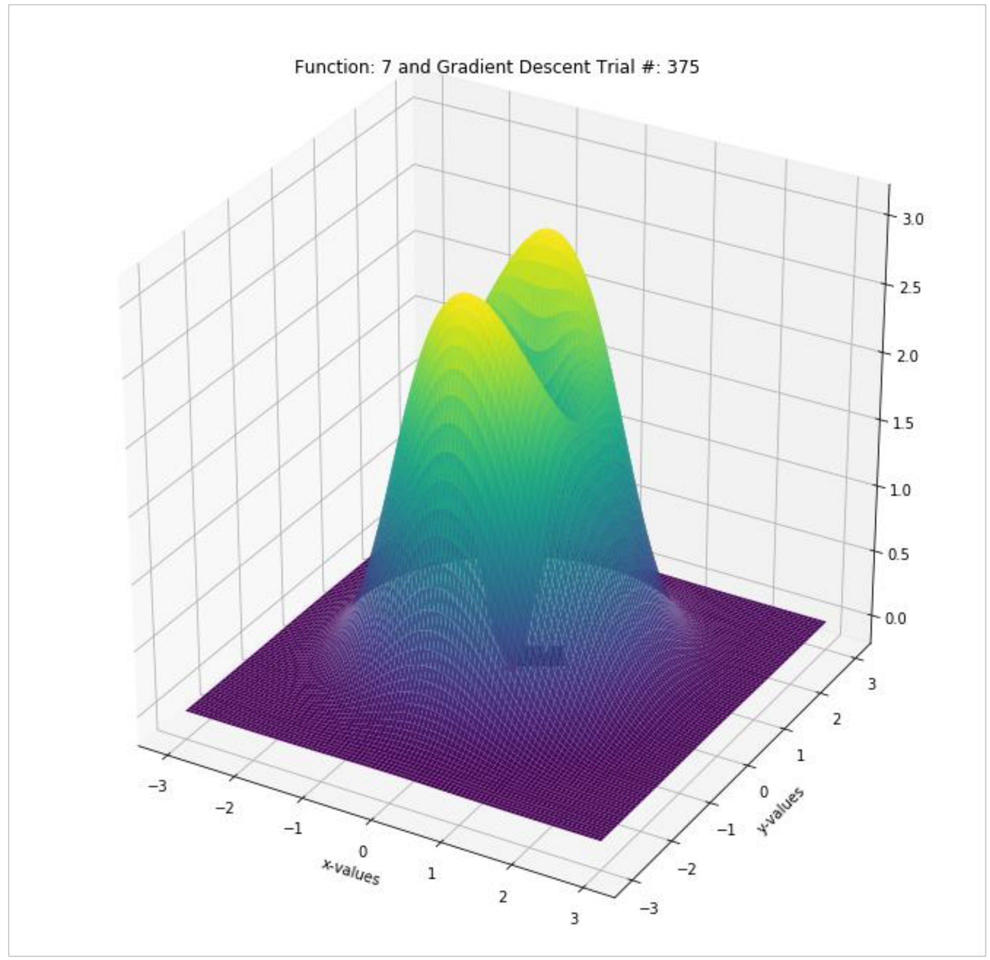
# How to Escape Saddle Points Efficiently

By: Carlos Quintero
and Sean Farrell

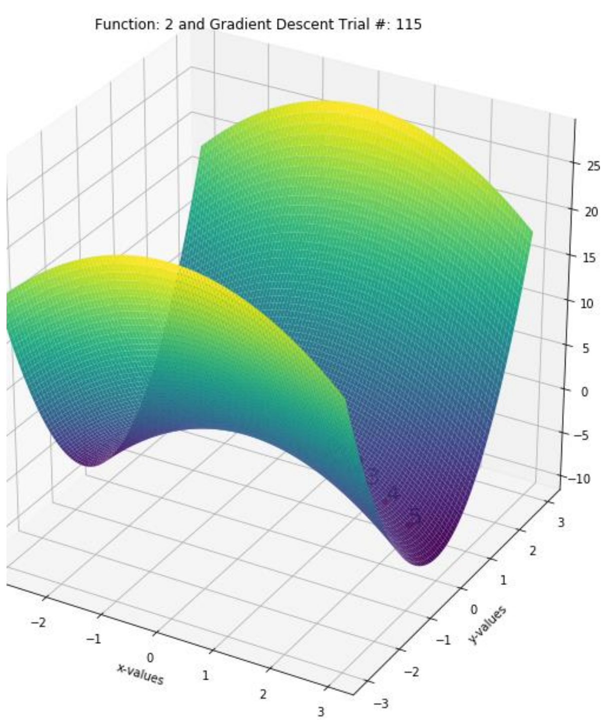
# Stationary Points

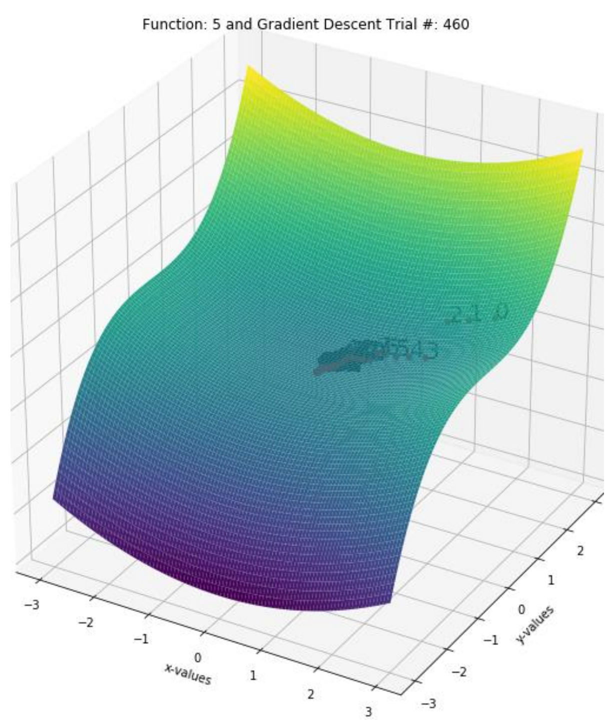
Twice differentiable function $f$, with stationary points at $x$ ($\nabla f(x) = 0$):

