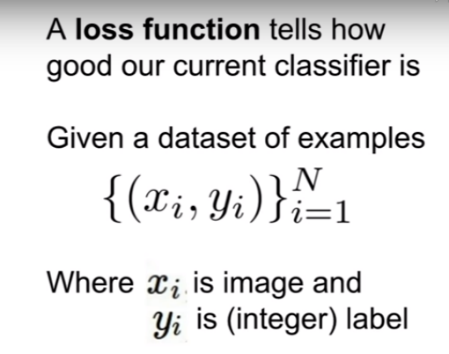
Sometimes with small datasets: cross validation

Linear Classification

Store all the parameters as W instaead of remembering all the data

f(x, W) = Wx + bias

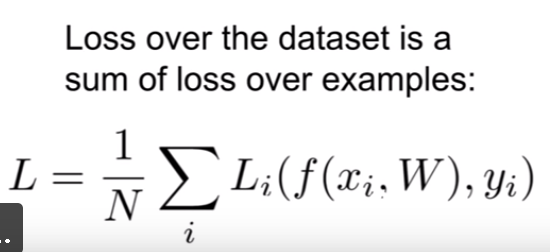
only one template for each category(average all the variations)

 Hard cases:

odd-even; distance(circles);patchy

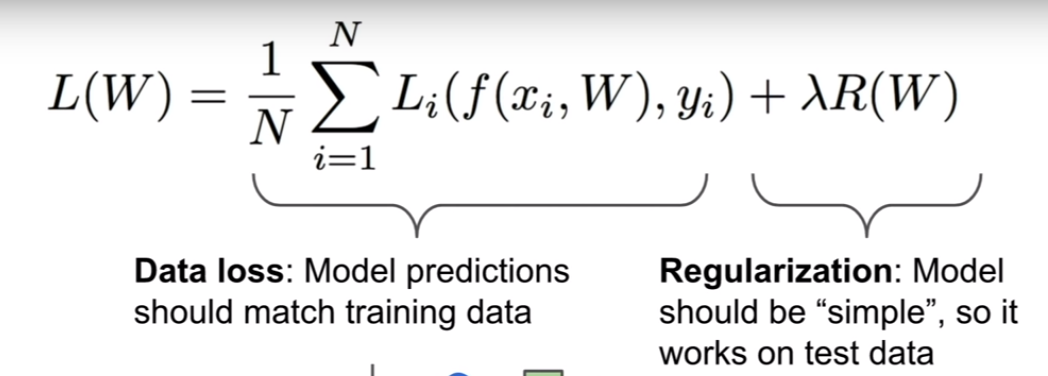
Lesson 3

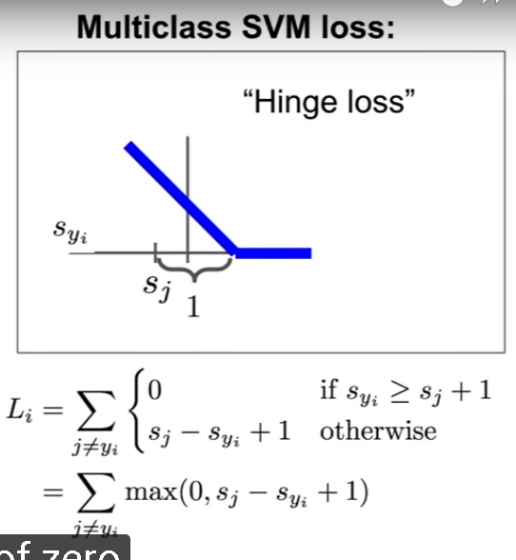
Loss function(Tell you the badness of W)



