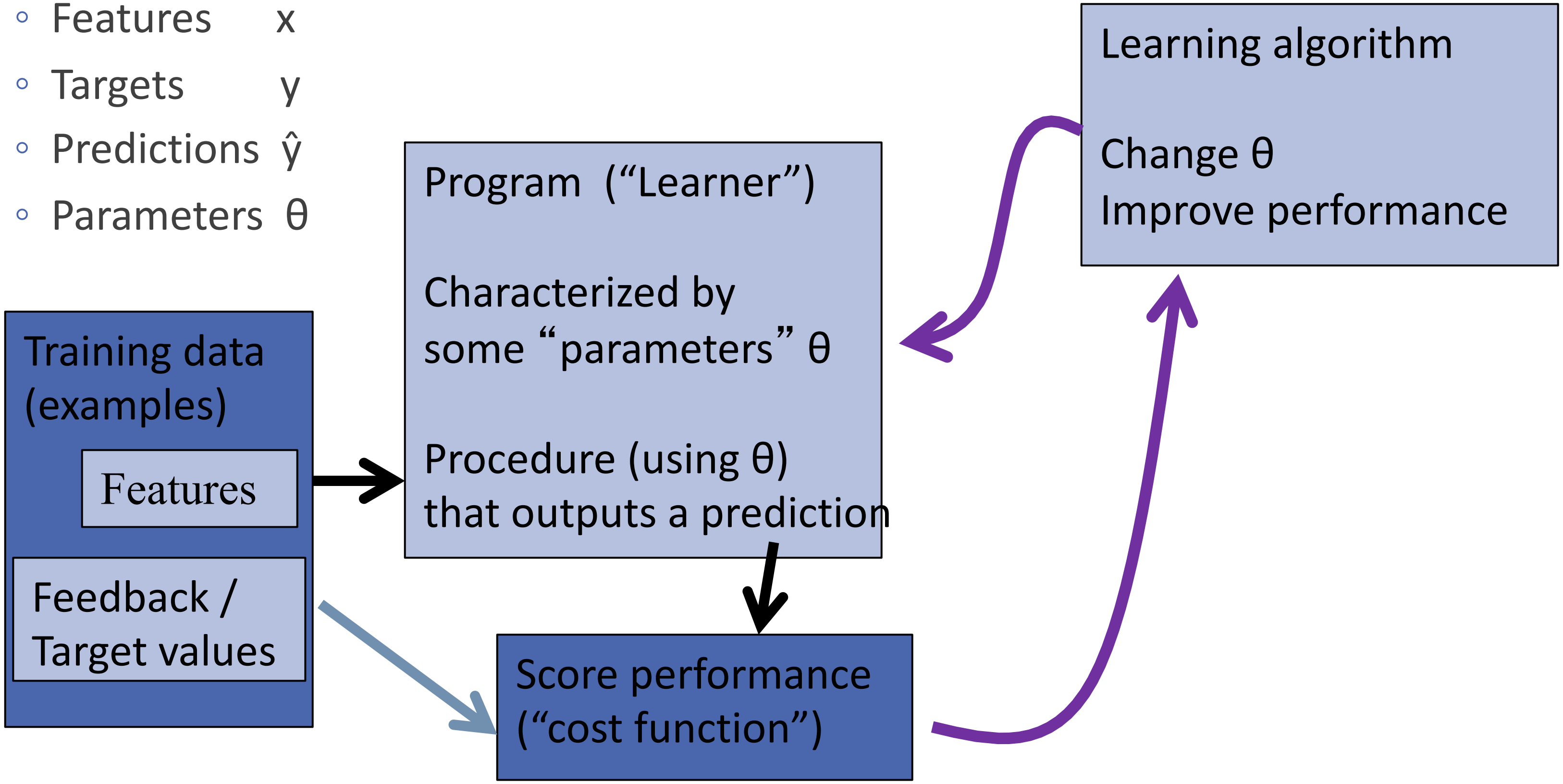
Supervised learning

Notation

