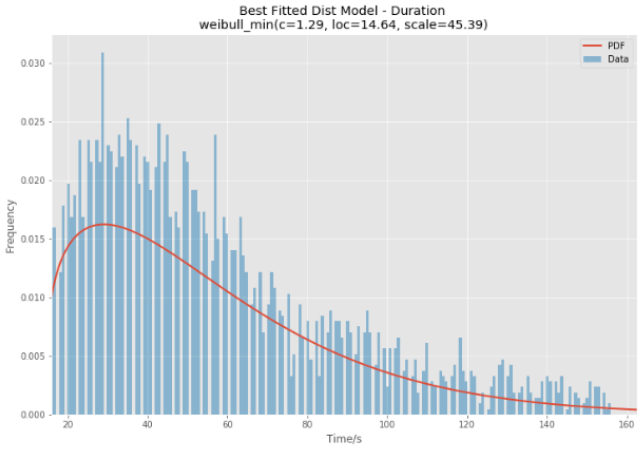
机器学习工程基础

弘飞

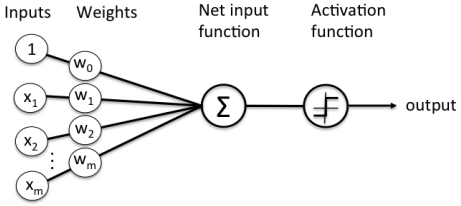
This page intentionally left blank



在机器学习的工程实践中，有三个要素非常重要，即策略构建、模型尝试、代码落地。这本小书用三个例子分别阐释。

用 Numpy 构建 感知机

最简单的感知机模型包含两个相对独立的计算过程：矩阵点积和激活函数。如图所示：



Schematic of Rosenblatt's perceptron.

现在就来通过 Numpy 来构建这样一个基本的感知机模型吧！首先我们需要一组数据。

```
import numpy as np

N, D_in, D_out = 640, 100, 1

x = np.random.randn(N, D_in)
y = np.random.randn(N, D_out)
```

这样就有了 640 条数据，x 中包含 100 个 Features 和一个 y 的标签共。为简单起见，所有数据都是正态随机生成。

感知机的学习问题可以概括为对参数 w 的优化，为此我们需要首先初始化参数 w。

```
w = np.random.randn(D_in, D_out)
```

整个优化过程可以分为四步：

1. 正向计算感知机的输出结果
2. 计算损失
3. 反向计算梯度
4. 根据梯度和学习率更新权重参数

我们一步一步来实现，首先是正向计算输出。这一步比较容易，只要定义好激活函数就不难完成。这里使用 Relu 函数：

```
h = x.dot(w)
y_pred = np.maximum(h, 0)
```

需要注意的是，`np.maximum` 和 `np.max` 不相同，前者是比较 h 和 0 的较大值，后者则是求某个序列的最大值。

这样就得到了输出值 y_pred 。接下来计算损失，我们使用最简单的均方误差来搞定：

```
loss = np.square(y_pred - y).sum()
```

均方误差对于误差的导数是两倍的误差值，反向计算第一层梯度时可以利用这一点：

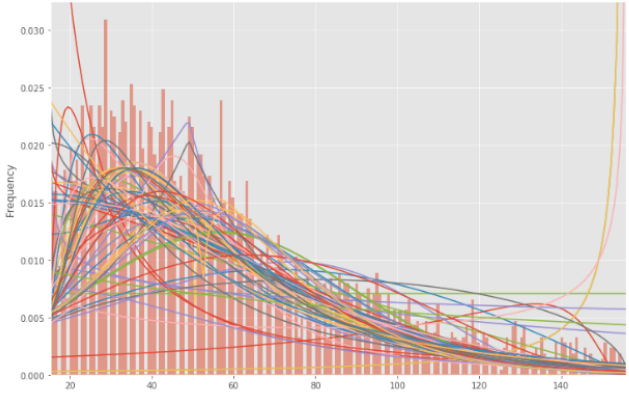
```
grad_y_pred = 2.0 * (y_pred - y)
grad_y_pred[grad_y_pred <= 0] = 0
grad_y_pred[grad_y_pred > 0] = 1
grad_w = x.T.dot(grad_y_pred)
```