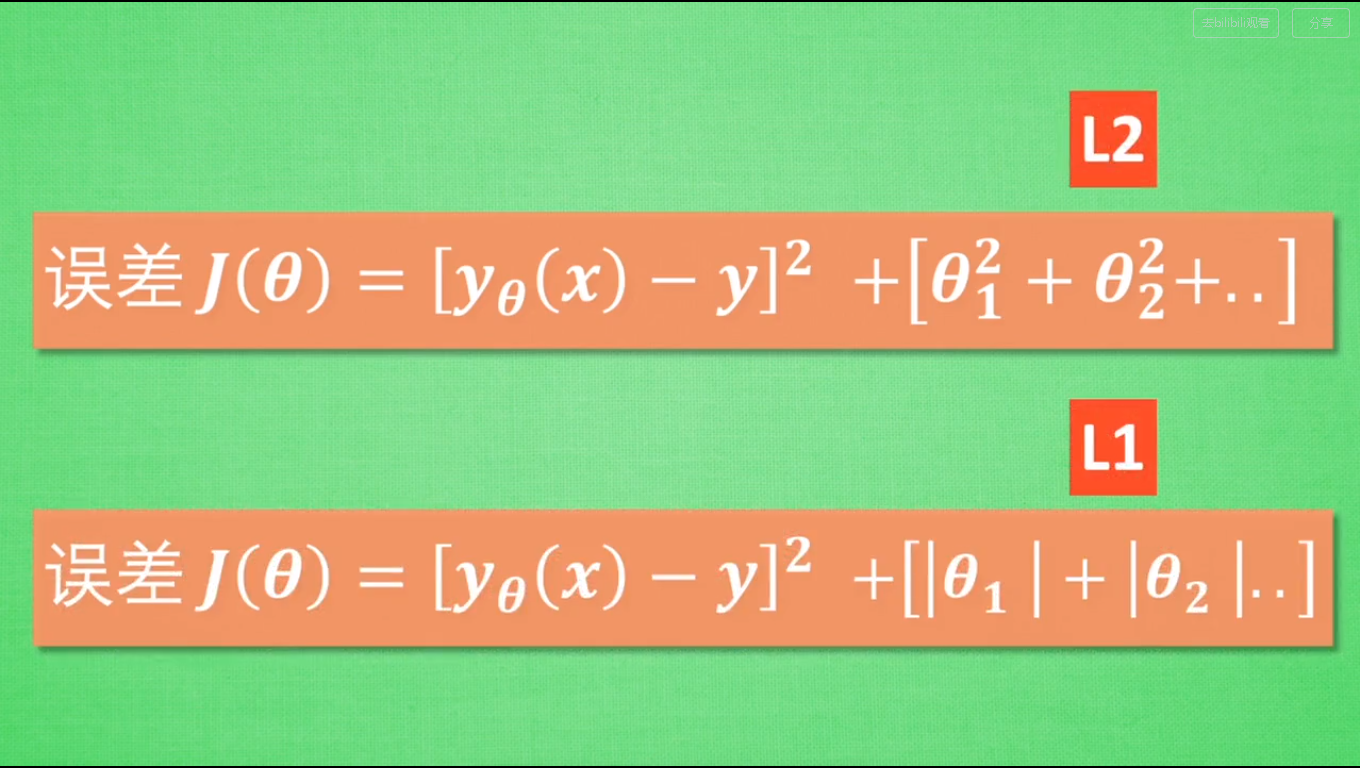
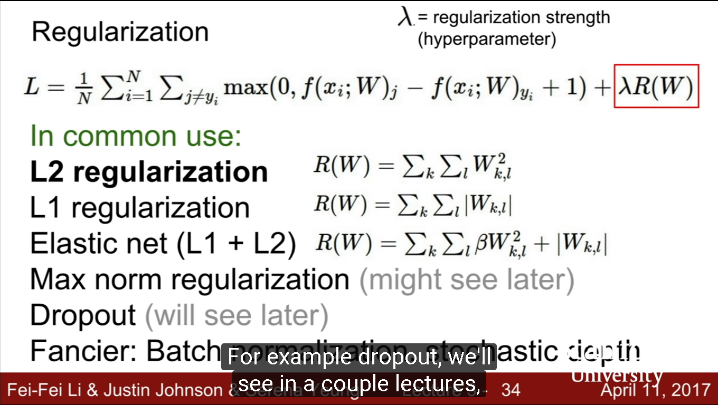
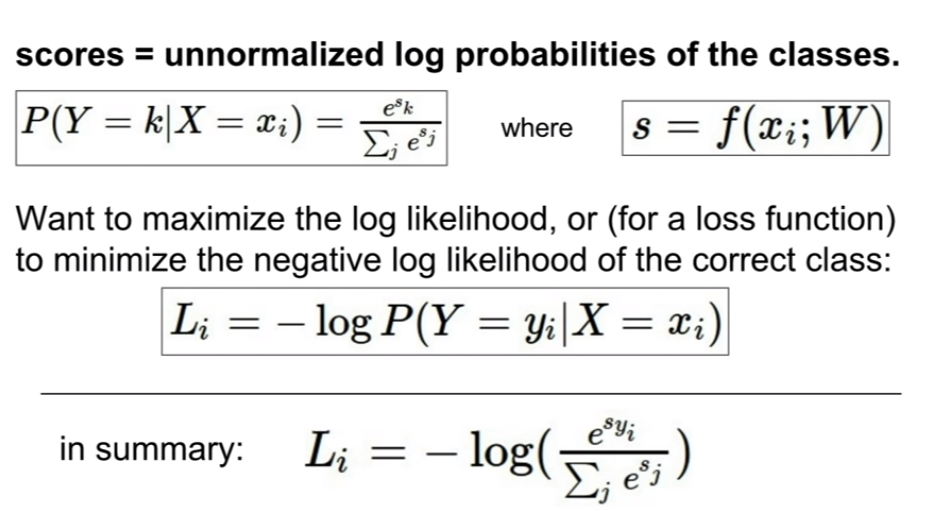
Multiclass SVM Loss:

