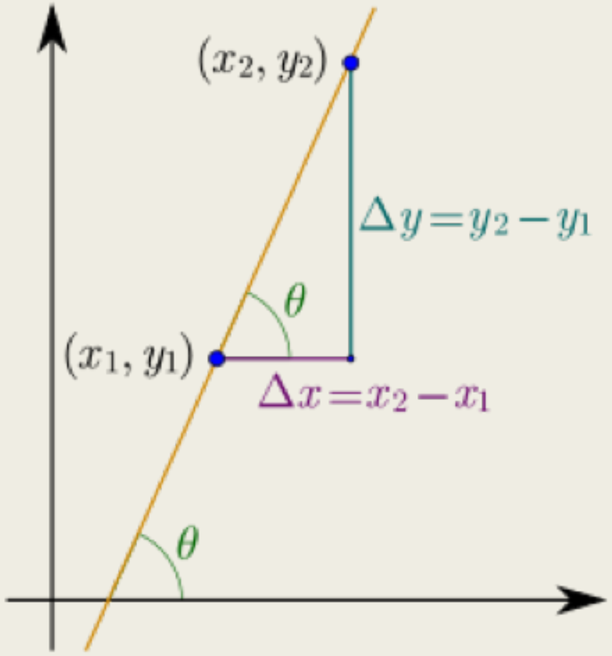
GRADIENT DESCENT

PREREQUISITES

- **Derivative of function** : Rate of change; bivariate

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$



- **Chain Rule**

$$f = f(g)\,;\quad g = g(x)$$

$$\frac{df}{dx} = \frac{df}{dg}\,\frac{dg}{dx}$$

- **Partial Derivatives**

$$f(x,y) = 9ye^{2xy}$$

$$f_x(x,y) = \frac{\partial f(x,y)}{\partial x} = \frac{\partial}{\partial x}[9ye^{2xy}]$$

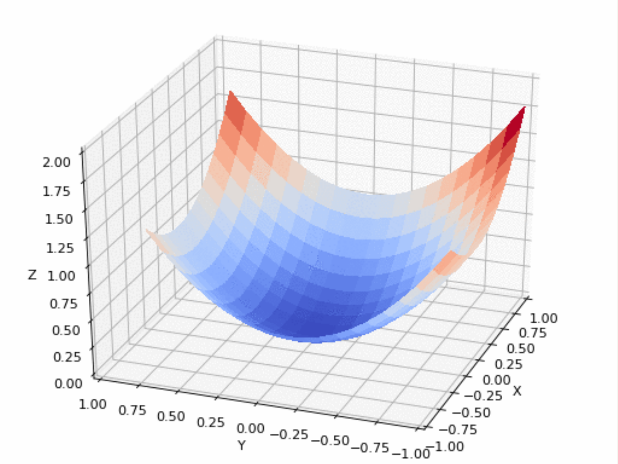
$$= 9ye^{2xy}(2y) \quad = \quad \boxed{18y^2e^{2xy}}$$

GRADIENT DESCENT: OPTIMIZATION TECHNIQUE

- **Gradient** : Multi-variate; vector of derivatives (partial derivatives) for each dimension in the input space; represented as Jacobian matrix

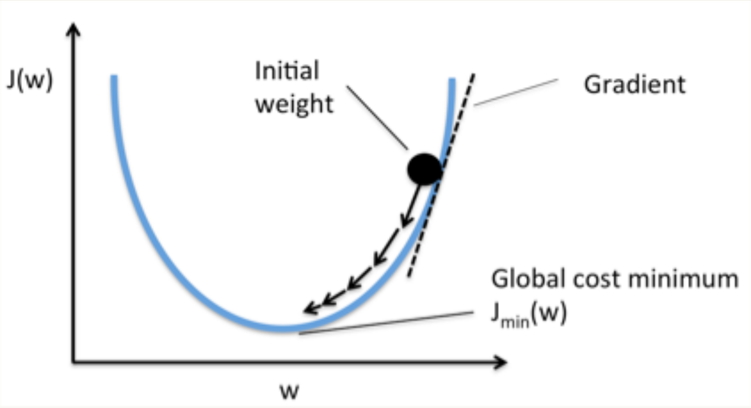
$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

- **Gradient Descent** : First-order optimization technique, steepest (best) step, opposite to the direction of gradient (Minimize)



- **Gradient Ascent** : Same direction as gradient (Maximize)

APPLICATION IN MACHINE LEARNING



- **Loss Function :** How do we know if what we are predicting is accurate or not?

