Outline

1. Hyperparameter Tuning
2. Batch Normalization
3. Multi-class classification
4. TensorFlow

Hyperparameter Tuning

Importance

1. Learning Rate α
2. Momentum β, minibatch size, number of hidden units
3. number of layers learning rate decay
4. others, liike the parameters in Adam

Scale of Learning Rate

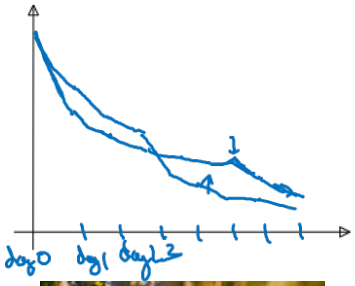
Learning rate should be sampled on different scales
For example, $\alpha \in [0.001, 0.01, 0.1, 1]$

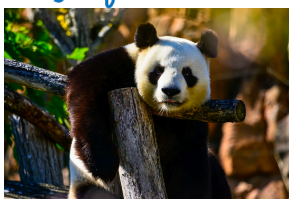
Intuitive example: learn w_0 in the problem $y = w_0 x$.
Loss function $l(w) = (y - wx)^2$
Derivative $l'(w) = 2x(wx - y)$
Update by $w^{(t+1)} = w^{(t)} - 2\alpha x(w^{(t)}x - y)$

Assume $w_0 = 2$ and we have train model twice on two different datasets
Dataset1: $(x, y) = \{(1, 2); (2, 4); \}$
Dataset2: $(x, y) = \{(10, 20); (20, 40); \}$
The learning rate in dataset 2 should be 100 times smaller than that in dataset 1!

Tuning Can be Slow

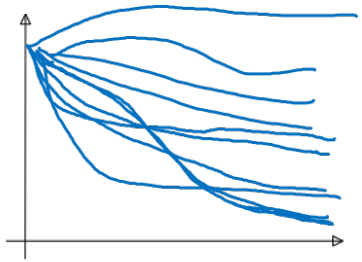
Babysitting one model





Panda

Training many models in parallel



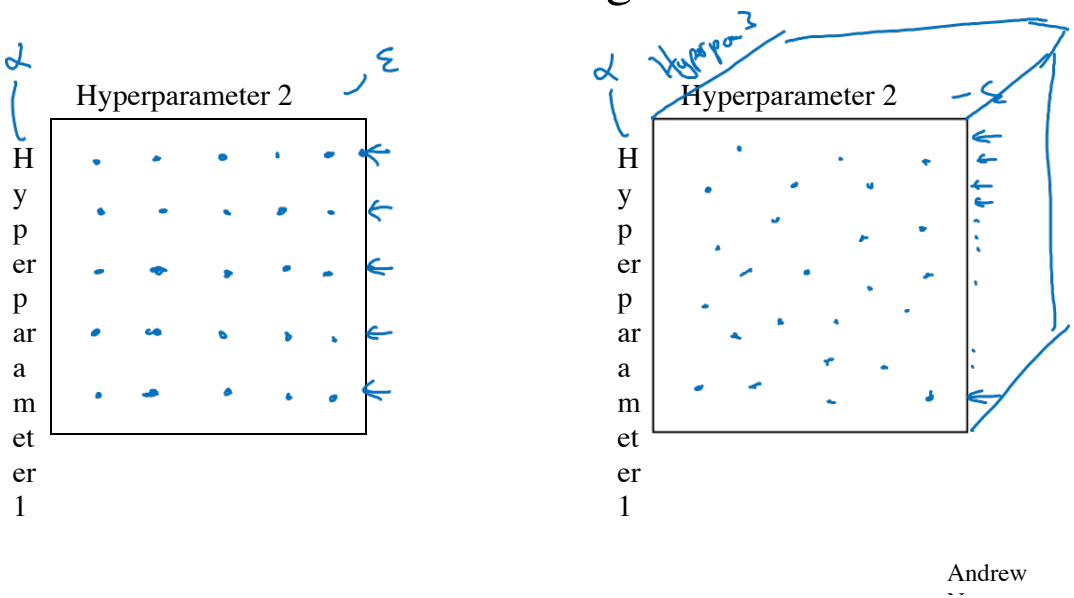
Caviar

Andrew

Other Parameters

Sample momentum $\beta \in [0.9, 0.99, 0.999]$ Computing the "moving average" of $1/(1-\beta) = [10, 100, 1000]$ minibatches Sample randomly