Average of distance between the y and the prediction\_y

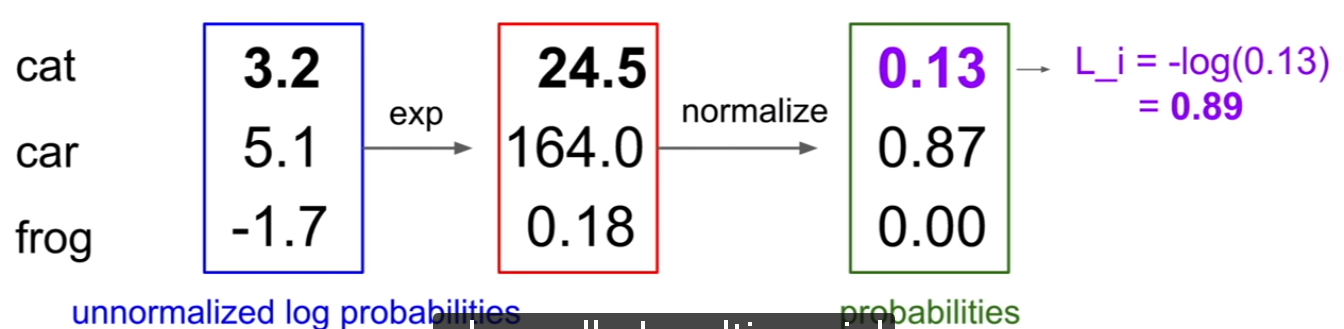
 Train from scratch: Loss = (num of catagories – 1) \* safety margin



In order not to fit only training data, add a regularization part



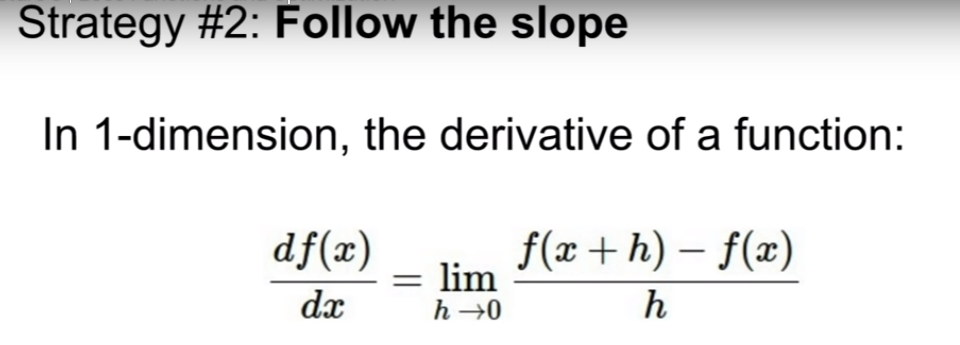
Softmax Loss



Optimization

random search: not so good

slope(gradient):



numerical gradient(slow, use to check)

analytical gradient(fast, error-prone)