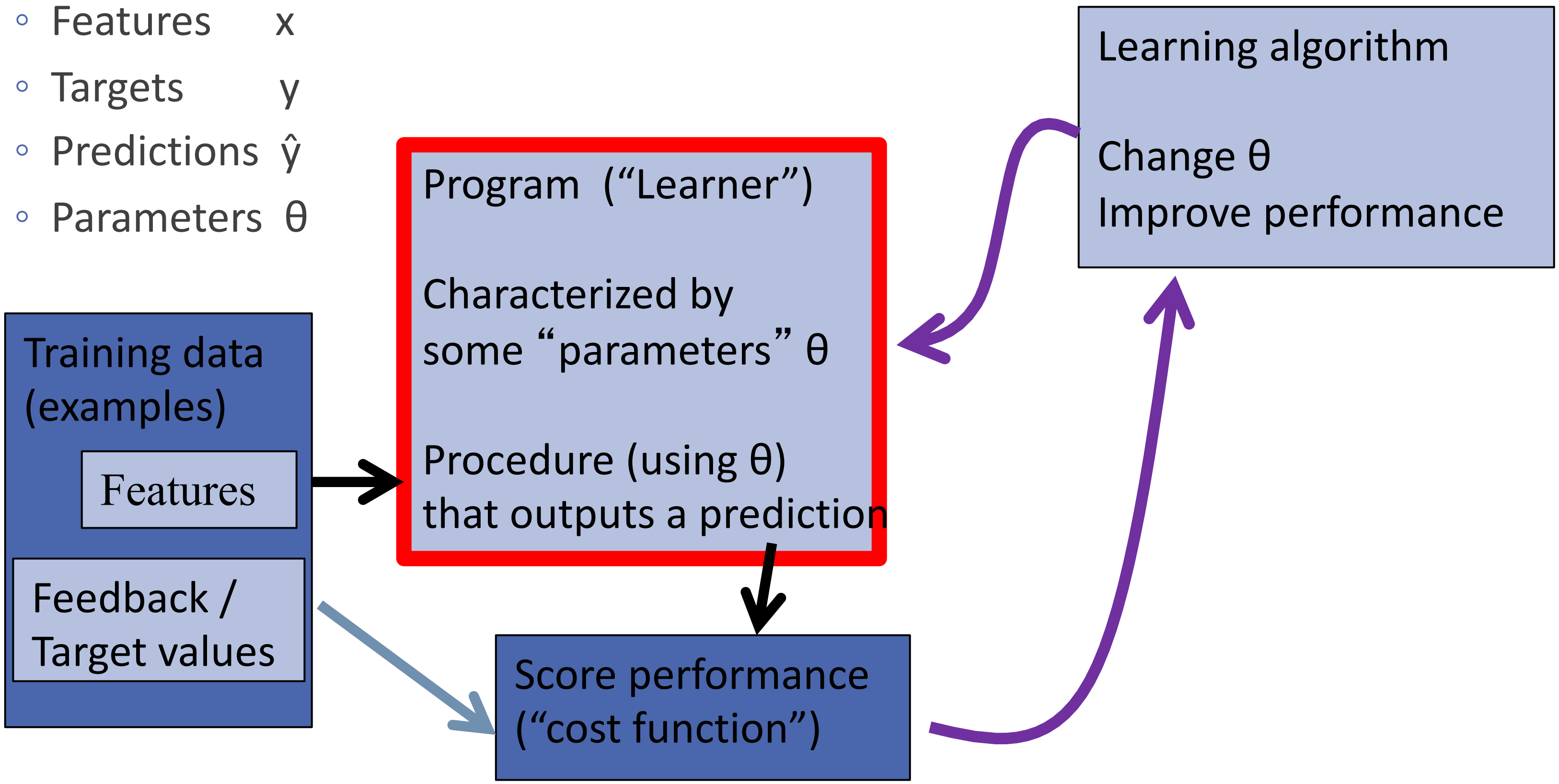
- Features x
- Targets y
- Predictions ŷ
- Parameters θ



