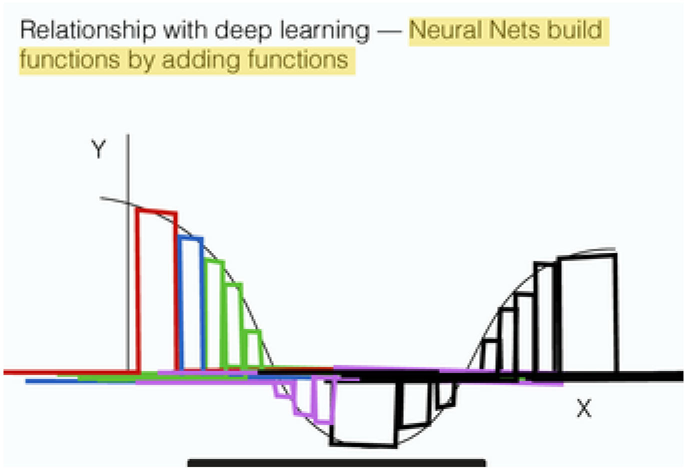
Understanding of Neural Network and CNN

7/3/2022 NN-DL

What is the node? How each node works?



Each node is a function

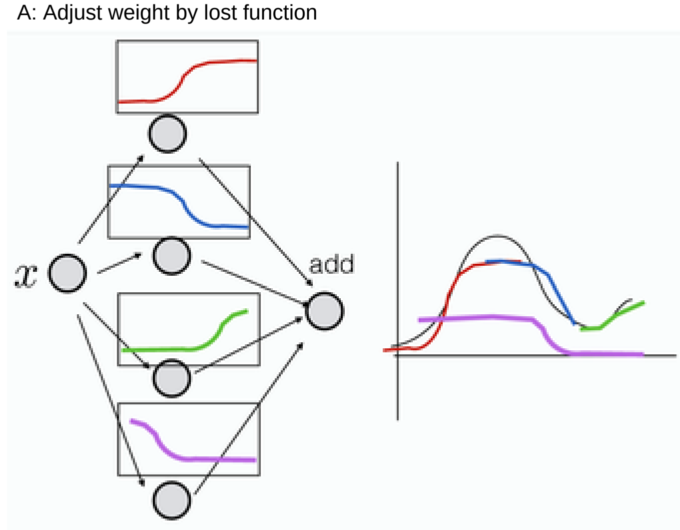
each input has its own pattern. if only classify x1, left are orange right are blue. if only classcify x2, buttom are orange and top are blue. I**nput themselves are functions** of the sample with the boundary **of that specific dimension.** Without first layer, the NN will adjust input function and get a **combined boundary**. But in this case only linear.

it is adding (activated) functions that give us nonlinearity.

Feel like h (as a function) is the linear combination of inputs

How does the network know how to fold data space?

A: Adjust weight by lost function



Here guess from v to m to o is the way of regulating output(scale).

The loss function is a surface

In theory, we could calculate the cost of all combinations of weight. But that is not efficient.

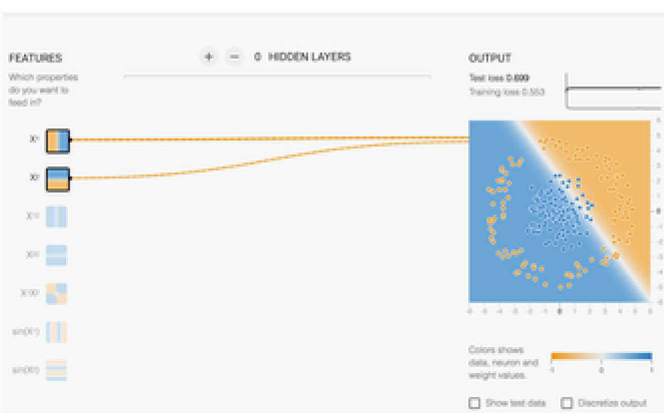
How to guide the network to find the lowest cost?

Here we need back propagation & gradient descent.

From these two slides, seems (positive or negative) gradient provides the direction of x1, And the value of the gradient provides how large one step is (how much should weight change). **Guess X is the weight**

Q: Why gradient could guide weight change in this way? How to calculate it?

1. Forward propagation output(O = v3)



2. Calculate cost C(y, o)
3. partial derivative (cuz we only care about weight) cost with weight (image cost is the function of weight as the figure above) -->gradient.
4. Let weight update follow the calculated gradient
5. Untill gradient = 0 (unchange of lost max/min). Converge Note here, gradient = 0 not means cost = 0, in the end, when converge , the gradient is zero but the cost can be any value.(even may not be the minimum (in global.))
6. As long as it is not 0, sill the loss can be reduced.