- **Local minimum** - eigenvalues of $\nabla^2 f(x)$ are all positive
- **Local maximum** - eigenvalues of $\nabla^2 f(x)$ are all negative
- **Saddle point** - eigenvalues of $\nabla^2 f(x)$ are not all positive or negative



# Strict and Degenerate Saddle Points



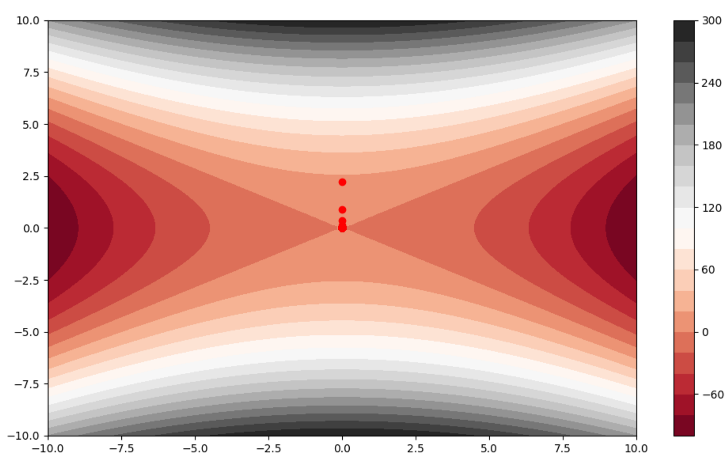

- A stationary point of $f$ is a **strict** saddle point if

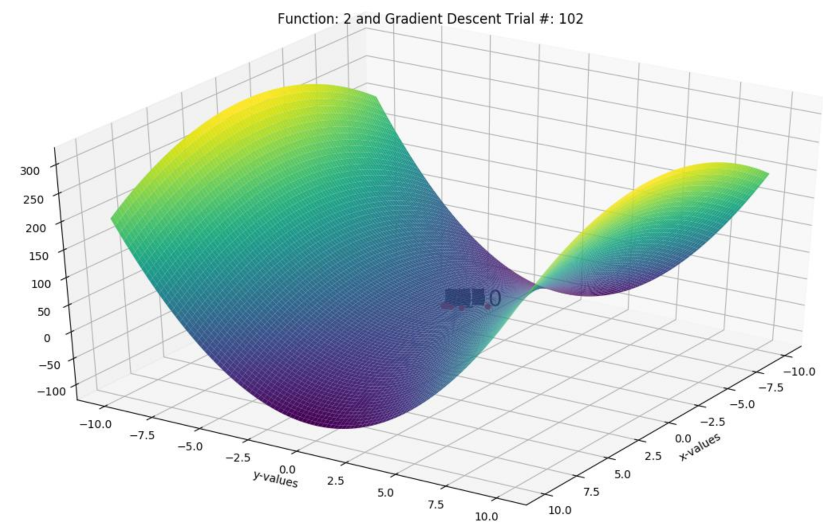
$$\lambda_{min}\left(\nabla^2 f(x)\right) \leq 0$$

- Otherwise it is a degenerate saddle point

# GD on strict saddle point

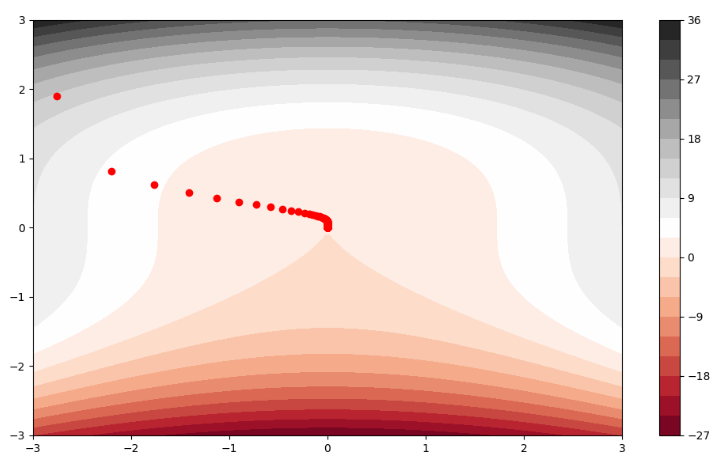
- $f(x, y) = -x^2 + 3y^2$
- One saddle point in $x = (0, 0)$