->super important:Gradient Descend

update rules

stochastic gradient descent—use minibatch(32, 64, 128)

Image Features

eg. color histogram, hog

difference between features and convnets:

features: first you need to extract them and then do a classification

convnets: input=images, sort of getting the features along the way

Lesson 4

Analytical gradient—back propagation

-a neural network can be huge, it’s difficult to compute the gradient of intermediate variables

group some of the nodes together—sigmoid gate

patterns:

add—distributor

max—router

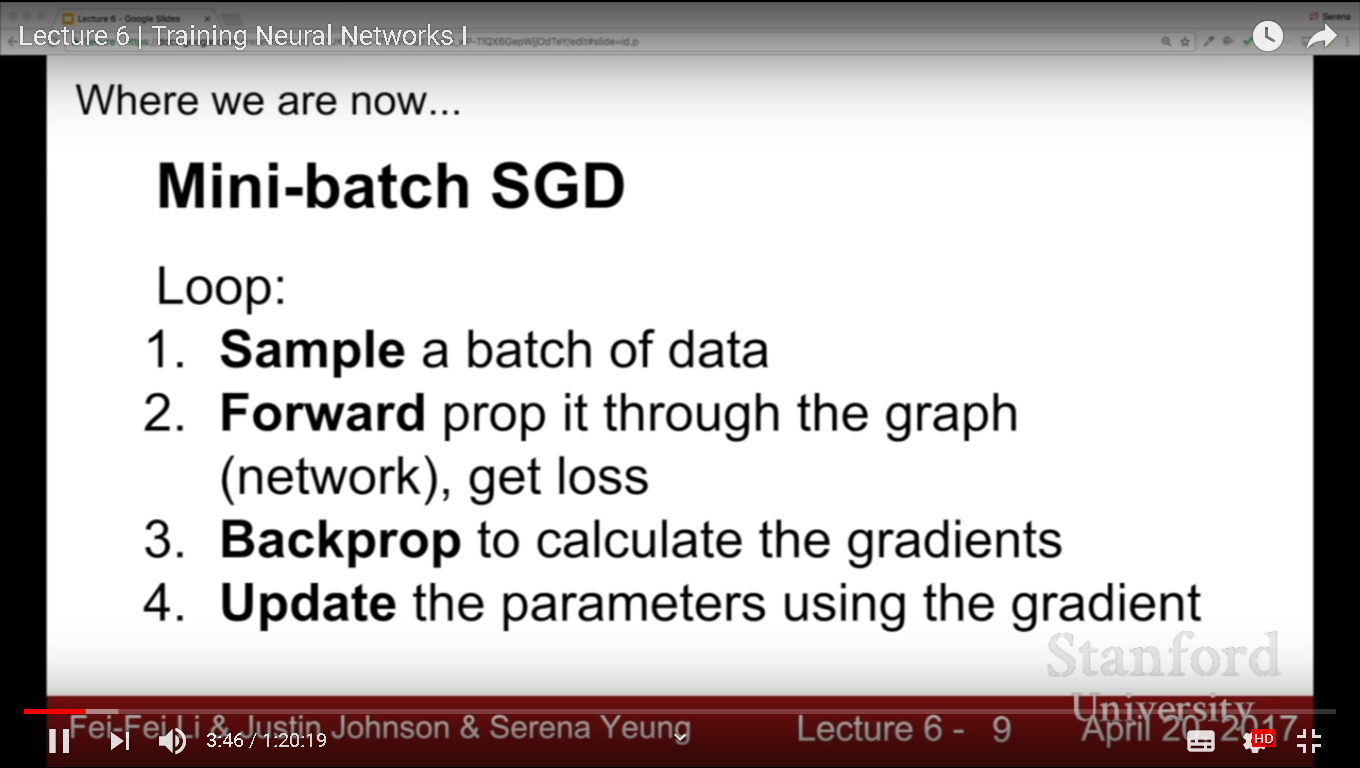
mul—switcher

vector-gradient:

Jacobian matrices—diagonal matrix

forward pass/backward pass

Neural networks



How to train a convnet

Activation Functions

sigmoid/tanh/ReLU/Leaky ReLU/PreLU/ELU

Data preprocessing

Initialization

Batch Normalization(Really tough one)