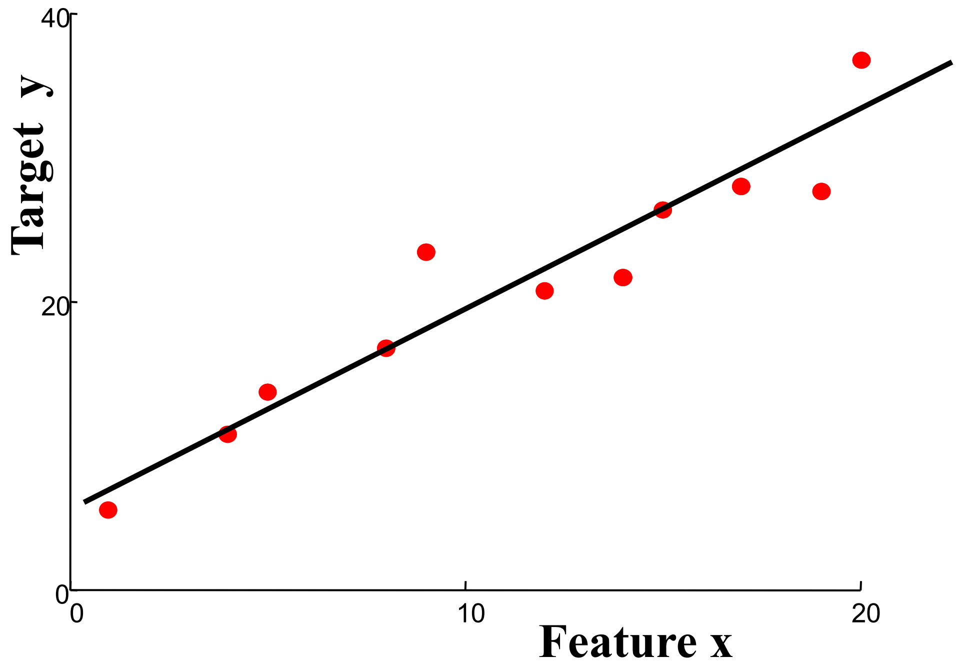
Supervised learning

Notation

- Features x
- Targets y
- Predictions ŷ
- Parameters θ



Linear regression



Define form of function f(x) explicitly

Find a good f(x) within that family

"Predictor":
Evaluate line:

$$r = \theta_0 + \theta_1 x_1$$

return r

Notation

$$\hat{y}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots$$

Define "feature" $x_0 = 1$ (constant)