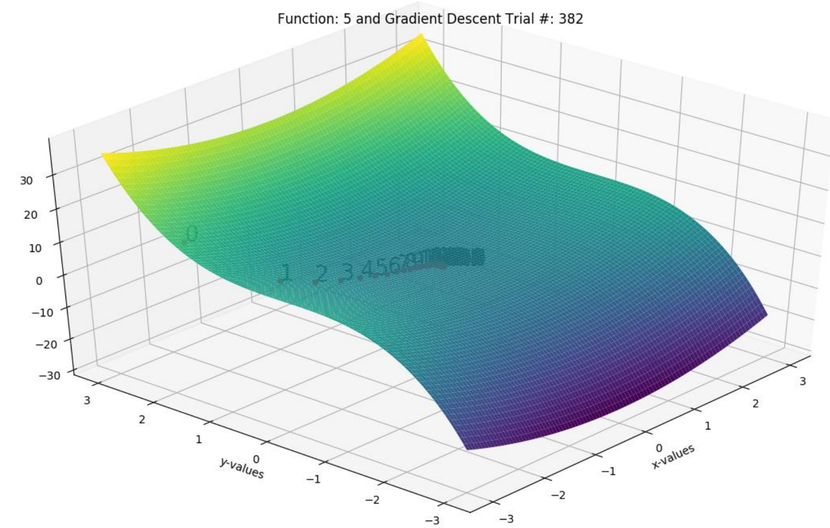


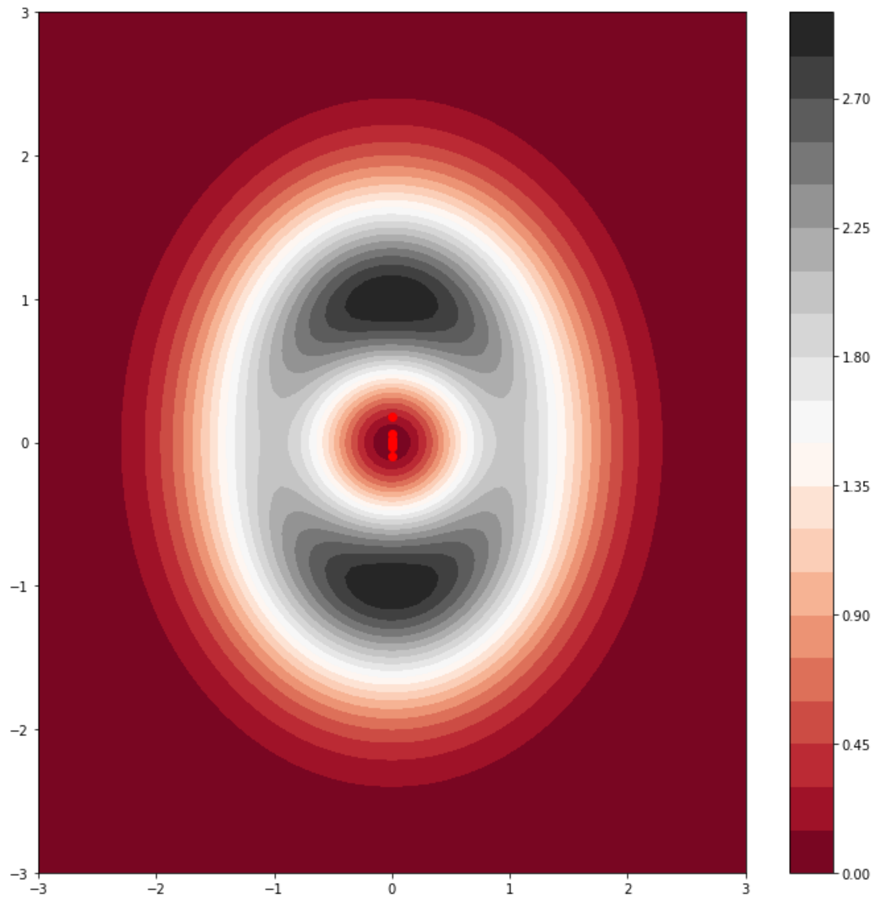


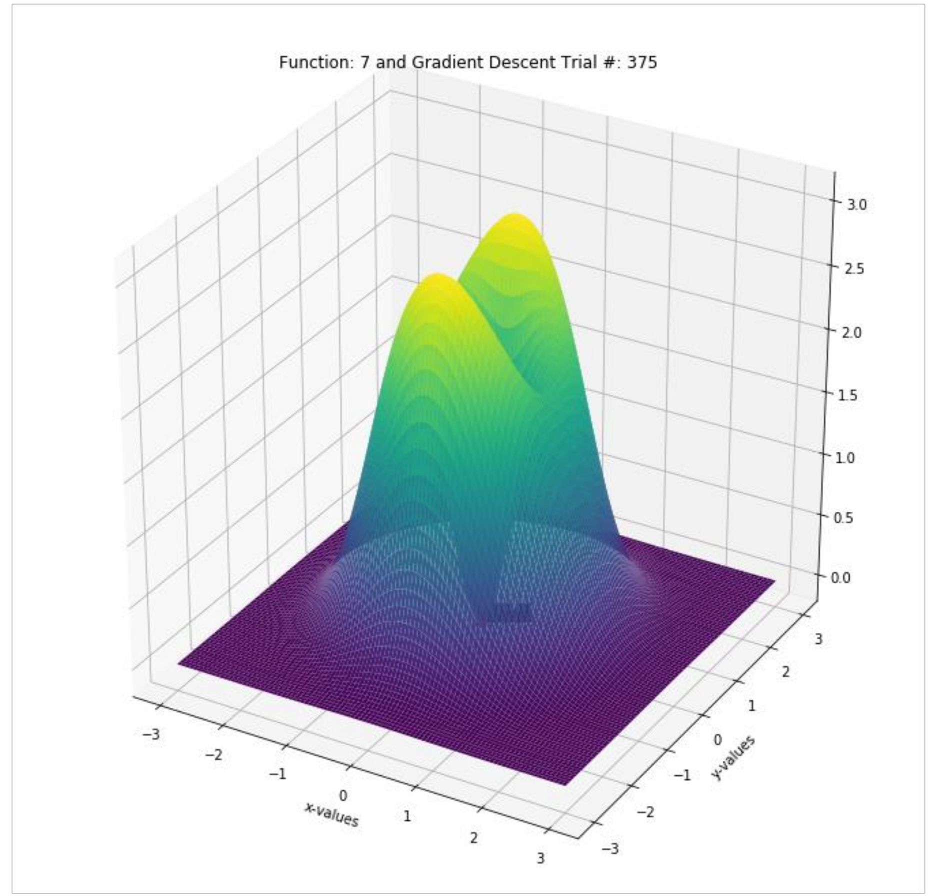
# GD on strict saddle point

- $f(x, y) = x^2 + y^3$
- Degenerate saddle point in $x = (0, 0)$

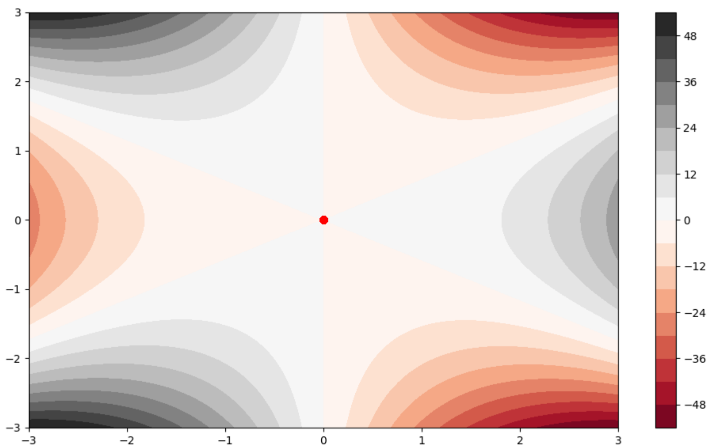


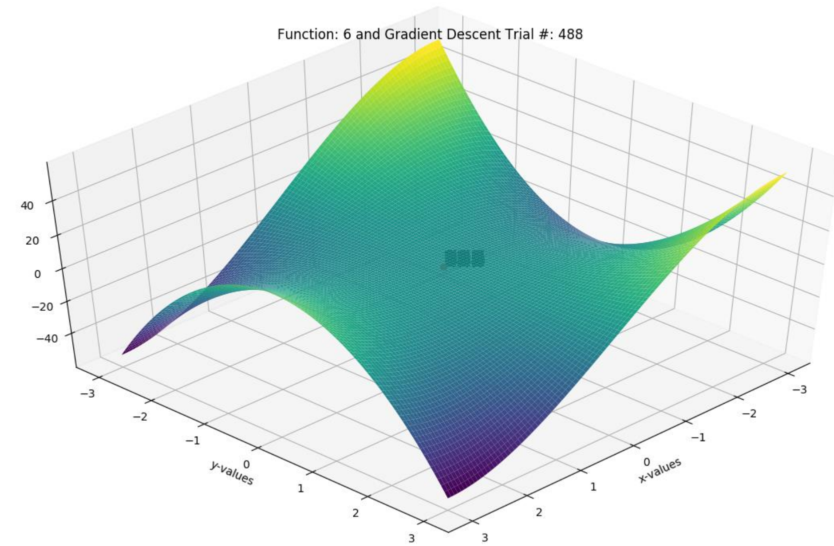



# Convergence to saddle points?









# Monkey Saddle

- $f(x) = x^3 - 3xy^2$
- Degenerate saddle point in $x = (0, 0)$

# Noisy gradient

- Variant of SGD with random noise added each iteration
- Proposed main benefit is guarantee of noise in every direction if not already, induce exploration around local neighborhood of saddle points

**Algorithm 1** Noisy Stochastic Gradient

**Require:** Stochastic gradient oracle $SG(w)$, initial point $w_0$, desired accuracy $\kappa$.
**Ensure:** $w_t$ that is close to some local minimum $w^\star$.

Choose $\eta = \min\{\tilde{O}(\kappa^2 / \log(1/\kappa)), \eta_{\max}\}, T = \tilde{O}(1/\eta^2)$
**for** $t = 0$ to $T - 1$ **do**
  Sample noise $n$ uniformly from unit sphere.
  $w_{t+1} \leftarrow w_t - \eta(SG(w) + n)$

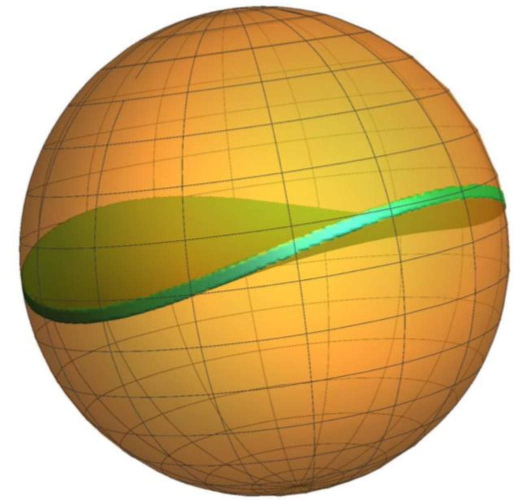
[Ge et al., 2015]

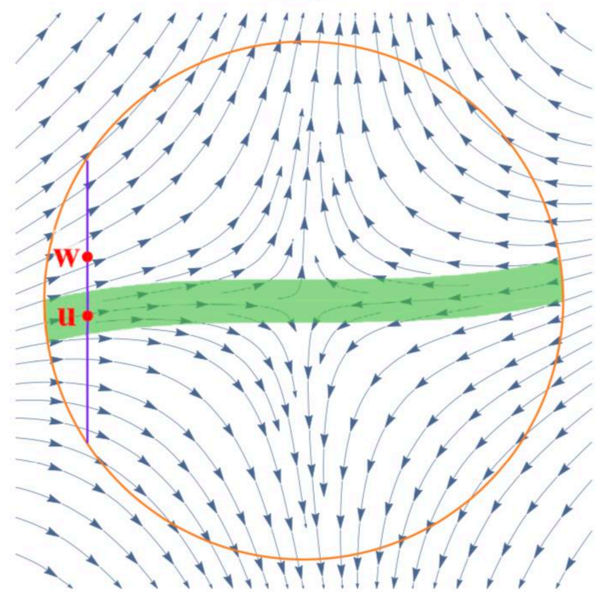
# Perturbed Gradient Descent (PGD)

- Main contribution is identifying stuck region of strict saddle point and perturbing gradient to avoid it with high probability

- Definition of second order stationary point