Regulation

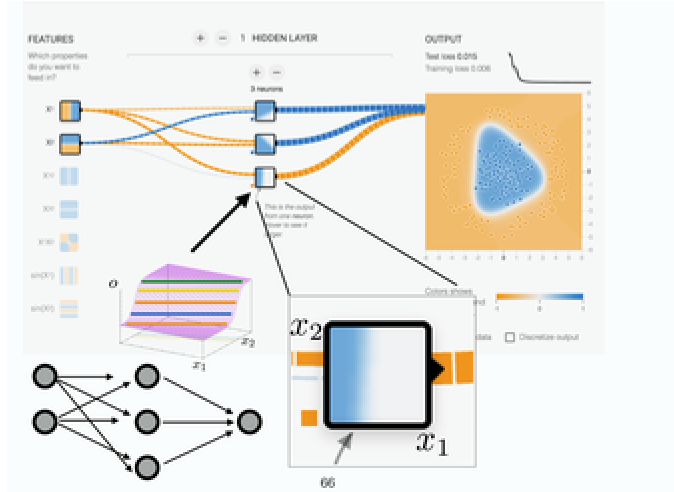
what is the complexity of neural network?

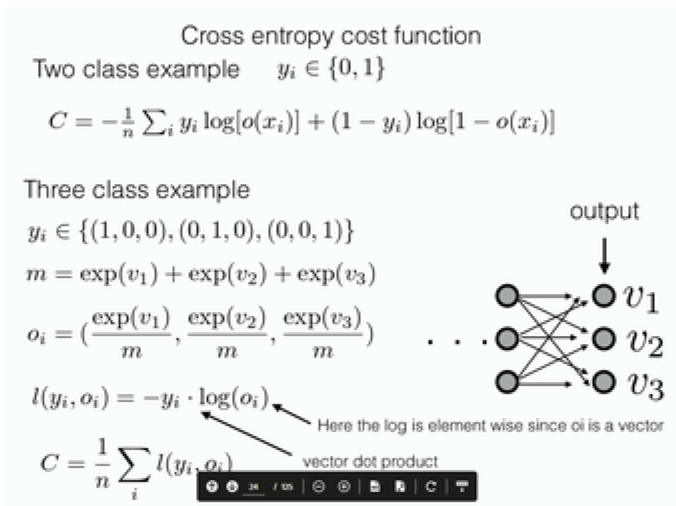
A: The complexity of the neural network (partly) is the number of weights (neural) in the nerwork.

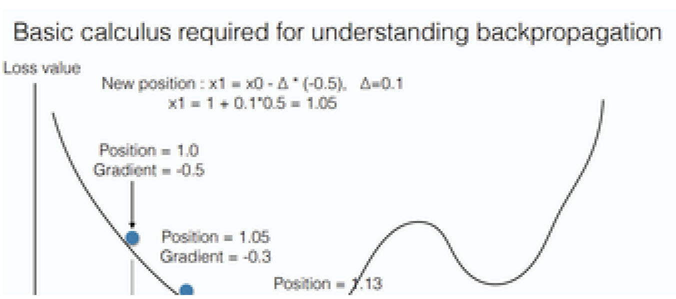
Seems reducing complexity of the neural network could prevent overfitting or at least one pf the way

How is this complexity related to the weight (L1, L2 regulation)?

if the weight is equal to zero (or close to), it cld reduce the number of neural of thus complexity of the network.





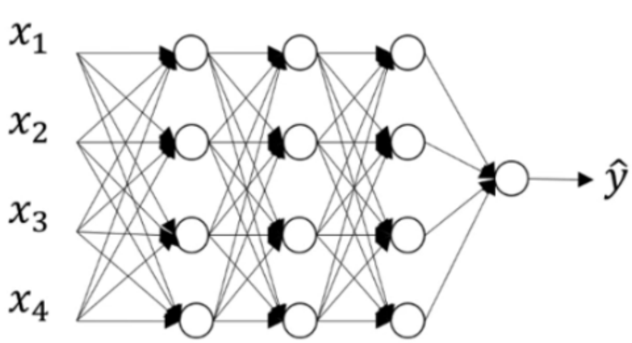


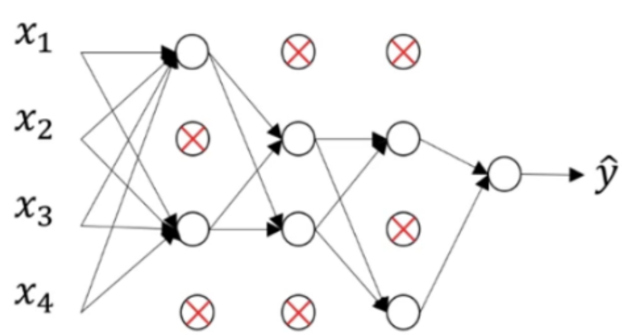
This is why the idea of regulation is to punish large weight (bias small weight)

.

The red ellipses is the contour of the cost(same cost value alone the contour), and all the points(with diff w1,w2 cld achieve the same cost). Now either L1 and L2 wanna find the pair closest to the center.

Dropout:





probability of **P=0.5** that a random neuron gets turned off during training would result in a neural network on the right side.

What is the local minimum problem?