Then

$$\hat{y}(x) = \theta\, x^T$$

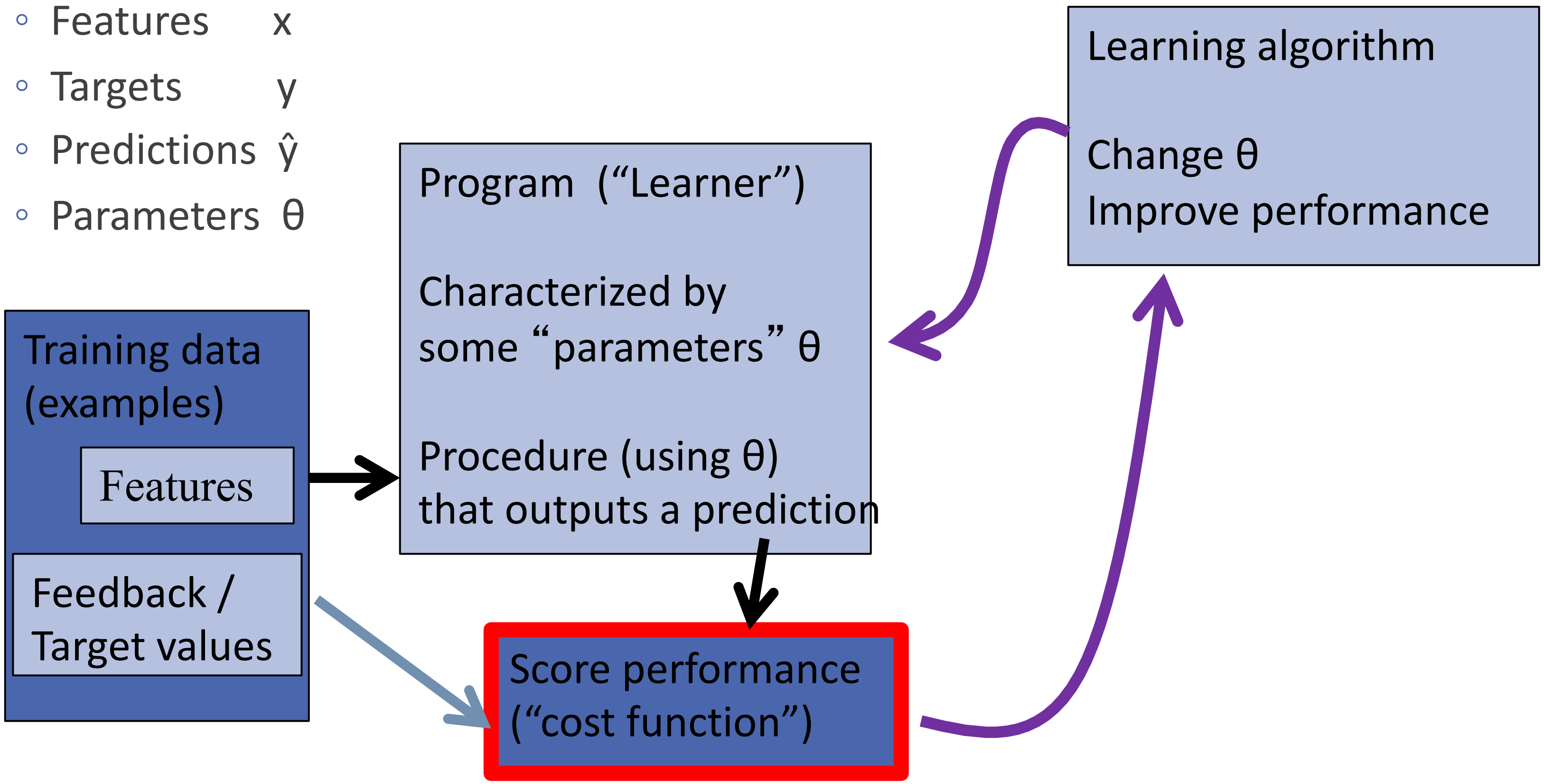
$$\underline{\theta} = [\theta_0, \ldots, \theta_n]$$