最后只需将得到的 w 的梯度更新到 w 上即可：

```
learning_rate = 1e-3
```



```
def make_pdf(dist, params, size=10000):
    """Generate distributions's Propbability Distribution Function """

    # Separate parts of parameters
    arg = params[:-2]
    loc = params[-2]
    scale = params[-1]

    # Get sane start and end points of distribution
    start = dist.ppf(0.01, *arg, loc=loc, scale=scale) if arg else dist.ppf
    end = dist.ppf(0.99, *arg, loc=loc, scale=scale) if arg else dist.ppf(0

    # Build PDF and turn into pandas Series
    x = np.linspace(start, end, size)
    y = dist.pdf(x, loc=loc, scale=scale, *arg)
    pdf = pd.Series(y, x)

    return pdf

# Make PDF
pdf = make_pdf(best_distribution, best_params)

# Display
plt.figure(figsize=(12,8))
ax = pdf.plot(lw=2, label='PDF', legend=True)
f_duration.plot(kind='hist', bins=200, normed=True, alpha=0.5, label='Data'


best_dist = getattr(st, best_distribution.name)
param_names = (best_dist.shapes + ', loc, scale').split(', ') if best_dist.
param_str = ', '.join(['{}={:0.2f}'.format(k,v) for k,v in zip(param_names,
dist_str = '{}({})'.format(best_distribution.name, param_str)

ax.set_title(u'Best Fitted Dist Model - Duration \n' + dist_str)
ax.set_xlabel(u'Time/s')
ax.set_ylabel('Frequency')
```



再来看看最终找到的分布模型及参数：


```
w -= learning_rate * grad_w
```

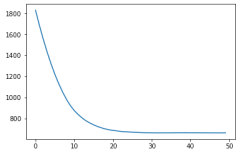
至此完成了一个完整的参数更新过程。我们把所有代码整合到一起，并且多来一些迭代，感受下效果：

```python
import numpy as np

N, D_in, D_out = 640, 10, 1

x = np.random.randn(N, D_in)
y = np.random.randn(N, D_out)

w = np.random.randn(D_in, D_out)
learning_rate = 1e-3
step = list()

for t in range(50):

    # Forward pass
    h = x.dot(w)
    y_pred = np.maximum(h, 0)

    # Compute and print loss
    loss = np.square(y_pred-y).sum()
    step.append(loss)
    print(loss)

    # Backward calc gradient
    grad_y_pred = 2.0 * (y_pred - y)
    grad_y_pred[grad_y_pred <= 0] = 0
    grad_y_pred[grad_y_pred > 0] = 1
    grad_w = x.T.dot(grad_y_pred)

    # Update parameters
    w -= learning_rate * grad_w
```

```
1830.171828902638
1686.4780197041364
1553.508774250499
1431.3911039105303
1320.0161410927292
1219.361190240088
1129.6530948528375
1050.220219445987
980.004837811834
921.2983375687231
874.136845616008
837.0796144184712
804.9862416641668
777.5387209932966
756.8852535832245
738.3937614727106
723.0074789872184
710.450887360988
699.4061790172179
691.4288848513102
685.8777529994169
```

```
%matplotlib inline
plt.plot(step)
```

```
[<matplotlib.lines.Line2D at 0x113243eb8>]
```



可以看到误差在不断地下降。

接下来请合上这本书，自己独立试试建立一个简单的感知机模型。前几次尝试可能会不确定或报错，请查阅上面的处理过程或 Google 一下，保证自己理解并且可以独立解决，直到能够不参考所有资料，独立写完并运行。

你还可以尝试改造和丰富这个感知机，如把损失函数换为 Sigmoid 或其他，或者在中间加一层隐藏层。Just try it!