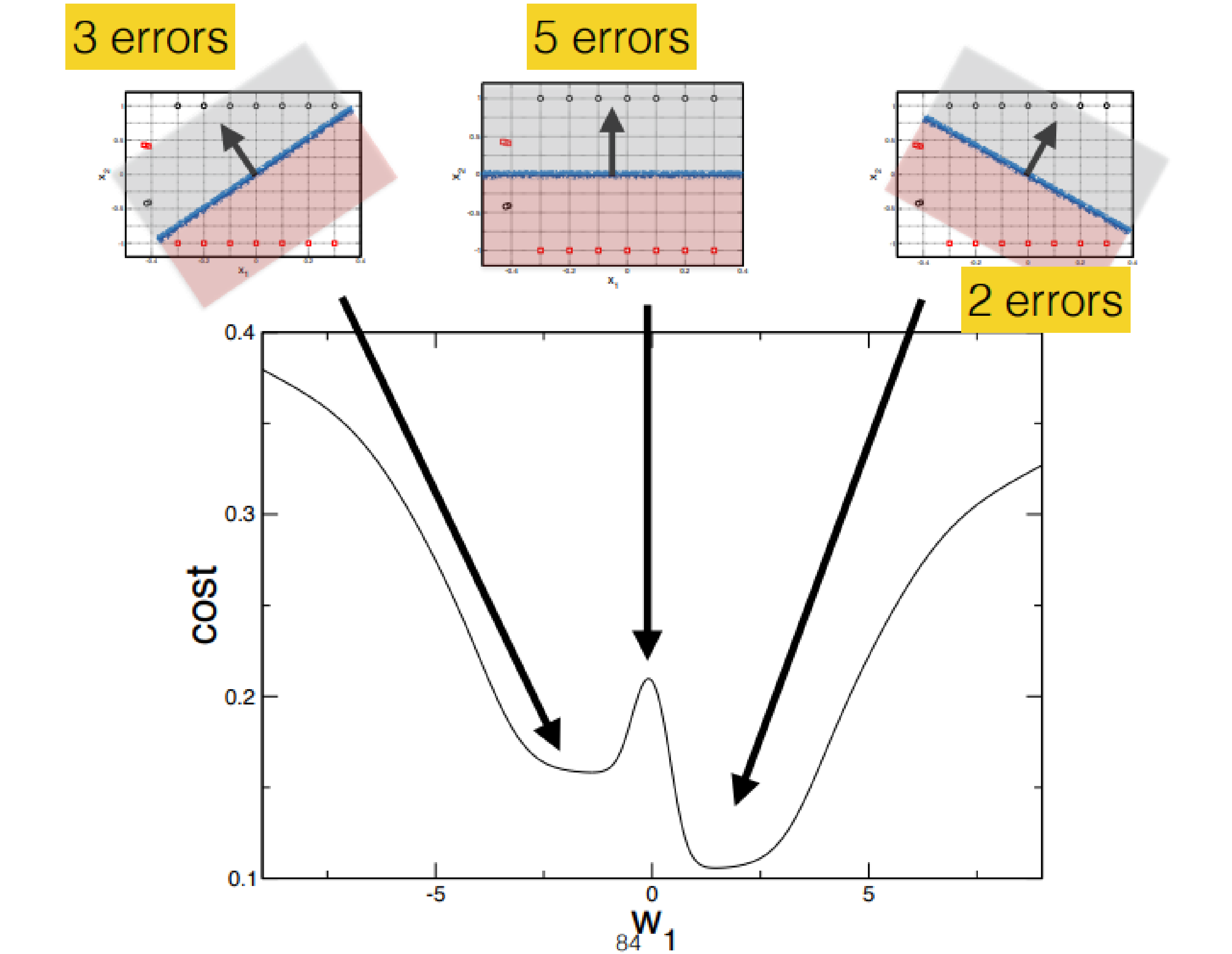
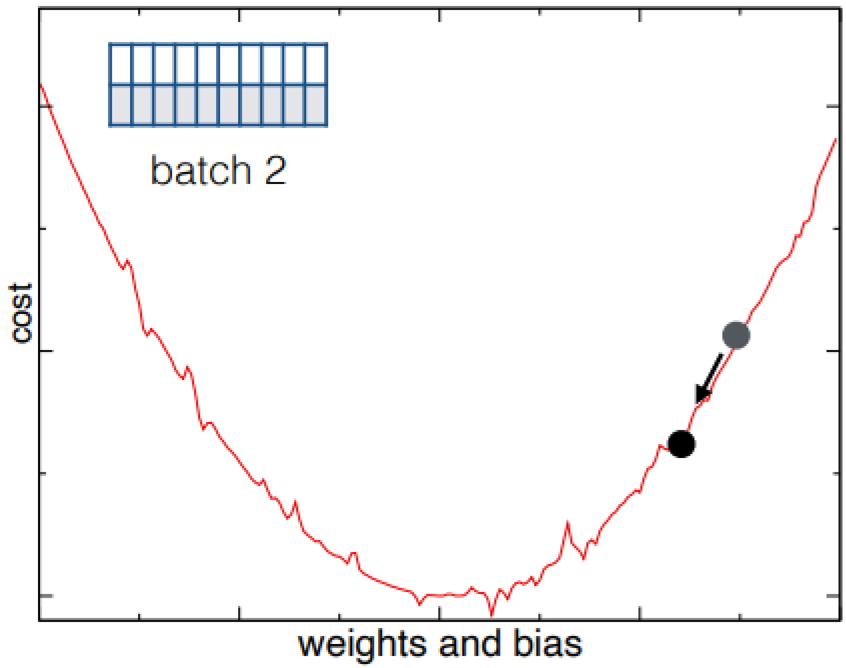


