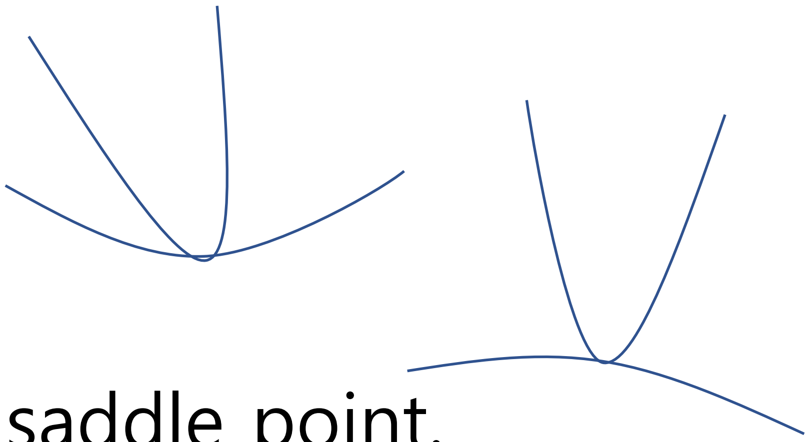
The problem of local optima



- Almost local optima in high dimension is a saddle point.
- But such bad optima can escape if the network is big enough.

- The problem of plateaus
- Derivative is almost zero.
- Learning speed decreases.
- Adam, RMSProp -> Change the direction

Tensorflow

- Example
- $J(w) = w^2 - 10w + 25$
- Finding minimum w = 5 from tensorflow
- Similar technic can be applied to real problem.

Tensorflow example

- w = tf.Variable(0,dtype=tf.float32)
- cost = tf.add(tf.add(w**2,tf.multiple(-10,w)),25)
- train = tf.train.GradientDescentOptimizer(0.01).minimize(Cost)

- init = tf.global_variables_initializer()
- session = tf.Session()
- session.run(init)
- session.run(w)

Tensor flow example

- For i in range(1000):

```
        session.run(train)
print(session.run(w))
```

Result : 4.99999

Tensorflow

- Define only forward propagation then tensorflow will calculate back-prop automatically. (In add, multiply, square)
- Only define cost function.
- We can use operater overloading (w**2-10*w+25)

Tensorflow

- coefficient = np.array([[1.][-10.][25.]])
- x = placeholder(tf.float(32),[3,1]) -> Give data later
- cost = x[0][0]*w**2-x[1][0]*w+w[2][0]
- session.run(train, feed_dict=(x:coefficient))

- Such a way, we can feed the training data.

Why ML Strategy?

- more data
- more diverse training set
- Train longer
- Adam (better algorithms)
- bigger network
- smaller network
- dropout
- L2 regularization
- Network architecture

- What is efficient ????

Orthogonalization

- One button -> Only one option
- One dimension change -> No change in other dimensions

- Chain of assumptions in ML
  - Tune to fit training set
  -> Bigger network, Adam(optimization algorithms)
  - Tune to fit development set
  -> Regularization, bigger training set
  - Tune to fit test set
  -> Bigger development set
  - Real world
  -> Change development set or cost function

  Early training mix the orthogonalized dimensions.