Ортонормирование данных вместо регуляризации

Усков Филипп

NLA, AIMasters, 2024

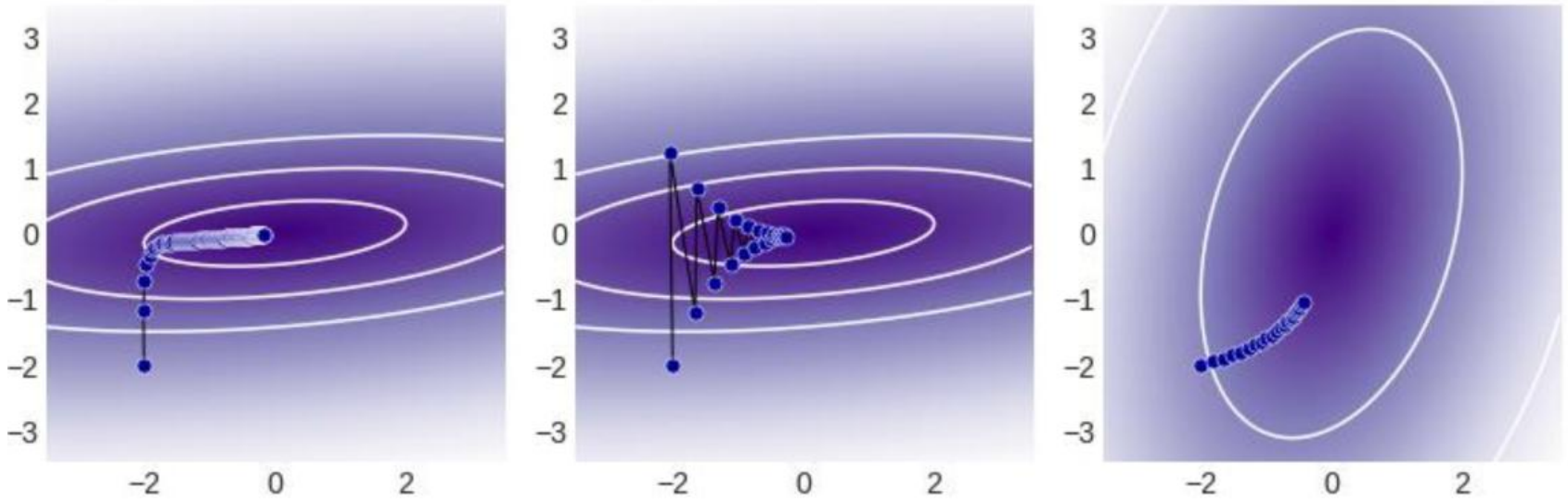
Плохая обусловленность это плохо

$$L = \sum_i L(f(X_{i\alpha} w_\alpha), y_i)$$

$$\frac{\partial L}{\partial w_\mu} = \sum_i L'(f(X_{i\alpha} w_\alpha), y_i) f'(X_{i\beta} w_\beta) X_{i\mu}$$

$$\frac{\partial^2 L}{\partial w_\mu \partial w_\nu} = \sum_i L''(f(X_{i\alpha} w_\alpha), y_i)(f'(X_{i\beta} w_\beta))^2 X_{i\mu} X_{i\nu} +$$

$$+ \sum_i L'(f(X_{i\alpha} w_\alpha), y_i) f''(X_{i\beta} w_\beta) X_{i\mu} X_{i\nu} = \sum_i k_i X_{i\mu} X_{i\nu}$$



Что делать?

- QR $y \sim f(Xw) = f(QRw) = f(Qw') \qquad w = R^{-1}w'$
- SVD $y \sim f(Xw) = f(U\Sigma V^T w) = f(Uw') \qquad w = V\Sigma^{-1}w'$
- PCA $X - \bar{X} = (U\Sigma)V^T \to U(\Sigma V^T)$

- регуляризация

$$\min \sum_i L(f(X_{i\alpha}w_\alpha), y_i) \neq \min \left(\sum_i L(f(X_{i\alpha}w_\alpha), y_i) + \lambda \|w\| \right)$$