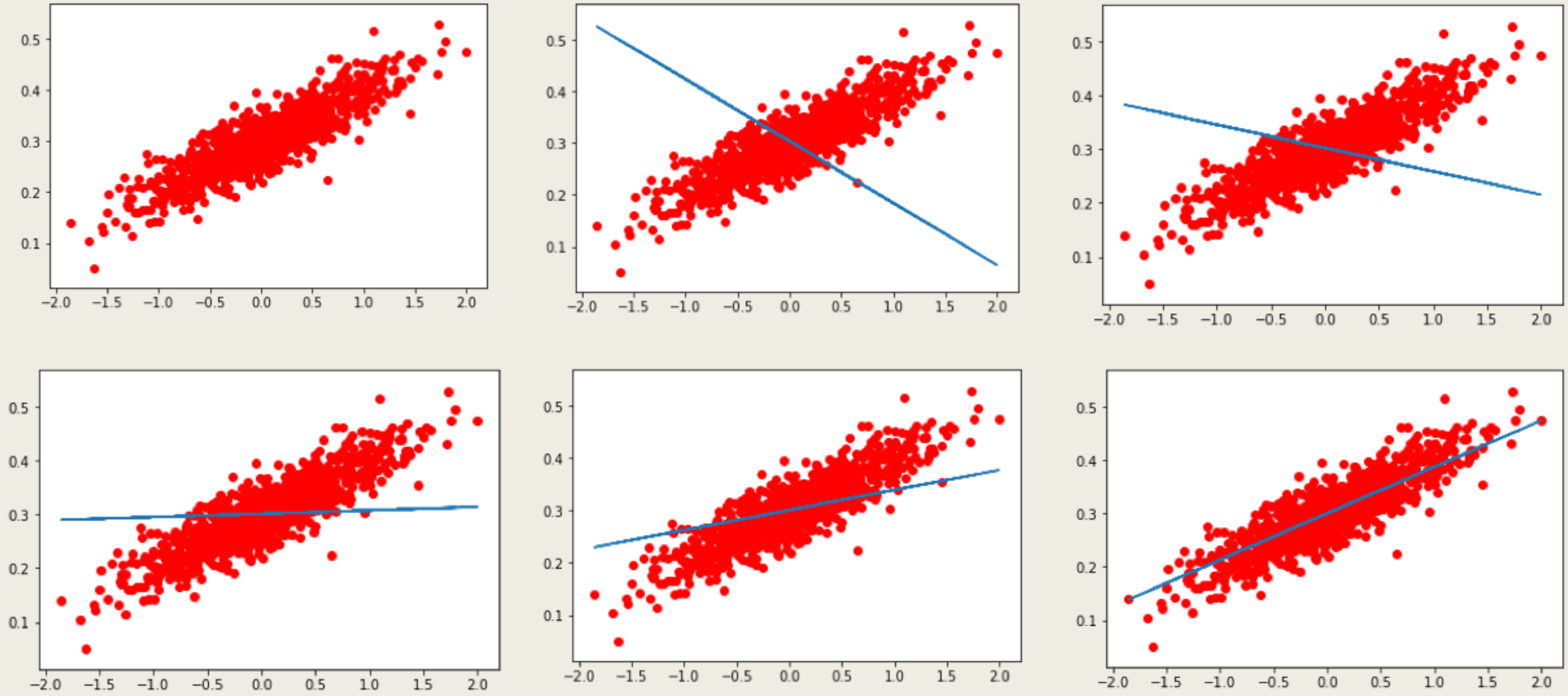
- **Parameters Update Rule :** Optimal set of parameters corresponding to minimum loss

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1, \theta_2)$$

$$\theta_2 := \theta_2 - \alpha \frac{\partial}{\partial \theta_2} J(\theta_1, \theta_2)$$

- **Learning Rate :** How much bigger step one should take?