$$\|\nabla f(x)\| \le \epsilon, \text{ and } \lambda_{min}\left(\nabla^2 f(x)\right) \ge -\sqrt{\rho\epsilon}$$






Figures from [Jin et al., 2017]

# PGD Algorithm

**Algorithm 2** Perturbed Gradient Descent: PGD$(\mathbf{x}_0, \ell, \rho, \epsilon, c, \delta, \Delta_f)$

$\chi \leftarrow 3\max\{\log(\frac{d\ell\Delta_f}{c\epsilon^2\delta}), 4\},\ \eta \leftarrow \frac{c}{\ell},\ r \leftarrow \frac{\sqrt{c}}{\chi^2}\cdot\frac{\epsilon}{\ell},\ g_{\text{thres}} \leftarrow \frac{\sqrt{c}}{\chi^2}\cdot\epsilon,\ f_{\text{thres}} \leftarrow \frac{c}{\chi^3}\cdot\sqrt{\frac{\epsilon^3}{\rho}},\ t_{\text{thres}} \leftarrow \frac{\chi}{c^2}\cdot\frac{\ell}{\sqrt{\rho\epsilon}}$
$t_{\text{noise}} \leftarrow -t_{\text{thres}} - 1$
**for** $t = 0, 1, \ldots$ **do**
  **if** $\|\nabla f(\mathbf{x}_t)\| \le g_{\text{thres}}$ and $t - t_{\text{noise}} > t_{\text{thres}}$ **then**
    $\tilde{\mathbf{x}}_t \leftarrow \mathbf{x}_t, \quad t_{\text{noise}} \leftarrow t$
    $\mathbf{x}_t \leftarrow \tilde{\mathbf{x}}_t + \xi_t, \qquad \xi_t$ uniformly $\sim \mathbb{B}_0(r)$
  **if** $t - t_{\text{noise}} = t_{\text{thres}}$ and $f(\mathbf{x}_t) - f(\tilde{\mathbf{x}}_{t_{\text{noise}}}) > -f_{\text{thres}}$ **then**
    **return** $\tilde{\mathbf{x}}_{t_{\text{noise}}}$
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta\nabla f(\mathbf{x}_t)$