多分布拟合

我们现在有很多用户的做题数据，记录了用户做题过程的时间、用时、正误等。在这个任务中，我们需要找到最能拟合用户做题用时分布的概率模型。

首先来准备数据。由于用户做题用时可能与题目本身有关，因此我们以一道题目的做题记录作为数据源。在 HUE 中，先找到一道题目出来：

My Snippet

```
SELECT problem_id, body
FROM dw.dim_problem
WHERE type = 'single_choice'
LIMIT 3
```

	problem_id	body
1	00ec42f6-57e9-11e7-bc9d-9bcf2b652f72	已知$5a-1$的平方根是 $ \pm 2$，$a+2b-1$的平方根是 $ \pm 4$，则$a+b$的平方根为（$\;\;$）
2	0195b52a-57e9-11e7-bcc0-93f8c6bba2771	若$ \sqrt {2.55}=1.597$，$ \sqrt {m}=15.97$，则$m=$_____.$$（本题中取值均为近似值）
3	02bd9b52-57e9-11e7-bcfb-7b5e87778de0	解方程$9x^2-144=0$得（$\;\;$）

我们就以第一道题为例，找到所有这道题的做题记录并根据用户分组去重，然后选择 10000 个用户：

```
# Plot for comparison
plt.figure(figsize=(12,8))
ax = f_duration.plot(kind='hist', bins=200, normed=True, alpha=0.5)
# Save plot limits
dataYLim = ax.get_ylim()


# Best holders
best_distribution = st.norm
best_params = (0.0, 1.0)
best_sse = np.inf

# Estimate distribution parameters from data
for distribution in DISTRIBUTIONS:
    # Try to fit the distribution
    try:
        # Ignore warnings from data that can't be fit
        with warnings.catch_warnings():
            warnings.filterwarnings('ignore')

            # fit dist to data
            params = distribution.fit(f_duration)

            # Separate parts of parameters
            arg = params[:-2]
            loc = params[-2]
            scale = params[-1]

            # Calculate fitted PDF and error with fit in distribution
            pdf = distribution.pdf(x, loc=loc, scale=scale, *arg)
            sse = np.sum(np.power(y - pdf, 2.0))

            # add to plot
            try:
                if ax:
                    pd.Series(pdf, x).plot(ax=ax)
                end
            except Exception:
                pass

            # identify if this distribution is better
            if best_sse > sse > 0:
                best_distribution = distribution
                best_params = params
                best_sse = sse

    except Exception:
        pass

ax.set_ylim(dataYLim)
```

：

最终结果如下图：

```
# Distributions to check
DISTRIBUTIONS = [
    st.alpha,st.anglit,st.arcsine,st.beta,st.betaprime,st.bradford,\
    st.burr,st.cauchy,st.chi,st.chi2,st.cosine, st.dgamma,\
    st.dweibull,st.erlang,st.expon,st.exponnorm,st.exponweib,\
    st.exponpow,st.f,st.fatiguelife,st.fisk,st.foldcauchy,st.foldnorm,\
    st.frechet_r,st.frechet_l,st.genlogistic,st.genpareto,\
    st.gennorm,st.genexpon,st.genextreme,st.gausshyper,st.gamma,\
    st.gengamma,st.genhalflogistic,st.gilbrat,st.gompertz,st.gumbel_r,\
    st.gumbel_l,st.halfcauchy,st.halflogistic,st.halfnorm,\
    st.halfgennorm,st.hypsecant,st.invgamma,st.invgauss,st.invweibull,\
    st.johnsonsb,st.johnsonsu,st.ksone,st.kstwobign,st.laplace,\
    st.levy,st.levy_l,st.levy_stable,st.logistic,st.loggamma,\
    st.loglaplace,st.lognorm,st.lomax,st.maxwell,st.mielke,\
    st.nakagami,st.ncx2,st.ncf,st.nct,st.norm,st.pareto,st.pearson3,\
    st.powerlaw,st.powerlognorm,st.powernorm,st.rdist,st.reciprocal,\
    st.rayleigh,st.rice,st.recipinvgauss,st.semicircular,st.t,\
    st.triang,st.truncexpon,st.truncnorm,st.tukeylambda,st.uniform,\
    st.vonmises,st.vonmises_line,st.wald,st.weibull_min,\
    st.weibull_max,st.wrapcauchy
]
```