QR

- $O(nd^2)$
- Однозначность:
- Порядок столбцов Q задаётся порядком X
- Можно определять ЛЗ столбцы X:

```
Q,R = np.linalg.qr(X1)
sign = (np.diag(R)>0).astype(float)*2-1
R = (sign*R.T).T
```

```
d = np.abs(np.diag(R))
np.fill_diagonal(R,0)
dd = np.linalg.norm(R,ord=np.inf, axis=0)
XX1 = X1[:,1e6*d > dd]
# ^ может исключить лишние (не ЛЗ) столбцы
# с вероятностью ~0
```

SVD/PCA

- $O(nd^2)$
- `svd_solver='covariance_eigh'` x3..x5 быстрее
- Порядок столбцов определяется сингуляными значениями

```
center = 0.5
threshold = 1e-4
mask = pca.singular_values_ > \
pca.singular_values_[int(len(pca.singular_values_)*center)]\
*threshold
XX1 = U[:,mask]
```

Тесты оптимайзеров sklearn

X.shape=(215290,105)
время QR разложения – 14с
tol=1e-8

метод	cond=298 , penalty=None	cond=298 , penalty=default	cond=1, penalty=None
lbfgs	>54 s	>53 s	2.7 s
liblinear	--	163 s	--
newton-cg	129 s	81 s	10.2 s
newton-cholesky	65 s	55 s	44 s
sag	>212 s	>213 s	>213 s
saga	>248 s	>273 s	>252 s

Выбираем оптимальный метод

- normPCA_regression: https://gist.github.com/FeelUsM/a8f2922167693ab4a861d8fd42b5bf92
- X.shape=(215290, 376), cond(X)=inf

метод	Время (разложение + минимизация + inference)	roc_auc
QR full	2m19s+59s+0s	0.51108637
QR+QR with cut	(2m19s+1m57s)+8s+0s	0.75039031
PCA+PCA_norm(n_components=335)	(32s+1m32s)+9s+12s	0.75039014
normPCA_regression	1m18s	0.75039014
direct(newton-cholesky)	0s+10m50s+0s	0.75010729

А что если это применить в нейронках?

- Нужно QR разложение
- Надо по строке X обновить R и получить строку Q не сложнее чем за $O(d^2)$
- Желательно скользящим образом (в R не хранить информацию с самого начала процесса, а постепенно её забывать)

Стохастическое QR разложение (по столбцам)

$$\operatorname{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u},$$

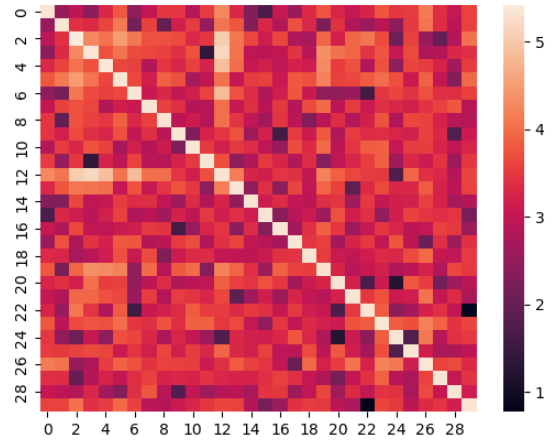
$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{v}_2 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{v}_3 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \operatorname{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
\mathbf{u}_4 &= \mathbf{v}_4 - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \operatorname{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \operatorname{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 &= \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \operatorname{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
\end{aligned}$$

```
reg = LinearRegression(fit_intercept=False)
reg.fit([e1,e2,e3],v4)
u4 = v4 - reg.predict([e1,e2,e3],v4)
```

Стохастическое QR разложение (по строкам)