[Jin et al., 2017]

**Algorithm 1** Perturbed Gradient Descent (PGD)

**Input:** $\mathbf{x}_0$, step size $\eta$, perturbation radius $r$.
  **for** $t = 0, 1, \ldots,$ **do**
    $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta(\nabla f(\mathbf{x}_t) + \xi_t), \qquad \xi_t \sim \mathcal{N}(\mathbf{0}, (r^2/d)\mathbf{I})$

[Jin et al., 2019]

# PGD Theorem

- **Theorem**: Assume that $f(\cdot)$ satisfies $l$-gradient Lipschitz and $\rho$-Hessian Lipschitz. Then there exists an absolute constant $c_{max}$ such that, for any $\delta > 0,\ \epsilon \leq \frac{l^2}{\rho},\ \Delta_f \geq f(x_0) - f^*$, and constant $c \leq c_{max}$, PGD($x_0, l, \rho, \epsilon, c, \delta, \Delta_f$) will output an $\epsilon$ – second order stationary point, with probability $1 - \delta$, and terminate in the following number of iterations:

$$O\left(\frac{l(f(x_0) - f^*)}{\epsilon^2} log^4\left(\frac{dl\Delta_f}{\epsilon^2\delta}\right)\right)$$

[Jin et al., 2017]

# Perturbed Stochastic Gradient Descent (PSGD)

**Algorithm** Perturbed Stochastic Gradient Descent (PSGD)

**Input:** $\mathbf{x}_0$, step size $\eta$, perturbation radius $r$.

**for** $t = 0, 1, \ldots,$ **do**

sample $\theta_t \sim \mathcal{D}$

$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta(\mathbf{g}(\mathbf{x}_t; \theta_t) + \xi_t), \qquad \xi_t \sim \mathcal{N}(\mathbf{0}, (r^2/d)\mathbf{I})$

**Algorithm** Mini-batch Perturbed Stochastic Gradient Descent (Mini-batch PSGD)

**Input:** $\mathbf{x}_0$, step size $\eta$, perturbation radius $r$.

**for** $t = 0, 1, \ldots,$ **do**

sample $\{\theta_t^{(1)}, \cdots \theta_t^{(m)}\} \sim \mathcal{D}$

$\mathbf{g}_t(\mathbf{x}_t) \leftarrow \sum_{i=1}^{m} \mathbf{g}(\mathbf{x}_t; \theta_t^{(i)})/m$

$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta(\mathbf{g}_t(\mathbf{x}_t) + \xi_t), \qquad \xi_t \sim \mathcal{N}(\mathbf{0}, (r^2/d)\mathbf{I})$

[Jin et al., 2019]

# Perturbed Stochastic Gradient Descent (PSGD)

**Theorem:** Let the function $f$ be $l$-gradient Lipschitz and $\rho$-Hessian Lipschitz. For any $\epsilon, \delta > 0$, the PSGD algorithm with parameter $(\eta, r)$ will visit an $\varepsilon$-second order stationary point at least once in the following number of iterations, with probability at least $1 - \delta$:

$$\tilde{O}\left(\frac{l(f(x_0) - f^*)}{\epsilon^2}\mathfrak{R}\right)$$

[Jin et al., 2019]

# Algorithm Complexity

| Setting | Algorithm | Iterations | Guarantees |
|---|---|---|---|
| Non-stochastic | GD [Nesterov, 2000] | $\mathcal{O}(\epsilon^{-2})$ | first-order stationary point |
| | **PGD** | $\tilde{\mathcal{O}}(\epsilon^{-2})$ | second-order stationary point |
| Stochastic | SGD [Ghadimi and Lan, 2013] | $\mathcal{O}(\epsilon^{-4})$ | first-order stationary point |
| | **PSGD** (*with* Assumption C) | $\tilde{\mathcal{O}}(\epsilon^{-4})$ | second-order stationary point |
| | **PSGD** (*no* Assumption C) | $\tilde{\mathcal{O}}(d\epsilon^{-4})$ | second-order stationary point |

[Jin et al., 2019]

# On the sufficiency of Second-Order Stationarity (Nonconvex problems in ML)

- Tensor Decomposition (Ge et al., 2015)
- Dictionary Learning (Sun et al., 2016a)
- Phase Retrieval (Sun et al., 2016b)
- Synchronization and MaxCut (Bandeira et al., 2016)
- Smooth Semidefinite Programs (Boumal et al., 2016)
- Matrix Sensing (Bhojanapalli et al., 2016)
- Matrix Completion (Get et al., 2016)

# On the sufficiency of Second-Order Stationarity

- All local minima are global minima
- All saddle points have at least one direction with strictly negative curvature

*If a function satisfies (a) all local minima are global minima; (b) all saddle points (including local maxima) are strict saddle points, then all second-order stationary points are global minima*

# Deep Neural Networks

- Very large number of saddle points
- All local minima are global minima [Kagawuchi, 2016]
- Every critical point that is not a global minimum is a saddle point [Kagawuchi, 2016]
- There exists degenerate saddle points for deeper networks [Kagawuchi, 2016]
  - Shallow networks have good saddles ---
- DNN converge to (degenerate) saddle points [Sankar et al., 2017]
  - Experiments on MLP and distilled Resnet show degenerate saddles ---

