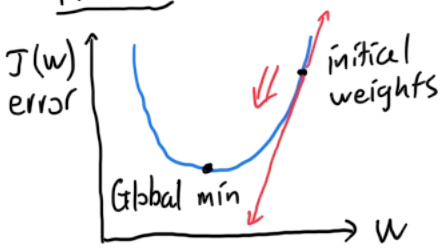
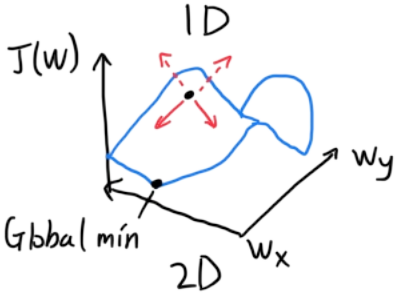
6) Gradient Descent

Gradient Descent

- Process: Initialize random guess of min, W
  - ↳ In each epoch (training iteration):
    W := W * learning rate * (-) its derivative
  - ↳ Stop until W reaches global min



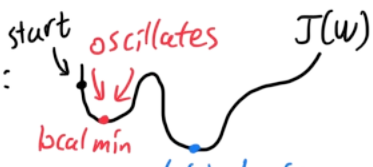


Process is the same except: ① W is now $[W_x, W_y]$
② Update W by gradient ∇ (a list of partial derivatives)

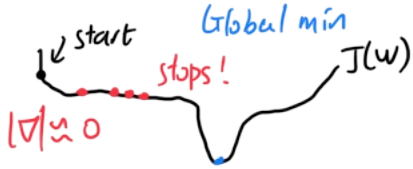
High-dim tradeoff: Harder to visualize but less local mins

- Cons:

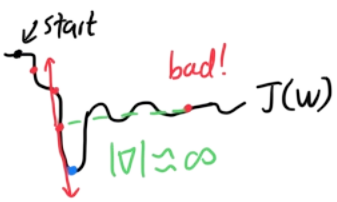
① Get stuck in Local min:



② Vanishing Gradients:



③ Exploding Gradients:



Address it by:
① Pick different start
② Increase dimensionality

---

① Reduce hidden layers of model
② Use non-saturating activation functions
③ Normalize weights
④ Pre-train model with autoencoders
⑤ Regularization ⑥ ResNet

Dynamic Learning Rates

① Gradient-based: In each epoch i, lr := lr * |gradient|

- Intrepretation: lr is larger the further ur away from final solution; smaller otherwise
  May requrie additional param for appropriate scaling

② Time-based: In each epoch i, $lr := lr * (1 - \frac{i+1}{\Sigma i})$

- Interpretation: $\lim_{i \to \infty} lr = 0$, learning effectiveness reduces as training progresses