$$\underline{x} = [1, x_1, \ldots, x_n]$$

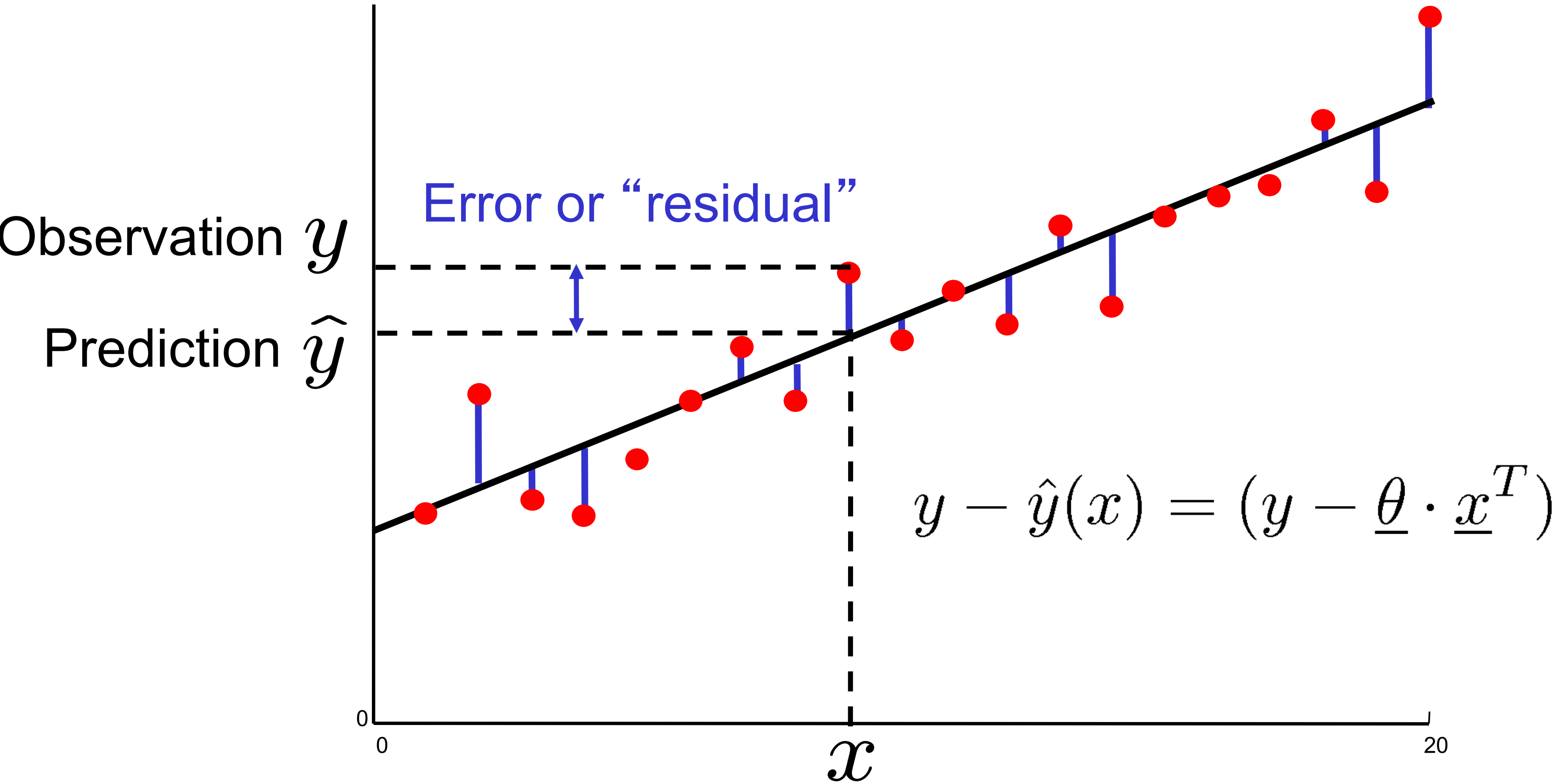
Supervised learning

Notation

- Features x
- Targets y
- Predictions ŷ
- Parameters θ



Measuring error



Mean squared error

How can we quantify the error?

$$\text{MSE, } J(\underline{\theta}) = \frac{1}{m} \sum_j (y^{(j)} - \hat{y}(x^{(j)}))^2$$

$$= \frac{1}{m} \sum_j (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)T})^2$$

Why choosing exactly this error measure and not something else?

- Computationally convenient (more later)
- Measures the variance of the residuals
- Corresponds to likelihood under Gaussian model of "noise"

$$N(y; \hat{y}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{1}{2\sigma^2}(y - \hat{y})^2\}$$

MSE Cost function

$$\text{MSE, } J(\underline{\theta}) = \frac{1}{m} \sum_j (y^{(j)} - \hat{y}(x^{(j)}))^2$$

$$= \frac{1}{m} \sum_j (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)T})^2$$

$$\underline{\theta} = [\theta_0, \ldots, \theta_n]$$

$$\underline{y} = \left[y^{(1)} \ldots, y^{(m)} \right]^T$$

$$\underline{X} = \begin{bmatrix} x_0^{(1)} & \ldots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & \ldots & x_n^{(m)} \end{bmatrix}$$