Figure 1: An unregularized model can overfit to noise/outliers in the training data

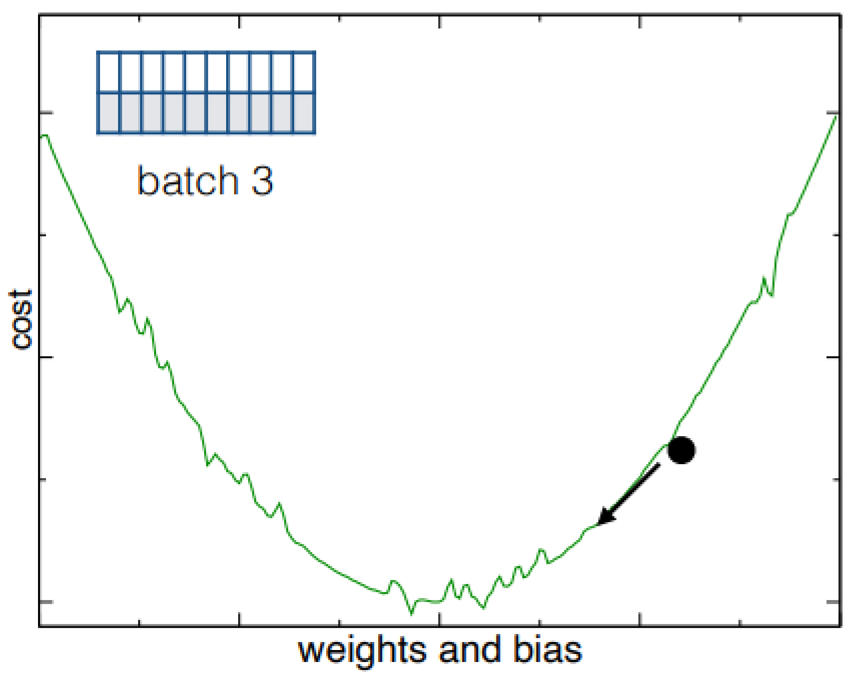
How to overcome it?

Stochastic Gradient Descent (SGD) – Randomly break dataset into small batches.

“Stochastic”, in plain terms means “random”.



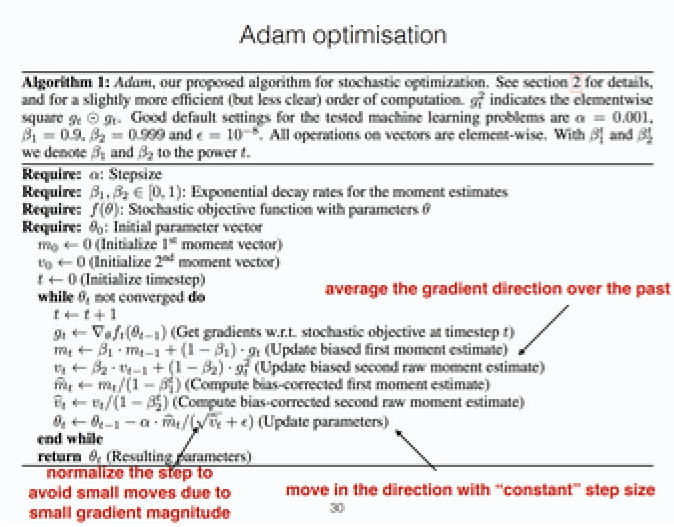
105



steps: updating two parameters mt and vt, which contain previous gradient and gradient square. Scale them obtain m and v hat. The results grasient is the combination of m and v hat.

Idea: take previous gradience as momentum.

Vanishing gradient problem


Adam optimisation

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

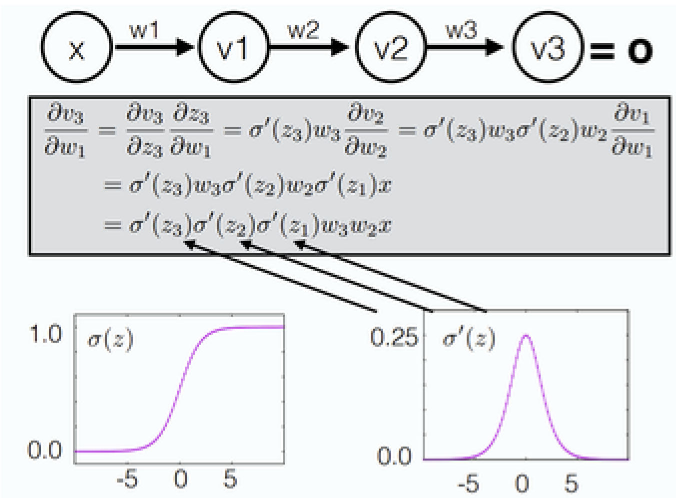
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1[st] moment vector)
  $v_0 \leftarrow 0$ (Initialize 2[nd] moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t/(\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

