d2l_tools(WhiteCat)

- use_svg_display() --- 使用svg格式在jupyter中显示绘图
- set_figsize(figsize=(3.5,2.5)) ---- 设置图表大小 （figsize 默认为 （3.5，2.5））
- set_axes(axes, xlabel, ylabel, xlim, ylim, xscale, yscale, legend) --- 生成图标的轴属性
- ***plot(X,Y=None, xlabel=None, ylabel=None, legend=None, xlim=None, ylim=None,***

  xscale='linear', yscale='linear',

  fmts=('-','m--','g-.','r:'),figsize=(3.5,2.5),axes=None)

  绘图（一个自变量，同时绘制多个因变量的图）
- ***Timer***类 （可记录多次运行时间（列表））

  start --- 启动计时器；

  stop --- 停止计时并返回最近一次的运行时长；

  avg --- 返回全部运行的平均时间；

  sum --- 返回全部运行的总时长

  cumsum --- 返回累计时间
- synthetic_data(w,b,num_examples)

  生成具有噪声*(x~(0,1), y*增加的噪声*e~(0,0.01))*的线性数据集
- linreg(x,w,b) --- 线性回归模型
- ***squared_loss(y_hat,y)*** --- 均方误差（未求Sum）
- ***sgd(params, lr, batch_size)*** --- 小批量随机梯度下降 （反向传播修改权重）
- Huber(y_hat,y,delta) --- Huber_loss
- ***load_array(data_arrays, batch_size, is_train=True)***

  生成Pytorch数据迭代器
- get_fashion_mnist_labels(labels) --- 将数字索引与文本名称进行转换(fashion_mnist)
- show_images(imgs, num_rows, num_cols, titles=None, scale=1.5) --- 可视化样本
- get_dataloader_workers() --- 使用4个线程读取数据
- ***load_data_fashion_mnist(batch_size, resize=None)*** --- 获取和读取Fashion-MNIST数据集
- softmax(X) --- Softmax函数
- cross_entropy(y_hat, y) --- 交叉熵函数（未求和）
- accuracy(y_hat, y) --- 一个batchsize上的准确的总数
- ***evaluate_accuracy(net, data_iter)*** --- 计算指定数据集上的模型精度
- ***Accumulator***类 --- 在多个变量上累加 (需要传入变量的个数)

  使用方法:

  metrics = Accumulator(number) number --- 为你的变量个数
  metrics.add(.....)
  metrics[index] --- 获取该索引下的值
- train_epoch_ch3(net, train_iter, loss, updater) --- 训练
- ***Animator***类 --- 单图层动画绘制数据

(xlabel=None, ylabel=None, legend=None, xlim=None,
ylim=None, xscale='linear', yscale='linear',
fmts=('-', 'm--', 'g-.', 'r:'), nrows=1, ncols=1,
figsize=(3.5,2.5)

使用方法：

```
animator = Animator(xlabels='x',ylabel='sin(x)')

for x in range(100):

  # 增加图表中增加多个数据点

  animator.add(x, np.sin(x))
```

- ***Multi_Animator***类 --- 多图层动画绘制数据

(self, xlabels='x', ylabels='y', legends=None, xlims=None,
ylims=None, xscale='linear', yscale='linear',
fmts=('-', 'm--', 'g-.', 'r:'), nrows=1, ncols=1,
figsize=(3.5,2.5))

使用案例:

```
multi_animator = Multi_Animator(xlabels=['x','x'],ylabels=['sin(x)','cos(x)'],nrows=2, ncols=1)
for x in range(100):
  multi_animator(x,[np.sin(x),np.cos(x)])
```

- train_ch3(net, train_iter, test_iter, loss, num_epochs, updater) --- 可视化训练进度
- predict_ch3(net, test_iter, n=6) --- # 预测并展示部分结果