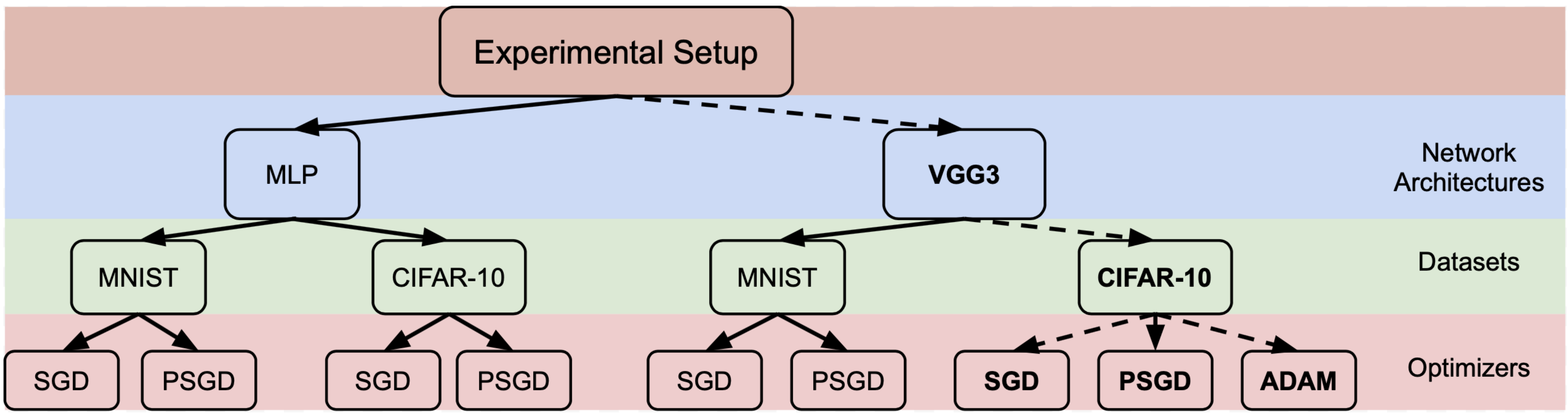
| Algorithm | Iterations (*with* Assumption C) | Iterations (*no* Assumption C) | Simplicity |
|---|---|---|---|
| Noisy GD [Ge et al., 2015] | $d^4\text{poly}(\epsilon^{-1})$ | $d^4\text{poly}(\epsilon^{-1})$ | single-loop |
| CNC-SGD [Daneshmand et al., 2018] | $\tilde{\mathcal{O}}(d^4\epsilon^{-5})$ | $\tilde{\mathcal{O}}(d^4\epsilon^{-5})$ | |
| **PSGD (this work)** | $\tilde{\mathcal{O}}(\epsilon^{-4})$ | $\tilde{\mathcal{O}}(d\epsilon^{-4})$ | |
| *SGD with averaging [Fang et al., 2019] | $\tilde{\mathcal{O}}(\epsilon^{-3.5})$ | × | |
| Natasha 2 [Allen-Zhu, 2018] | $\tilde{\mathcal{O}}(\epsilon^{-3.5})$ | × | double-loop |
| Stochastic Cubic [Tripuraneni et al., 2018] | $\tilde{\mathcal{O}}(\epsilon^{-3.5})$ | × | |
| SPIDER [Fang et al., 2018] | $\tilde{\mathcal{O}}(\epsilon^{-3})$ | × | |
| SRVRC [Zhou and Gu, 2019] | $\tilde{\mathcal{O}}(\epsilon^{-3})$ | × | |

# Hypothesis

PSGD has faster convergence than SGD for neural networks

- There is a gap between theory and practice
- There are contradictory claims

# Experimental setup



# Network Architecture

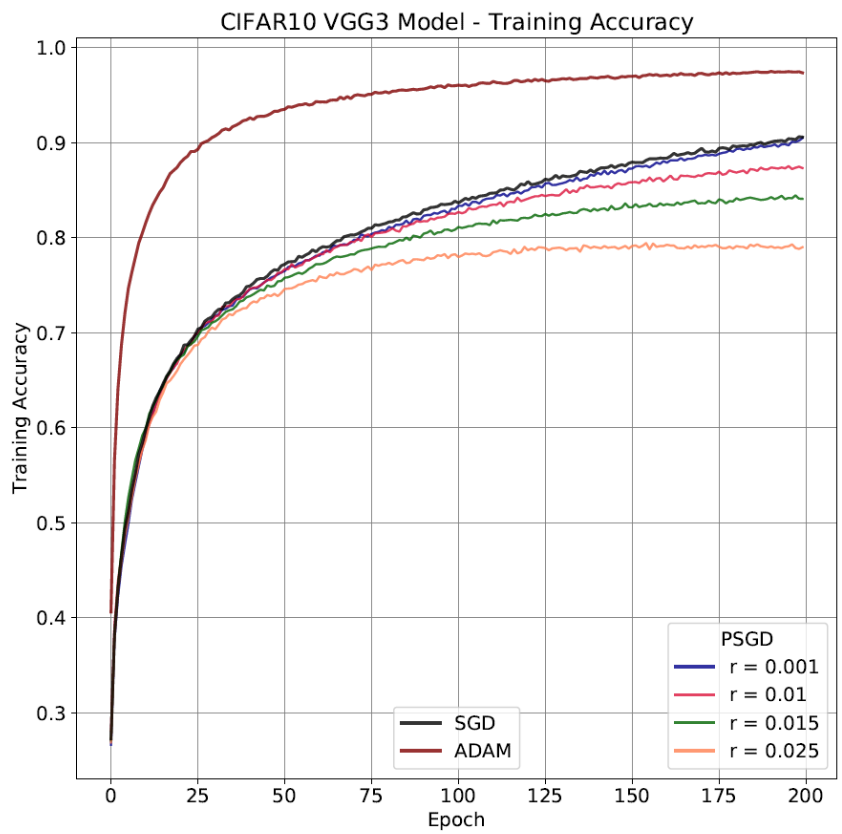
| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 32, 32, 32) | 896 |
| batch_normalization_1 (Batch | (None, 32, 32, 32) | 128 |
| conv2d_2 (Conv2D) | (None, 32, 32, 32) | 9248 |
| batch_normalization_2 (Batch | (None, 32, 32, 32) | 128 |
| max_pooling2d_1 (MaxPooling2 | (None, 16, 16, 32) | 0 |
| dropout_1 (Dropout) | (None, 16, 16, 32) | 0 |
| conv2d_3 (Conv2D) | (None, 16, 16, 64) | 18496 |
| batch_normalization_3 (Batch | (None, 16, 16, 64) | 256 |
| conv2d_4 (Conv2D) | (None, 16, 16, 64) | 36928 |
| batch_normalization_4 (Batch | (None, 16, 16, 64) | 256 |
| max_pooling2d_2 (MaxPooling2 | (None, 8, 8, 64) | 0 |
| dropout_2 (Dropout) | (None, 8, 8, 64) | 0 |
| conv2d_5 (Conv2D) | (None, 8, 8, 128) | 73856 |
| batch_normalization_5 (Batch | (None, 8, 8, 128) | 512 |
| conv2d_6 (Conv2D) | (None, 8, 8, 128) | 147584 |
| batch_normalization_6 (Batch | (None, 8, 8, 128) | 512 |
| max_pooling2d_3 (MaxPooling2 | (None, 4, 4, 128) | 0 |
| dropout_3 (Dropout) | (None, 4, 4, 128) | 0 |
| flatten_1 (Flatten) | (None, 2048) | 0 |
| dense_1 (Dense) | (None, 128) | 262272 |
| dropout_4 (Dropout) | (None, 128) | 0 |
| dense_2 (Dense) | (None, 10) | 1290 |