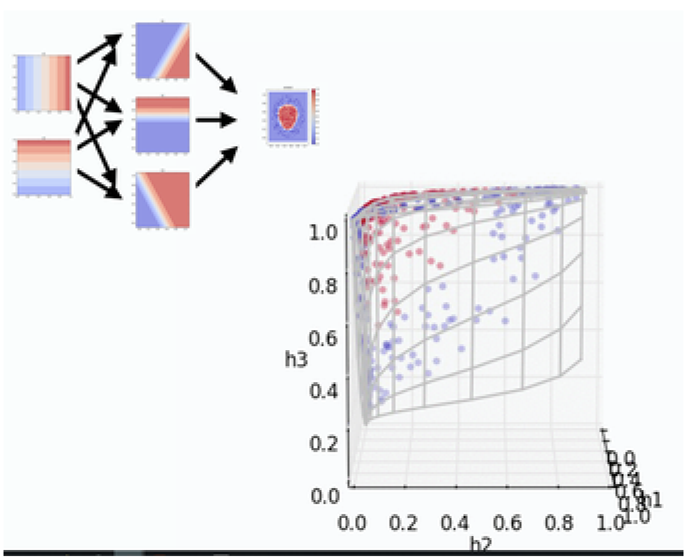
average the gradient direction over the past

normalize the step to avoid small moves due to small gradient magnitude

move in the direction with "constant" step size

30





When doing the back propagation, the result after active function always in (0,1), then when do the deviation, the deciation might be super small.

Then the gradient might equal to zero.

To overcome it, using short-cuts (residual net)

How to understand manifold?

A: Can interpret it as the rules that contain datas into a space in a certain way. This rules is learned by nerual networks.

What are encoding and decoding?

Feel like encoding it the operation that NN map data in a way(certain rules with weights) resulting in latent space that easiler to perforam the job(eg. classification). Decoding is from latent space to high dimension space.

What is dropout and why?

Drop out is the operation in the NN. It could help to simplify NN structure and thus prevent overfitting.

How does dropout works?

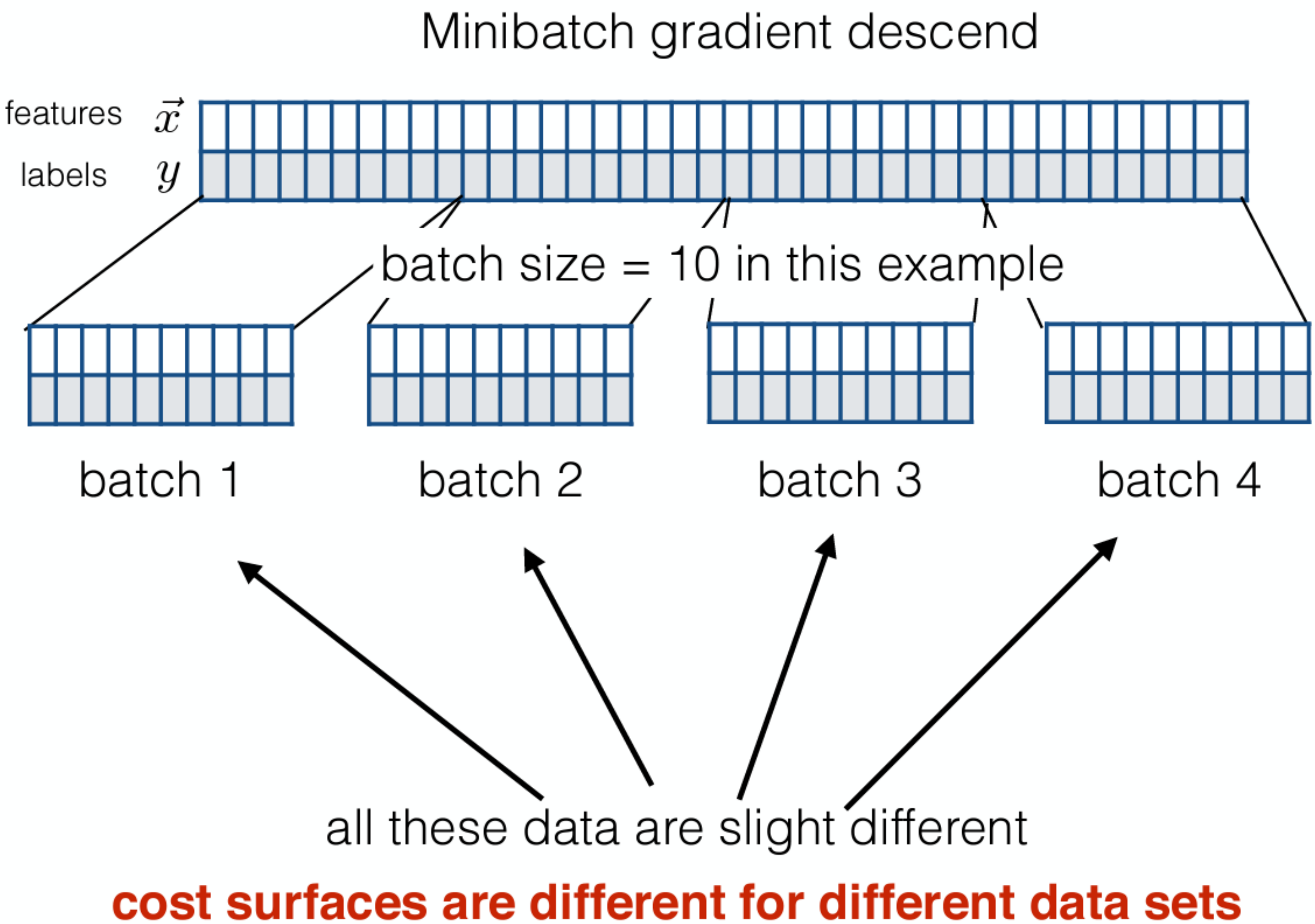
Can imagine drop out to simple the NN structure.