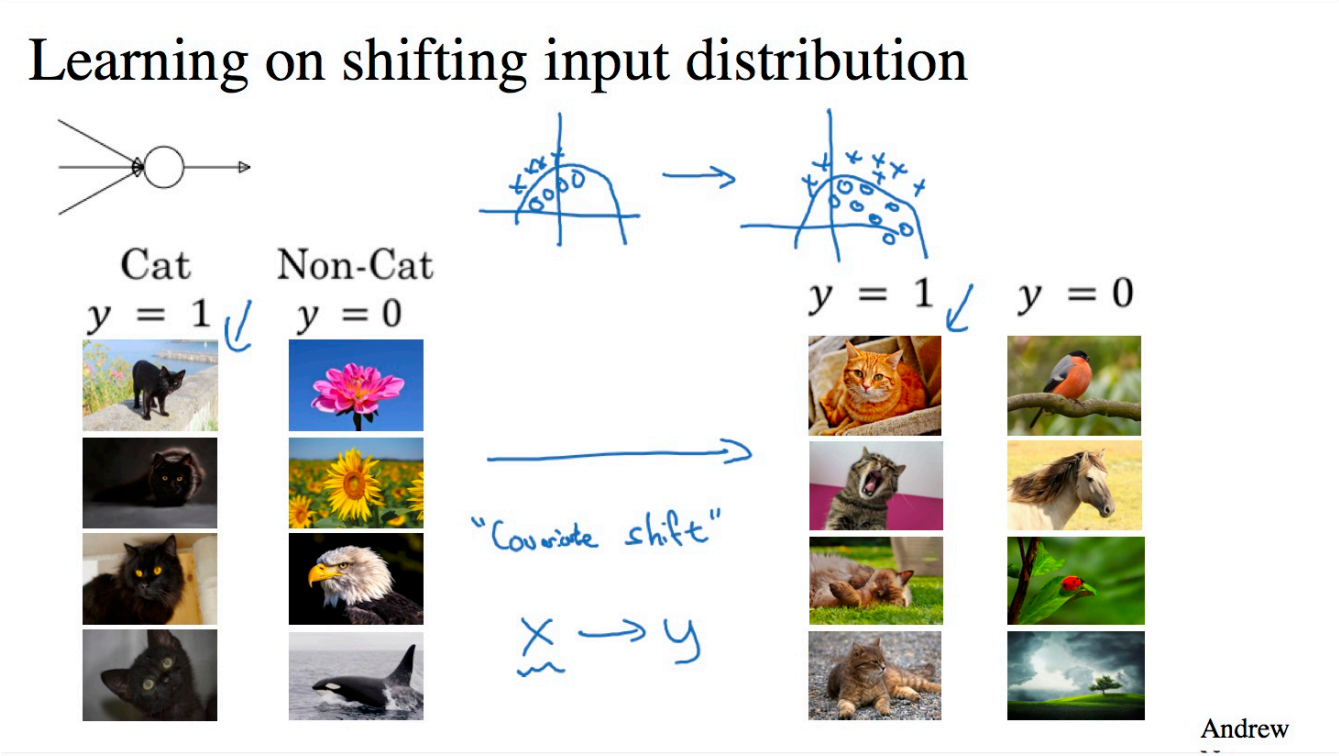
Try random values: Don't use a grid



Andrew

Batch Normalization

Problem: Minibatch has small size and not i.i.d. between them
Different colors of cats in different mini-batches



ALL neuron activities will be affected and need to be adjusted
Not only input ones

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Subtract mean and variance information from minibatch
If any of these are necessary, then learn it through the network.
Ideally, in the cat example, $\hat{\beta} = 0$ if we don't want to distinguish cats with different colors, and $\hat{\beta}! = 0$ otherwise.