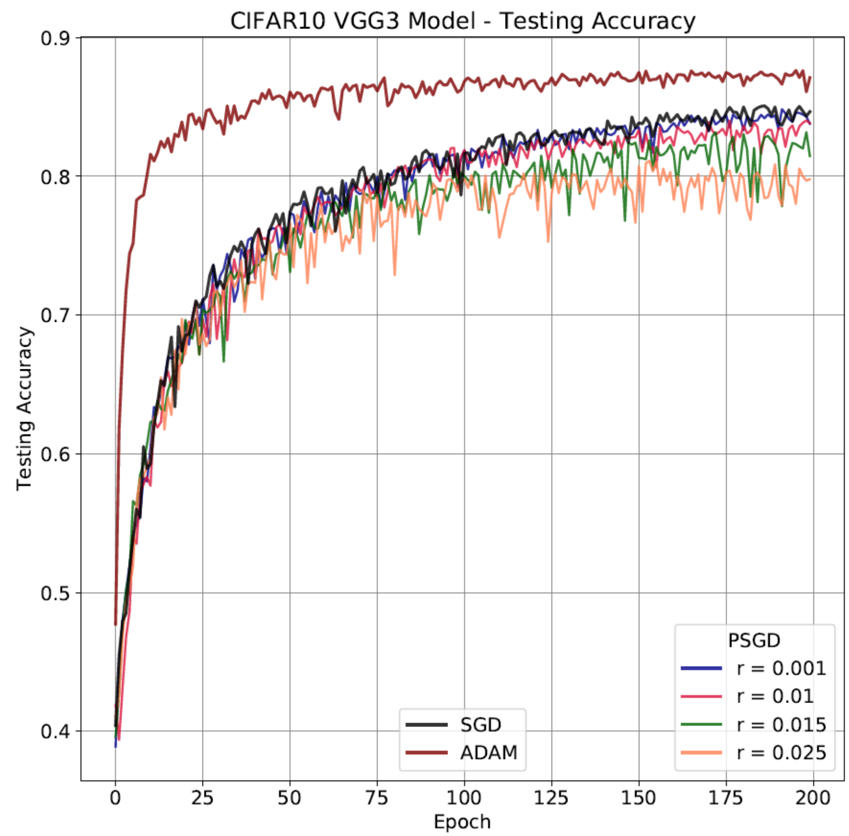
Total params: 552,362
Trainable params: 551,466

# Tools

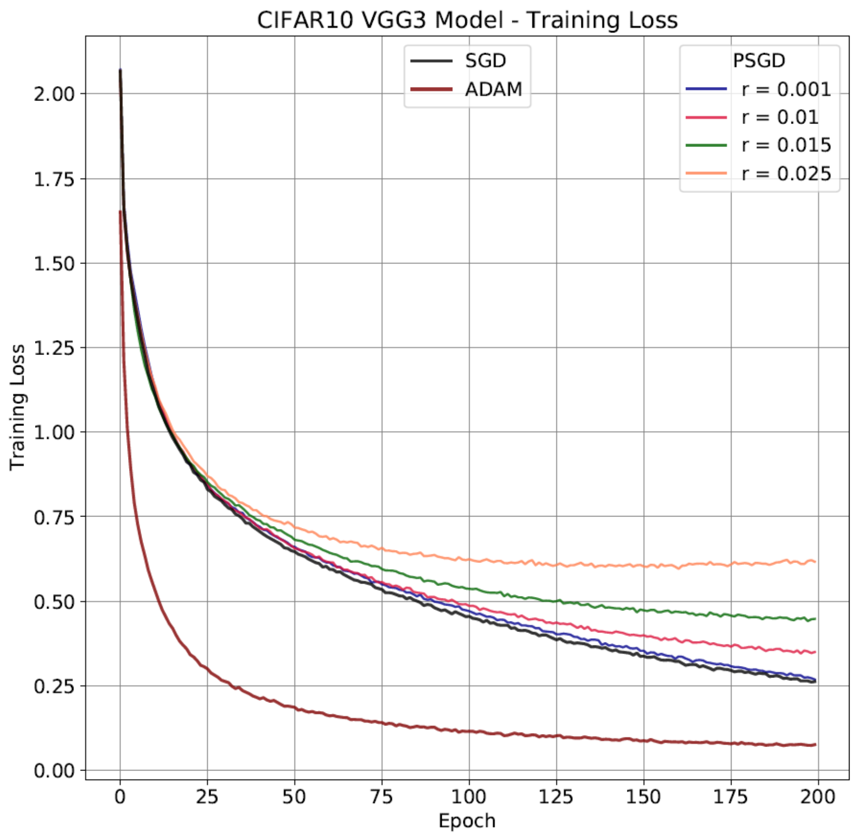
- Google Colab GPUs
- Keras+Tensorflow
- Keras Optimizer API