SIMPLE ILLUSTRATION: REGRESSION



VARIANTS

- **Vanilla (Batch) Gradient Descent :**

(All training examples for one update rule)

Computationally expensive, more accurate

Vanilla (Batch) G.D.

$$\theta_j := \theta_j - \alpha \cdot \underbrace{\frac{\partial}{\partial \theta_j} J(\theta)}_{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}^i - y^i)x_j^i}$$

- **Stochastic Gradient Descent :**

(One training example at a time)

Computationally fast, noisier (random) path,

slightly off

Stochastic G.D.

for i in range(m):

$$\theta_j := \theta_j - \alpha \cdot \boxed{\text{only one example}}$$

$$(\hat{y}^i - y^i)x_j^i$$

- **Mini-Batch Gradient Descent:**

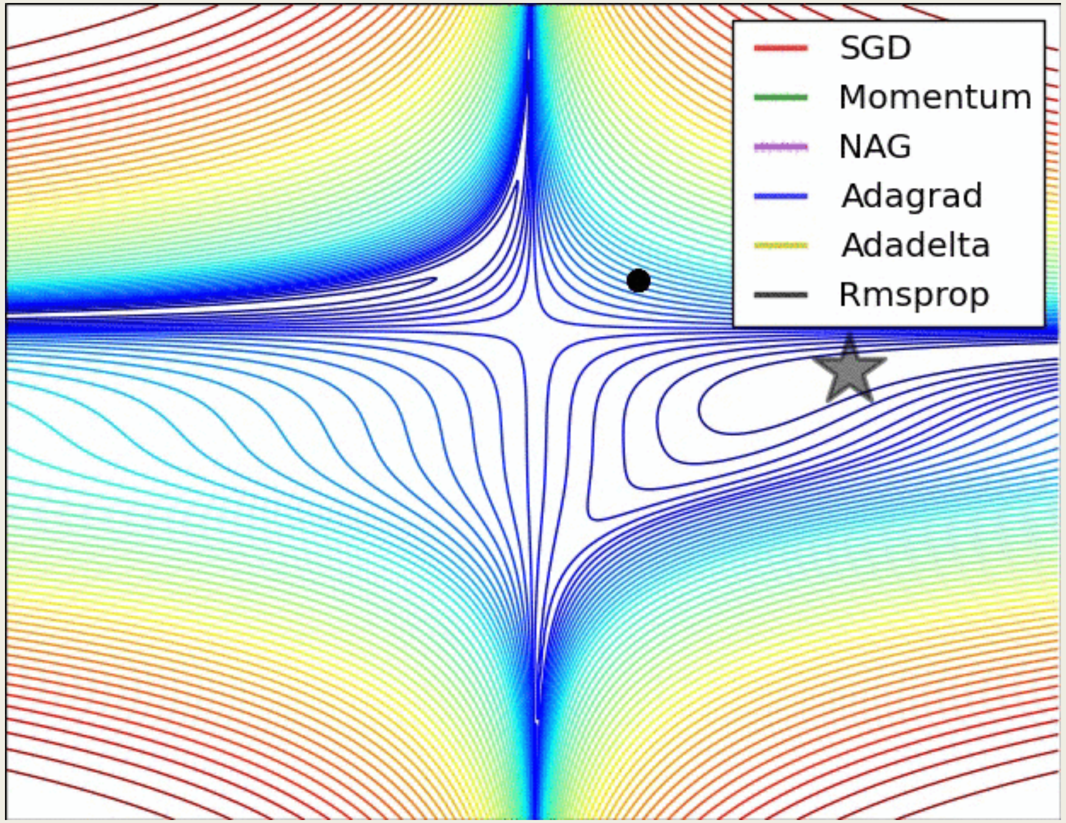
Small batches of training examples. Computationally fast and accurate

DRAWBACKS

- Local Optima
- Finding gradient of high-dimensional loss function
- Learning Rate?

OTHER OPTIMIZATION TECHNIQUES

- RMSProp
- Adam
- Momentum
- Adagrad
- AdaDelta



THANK YOU