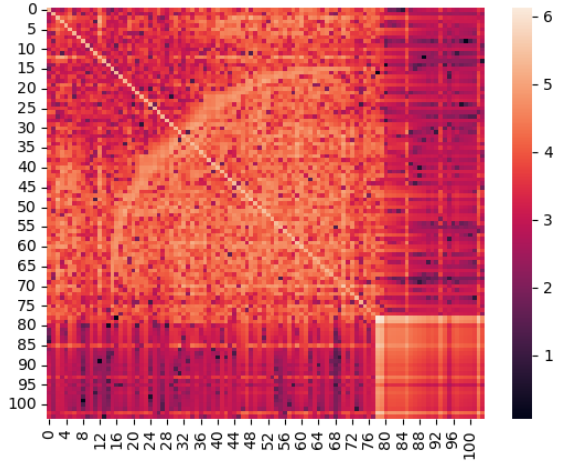
```
S = np.zeros((X1.shape[1],X1.shape[1]))
sqd = X1[0,:]**2

for i in tqdm(range(X1.shape[0])):
    alpha = 0.01/(i+2)**0.3 # Learning rate
    X = X1[i]

    R = np.eye(X1.shape[1])*np.sqrt(sqd)+S

    Q = sp.linalg.solve_triangular(R.T, X, lower=True)

    dS = - Q[:,None]*(X-Q@S)[None,:]
    np.fill_diagonal(dS,0)
    S -= alpha* np.triu(dS)

    sqd = (1-alpha)*sqd + alpha*(X-Q@S)**2

R = np.eye(X1.shape[1])*np.sqrt(sqd)+S
```

```
QTQ = Q.T @ Q
sns.heatmap(np.log10(np.abs(QTQ)))
```



обусловленность:5.1935

исходной: 135.29813

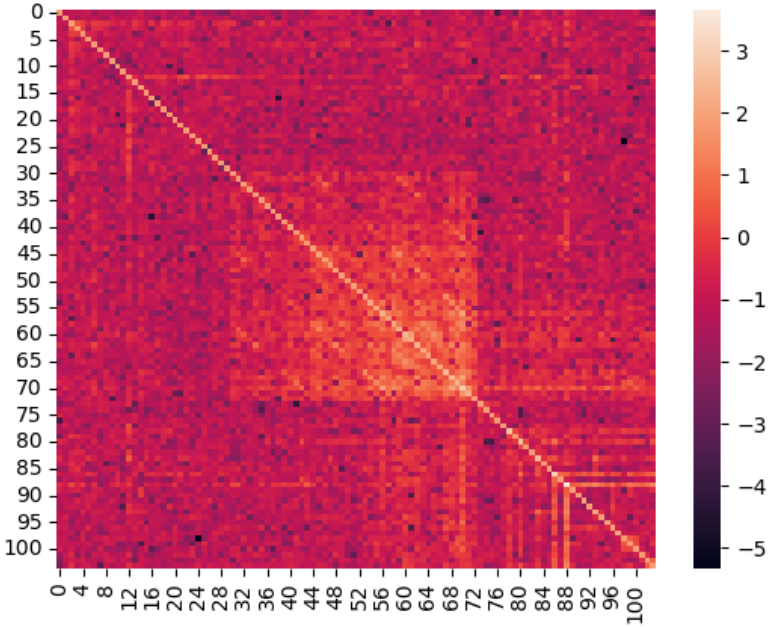


Скользящий batch

```
lr= 1
p = 0.3
n = 30

d = int(X1.shape[1] *n)
Q,RR = np.linalg.qr(X1[:d,:])
sign = (np.diag(RR)>0).astype(float)*2-1
RR = (sign*RR.T).T

for i in tqdm(range(d, X1.shape[0]-d,d)):
    alpha = lr/(i+2)**p # Learning rate

    Q,R = np.linalg.qr(X1[i:i+d,:])
    sign = (np.diag(R)>0).astype(float)*2-1
    R = (sign*R.T).T

    RR = (1-alpha)*RR + alpha*R
R = RR
```

обусловленность:8.219

исходной: 297.780



scipy.linalg.qr_insert()

- Daniel, J. W., Gragg, W. B., Kaufman, L. & Stewart, G. W. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. Math. Comput. 30, 772-795 (1976).

$$G = \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix}$$

	обусловленность
X	297.780
Обычный QR	1.0000000000000138
update_R иниц. на R=QR(X[:d , :])	1.0000000000376412
update_R иниц. на R=I	1.0139012946617278

```
def update_R(R,x):
    q = np.zeros(x.shape)
    v = 1.
    n = len(x)
    for l in range(n):
        # x[l]:=0
        c,s = compute_reflector(R[l,l],x[l])
        apply_reflector(c,s, R[l,l:],x[l:])
        apply_reflector(c,s, q[l], v)
    return q
```

Эксперименты на нейросетях

- Их пока нет
- Но вы держитесь

- Всего доброго

- Эксперименты:https://github.com/FeelUsM/QR-PCA/blob/master/QR-PCA.ipynb