接下来准备好上一步筛选好的做题数据。为了让分布更加平滑，我们可以对每个分组的频率值进行和下一个分组的频率求平均：

```
# Get histogram of original data
y, x = np.histogram(f_duration, bins=200, normed=True)
x = (x + np.roll(x, -1))[:-1] / 2.0
```

接着就是为每个分布进行拟合了，每一次拟合后我们都比较一下均方误差，并把误差最低的分布记录下来，最终把所有模型的拟合情况展示出来：


```
SELECT u_user, FIRST(duration) AS duration
FROM dw.factr_user_problem
WHERE  problem_id = '00ec42f6-57e9-11e7-bc9d-9bcf2b652f72'
GROUP BY u_user
LIMIT 10000
```

	u_user	duration
1	5aa3181d1a289c288d9df7ee	14
2	59fb225cd0d4e03d621b4708	5
3	59cf7a8068b5f71f6c1b54af	27
4	59e8b54359e5d906005c9526	15
5	59f051615d4290276088&c61	40
6	5ab242b54661e928af7997c0	14
7	59abb794b64f714be5d73b09	84
8	5901adaa9725c5b7e050da6	62
9	5a77e93de150b574950e39a8	42
10	5a55f37244589e7a70af3930	103

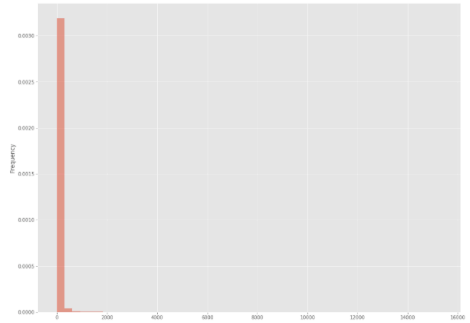
通过 Hive 下载 CSV 到本地，然后在 Python 里导入：

```
import pandas as pd

df = pd.read_csv('duration.csv')
df.columns = ['uid', 'duration']

df.head()
```

	uid	duration
0	5a94fae988095227d0439a01	61
1	5a619af83dfd886190dc3348	22
2	59e0c66216543644d5827cde	3
3	59f4767d5e14b5071fb7b65a	31
4	59be1a6ed4bbfb078f408307	24

这个任务不需要用户的 ID，因此我们可以只拿取 duration 字段，先做个频率分布直方图出来看看：

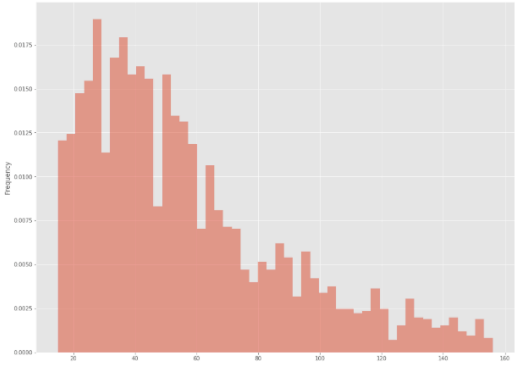
```
import matplotlib

duration = df['duration']
matplotlib.rcParams['figure.figsize'] = (16., 12.)
matplotlib.style.use('ggplot')

ax = duration.plot(kind='hist', bins=50, normed=True, alpha=0.5)
```



这样看起来并不理想，主要原因是个别的极大值数据使得整个分布 Tail 很长。我们可以做个简单的过滤，先根据绝对阈值筛选，再把标准差超

过一定阈值的数据都移除掉：

```
def reject_outliers(data, m=2, min_thres=10, max_thres=200):
    d = data[data <= max_thres]
    d = d[d >= min_thres]
    return d[abs(d - np.mean(d)) < m * np.std(d)]

f_duration = reject_outliers(duration, m=1.8, min_thres=15, max_thres=300)
ax = f_duration.plot(kind='hist', bins=50, normed=True, alpha=0.5)
```



接下来就可以寻找最拟合做题用时分布的模型了。最直接的做法是找到所有的统计分布模型，然后用每个模型拟合数据，最后返回误差最小的模型及参数。

在 Scipy 的 statsmodel 中，有很多统计模型，我们先把这些模型全部放入一个列表中：