Parameter updates

Vanilla update

```
# Vanilla update
x += - learning_rate * dx
```

The simplest form of update is to change the parameters along the negative gradient direction
since the gradient indicates the direction of increase, but we usually wish to minimize a loss function

---

Momentum update

```
# Momentum update
v = mu * v - learning_rate * dx # integrate velocity
x += v # integrate position
```

Annealing the learning rate

随着时间的推移，适当减小learning rate

Adagrad is an adaptive learning rate method originally：

```
# Assume the gradient dx and parameter vector x
cache += dx**2
x += - learning_rate * dx / (np.sqrt(cache) + eps)
```

RMSprop：

```
cache = decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + eps)
```

Adam is a recently proposed update that looks a bit like RMSProp with momentum. The (simplified) update looks as follows:

```
m = beta1*m + (1-beta1)*dx
v = beta2*v + (1-beta2)*(dx**2)
x += - learning_rate * m / (np.sqrt(v) + eps)
```

Hyperparameter optimization

The most common hyperparameters in context of Neural Networks include:

- the initial learning rate
- learning rate decay schedule (such as the decay constant)
- regularization strength (L2 penalty, dropout strength)