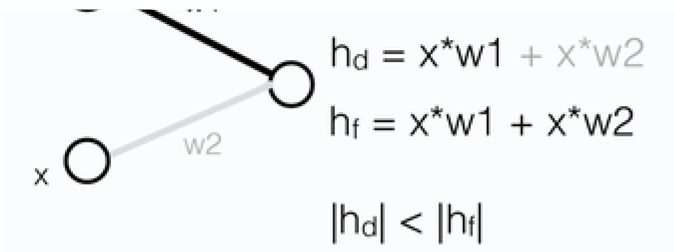
Specificly, it mainly works in the back propagation, Partial deviation of a certain node is the sum of the partial deviation of entries of that node.

Drop out means reducing the entries in the sum.

What is batch normalization?

Normalization is to remove the variance between data/samples in one batch (batch is the subset of the whole datasets) .





So normalization can be performed between layers, which can be set by ppl. Without normalization, the large become larger, and the small becomes smaller

What is depthwise separable convolution?

Task: Given 12*12*3 raw image to achieve 125 convolued

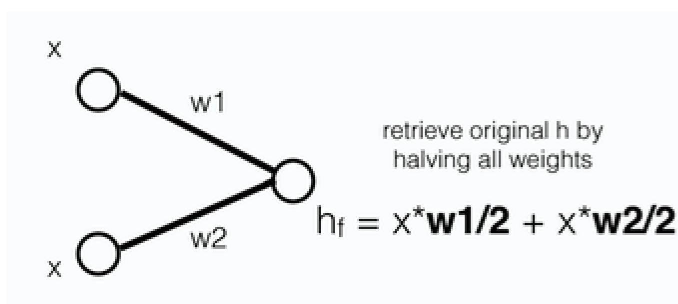
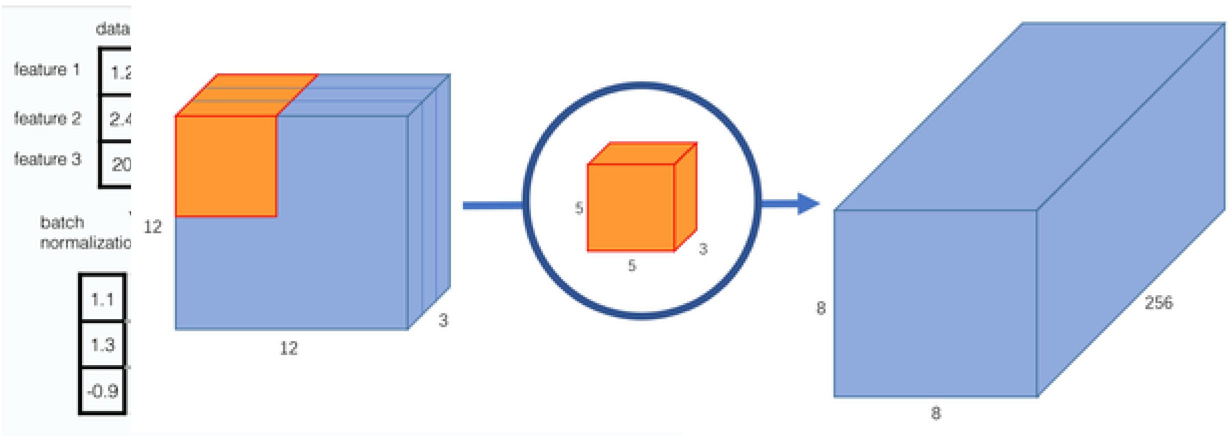