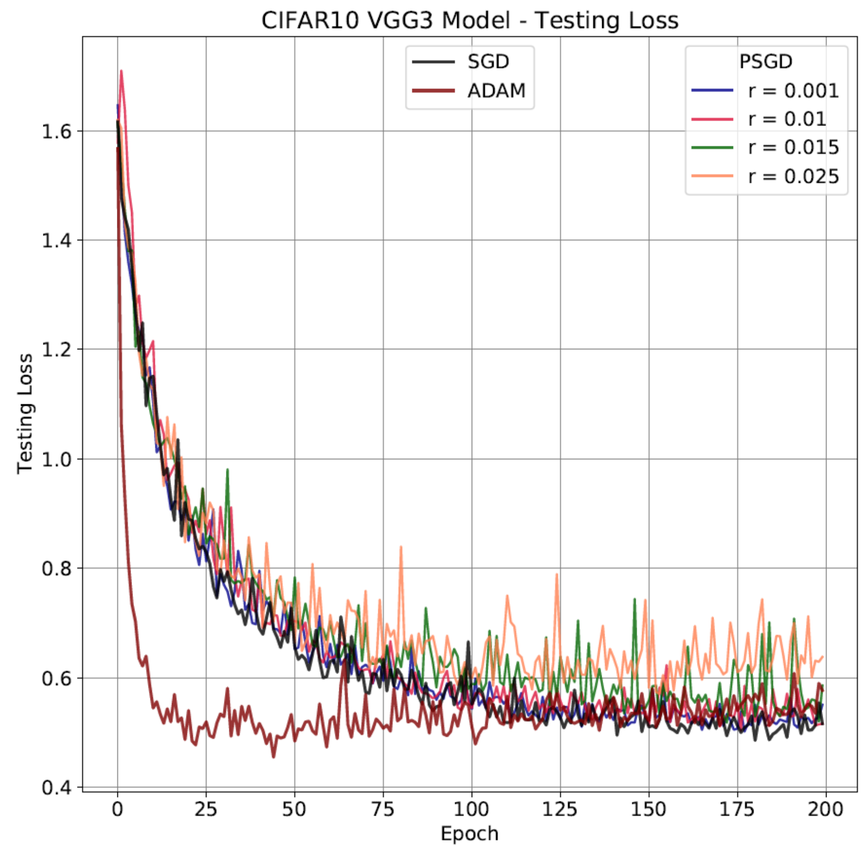
# Experimental Accuracy Results



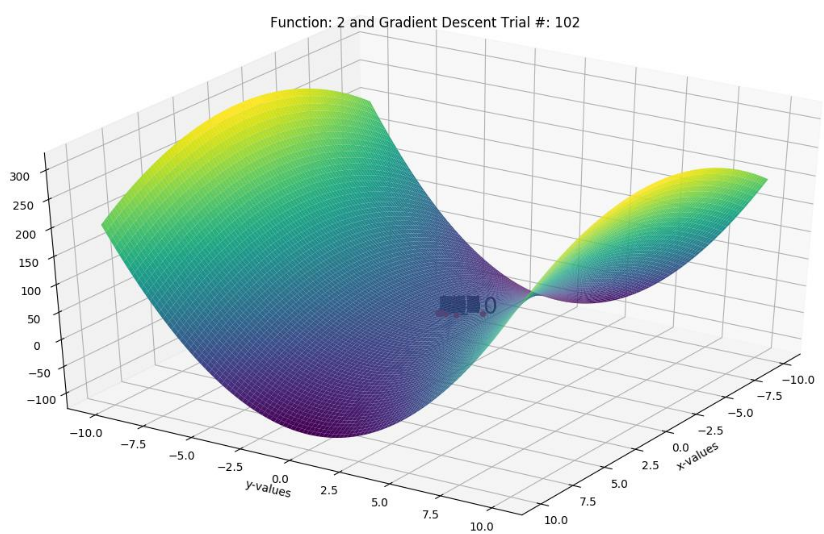
# Experimental Loss Results

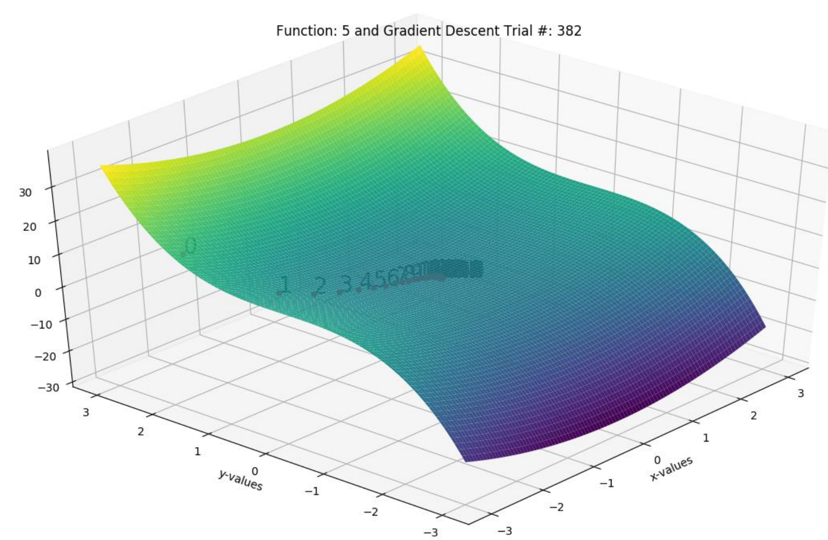


# Conclusion

- Saddle points have high degeneracy in our experiment
- Stochastic nature of SGD is more significant than added perturbations in this case



# References


[1] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping From Saddle Points – Online Stochastic Gradient for Tensor Decomposition," pp. 1–46, Mar. 2015, https://arxiv.org/abs/1503.02101v1.

[2] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. Jordan, "How to Escape Saddle Points Efficiently," pp. 1–35, Mar. 2017., https://arxiv.org/abs/1703.00887v1.

[3] C. Jin, P. Netrapalli, R. Ge, S. M. Kakade, and M. I. Jordan, "On Nonconvex Optimization for Machine Learning: Gradients, Stochasticity, and Saddle Points," pp. 1–31, Sep. 2019,https://arxiv.org/abs/1902.04811v2.

[4] S. Bhojanapalli, B. Neyshabur, and N. Srebro. Global optimality of local search for low rank matrix recovery. In Advances in Neural Information Processing Systems, pages 3873–3881, 2016.

[5] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere I: Overview and the geometric picture. IEEE Transactions on Information Theory, 2016a.

[6] J. Sun, Q. Qu, and J. Wright. A geometric analysis of phase retrieval. In Information Theory (ISIT), 2016 IEEE International Symposium on, pages 2379–2383. IEEE, 2016b.

[7] A. S. Bandeira, N. Boumal, and V. Voroninski. On the low-rank approach for semidefinite programs arising in synchronization and community detection. In Conference on Learning Theory, pages 361–382, 2016.

[8] N. Boumal, V. Voroninski, and A. Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. In Advances in Neural Information Processing Systems, pages 2757–2765, 2016.