Graph

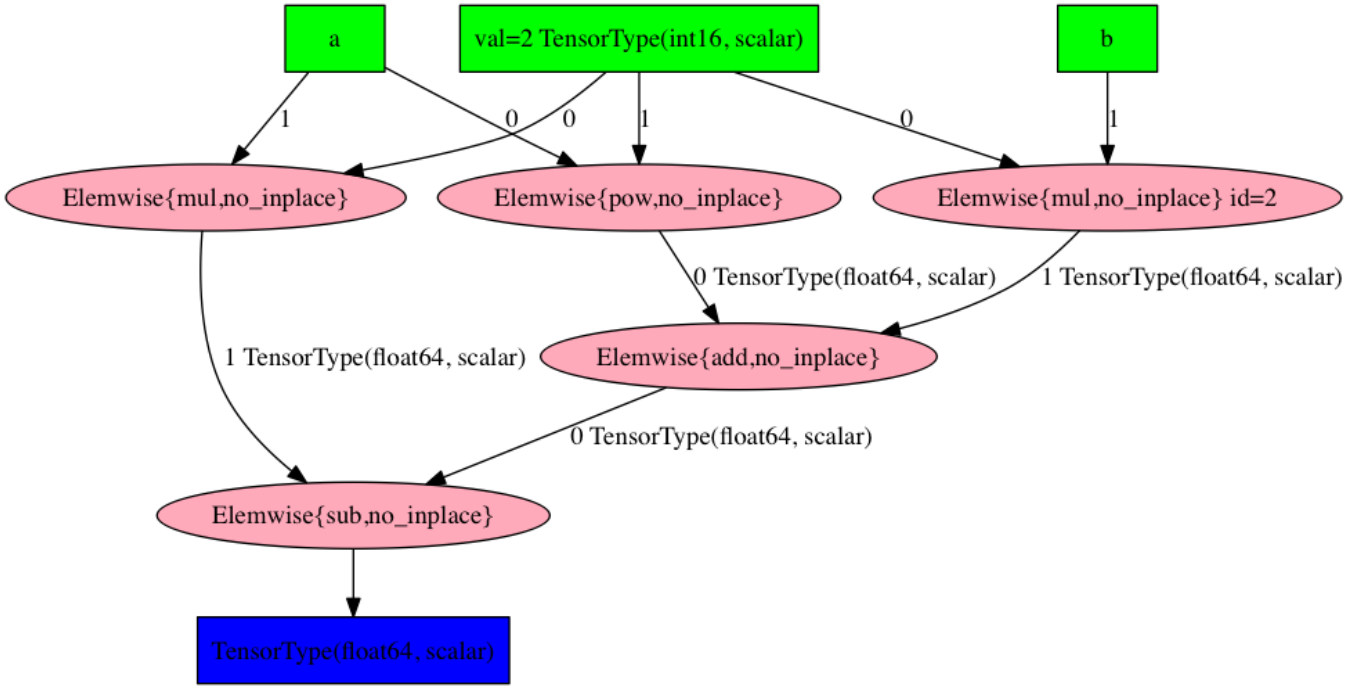
Value calculations

$$c = a^2 + 2b - 2a$$

$temp1 = a^2$
$temp2 = 2 * b$
$temp3 = temp1 + temp2$
$temp4 = 2 * a$
$c = temp3 - temp4$
No connection. How to compute gradient?