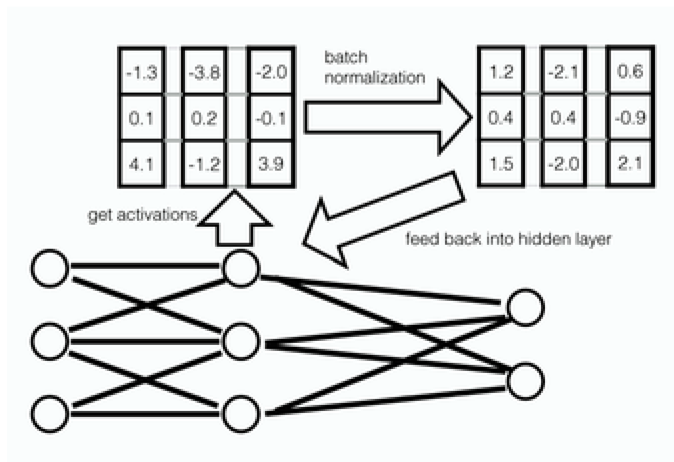
image stack

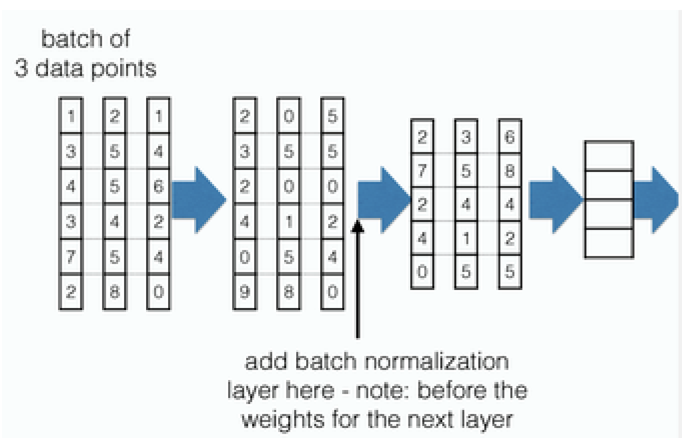
Normal:

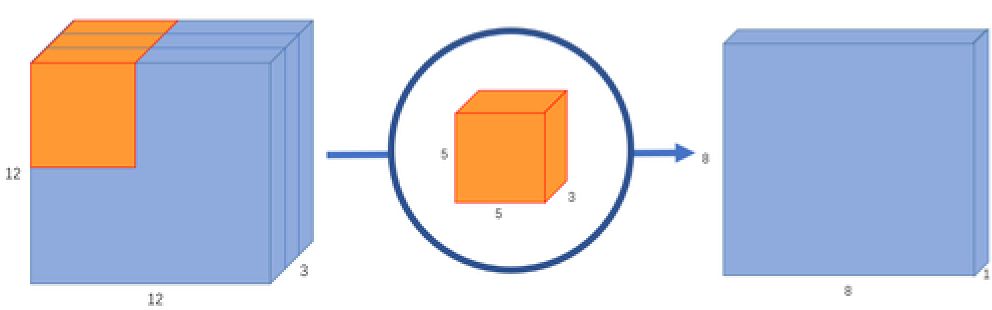
weight in kernel = 5*5*3 = 75 One kernel 75 multiplication

Moving 8*8 time to do convolution: 5*5*3*8*8 = 4800 multi



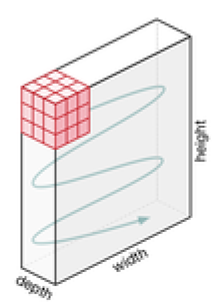






125 same size but diff weight kernel 125*8*8*1 image:

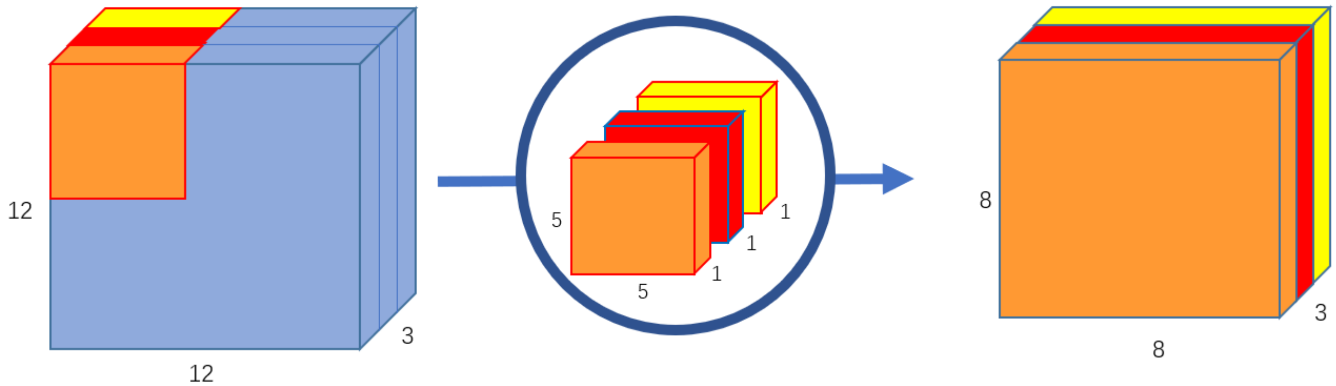
125*5*5*3*8*8 = 1,228,800 multi



depthwise separable

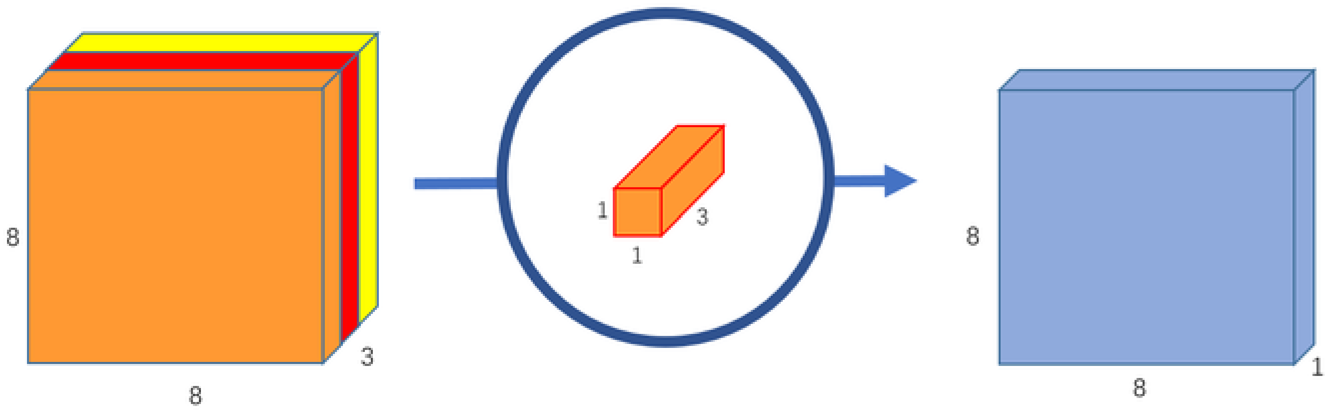
convolution:

part 1: Depthwise Convolution:



Notice the size of kernel is the same (3*5*5*1) , just do not sum them together: 5*5*1*8*8*3 = 4800 multi no sum

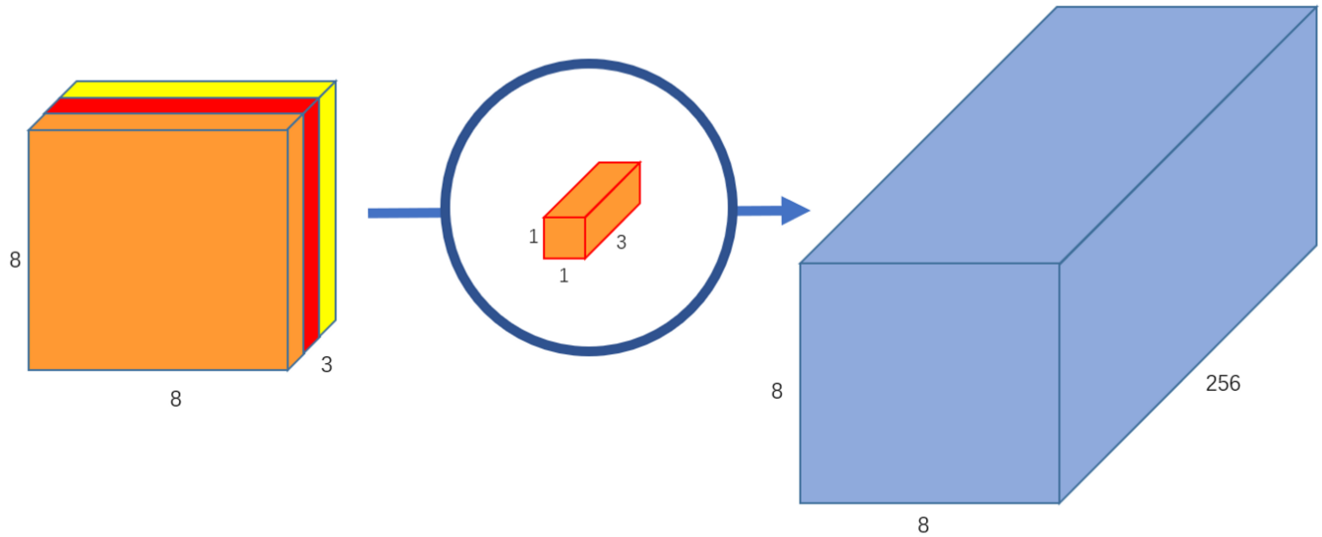
Part 2 — Pointwise Convolution:



This step is to generate fidd combination (scale and sum) of the previous resulted 3*8*8*1 images. And thus why we couldn't sum them together for the seek of having space to combine them .

But what if the space is different?

The bottom line what why normal is equal to this two steps convolution is that: if the column space of (5*5*3) kernel is the same as 3*5*5*1 kernel, then any 5*5*3 kerne in normal wayl could be generated through combination (from pointwise conv).



This step: 1*1*3*8*8*125 = 49,152

So in total: two steps conv: 4800+49,152 = 53,952 muti

pointwise nonlinearity

means apply nonlinear function(active function) to individual component without mixing entries.