Graph

$$c = a^2 + 2b - 2a$$

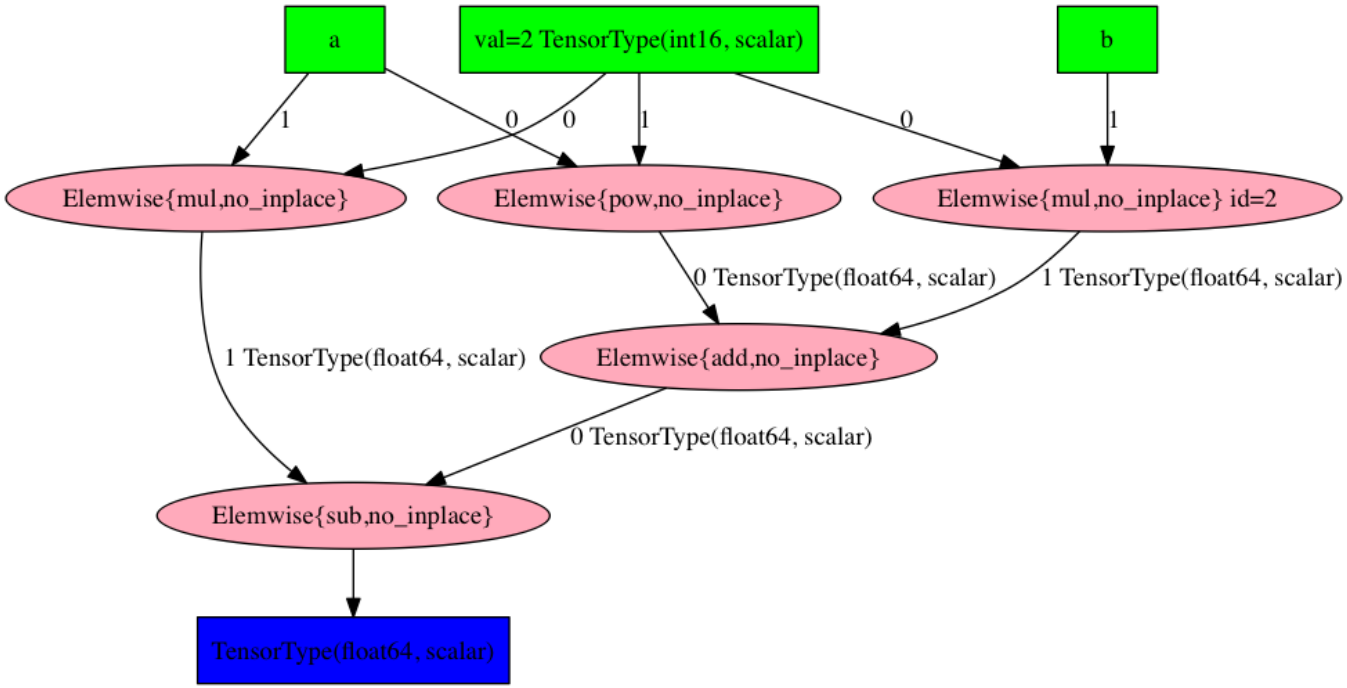


Session, Operation and Tensor

Operation: a graph node that performs operations on tensors
Tensors: represents one of the outputs of operations
Session: where operations are executed and tensors are evaluated

Graph

$$c = a^2 + 2b - 2a$$



c is connected to a and b!

Handle

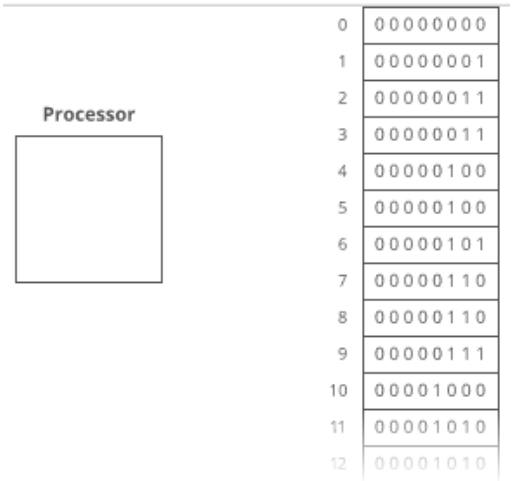
Imagine you want to read a book in a library. You have two choices:
1) obtain the **content** of the book.
2) obtain the **address** of the book.
Handle: use method 2.
+ Gradient calculation is automatic
+ It consumes no time to obtain an address in memory
+ Copying a large object is slow
+ Higher level of abstraction. Same code can run on CPU and GPU
- Unexpected changes in its value.
Ex: $a = 3; b = a; a = 4; b =?$

Tensor Types

Constant: constant tensors. Widely used implicitly by TensorFlow.
e.g., $b = 2a$, 2 will be converted to a constant tensor.
Variable: a variable
1) has a fixed shape;
2) need to be initialized before calling operations on it.
3) is automatically collected in the graph.
examples of variables?
Placeholder: used as a handle to feed data.
*(*Go to exercise part1)

Handle Revisited: Datatype

Knowing address is not enough.
"int32" and 'float32" are datatypes that specifies
1) how long the data is
2) how to decode the "random" 0-1 strings.



No such problems when we assign values $a = b$!

Tensor types revisited

Constant:

1. usage: implicit on constant numbers mostly
2. datatype: implicit
3. shape: implicit

Variable:

1. usage: parameters and cost function
2. datatype: user-specified
3. shape: user-specified

Placeholder

1. usage: data input
2. datatype: user-specified
3. shape: given by real input data