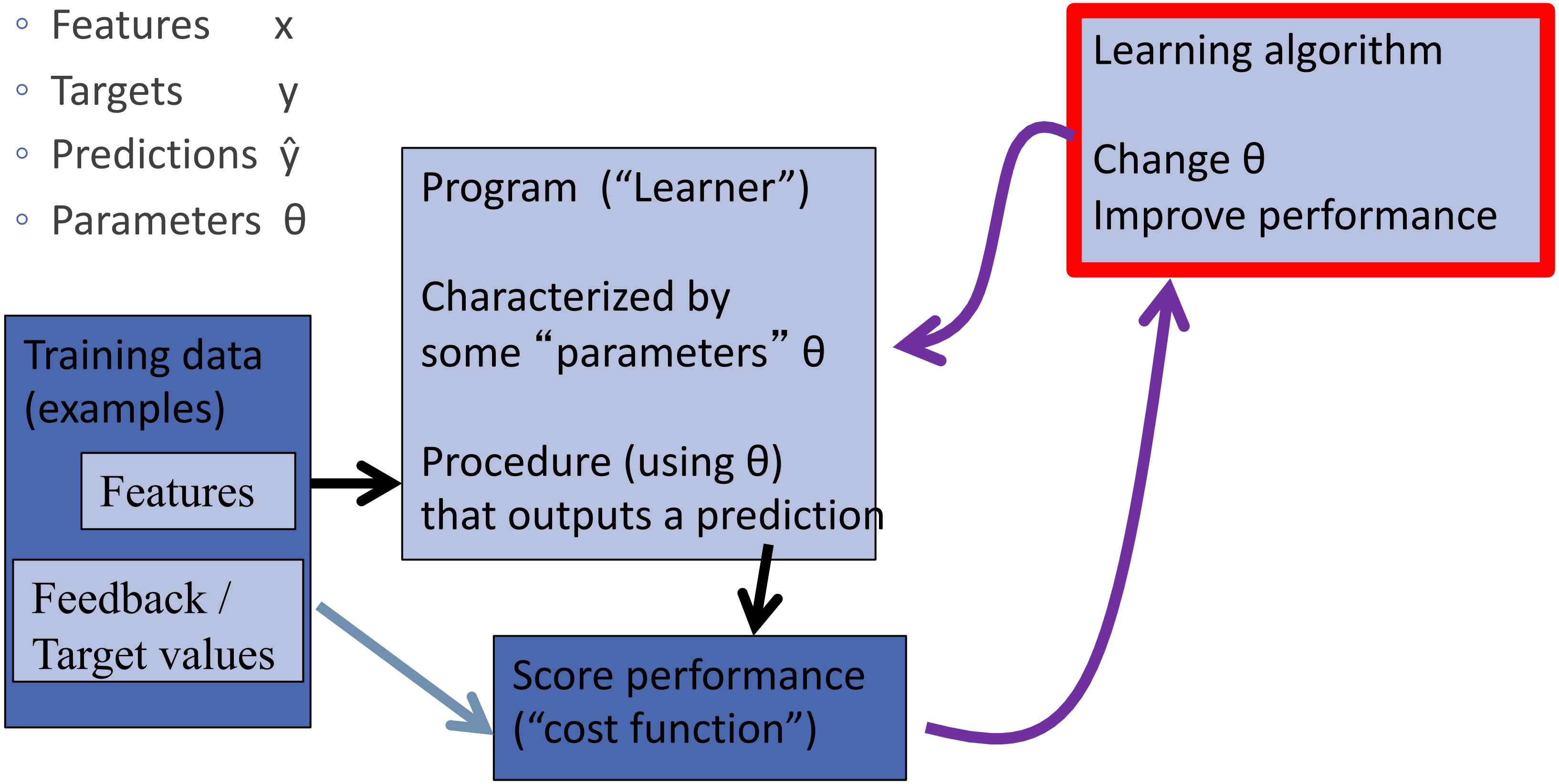
$$J(\underline{\theta}) = \frac{1}{m} (\underline{y}^T - \underline{\theta}\, \underline{X}^T) \cdot (\underline{y}^T - \underline{\theta}\, \underline{X}^T)^T$$

```
# Python / NumPy:
e = Y - X.dot( theta.T );
J = e.T.dot( e ) / m  # = np.mean( e ** 2 )
```

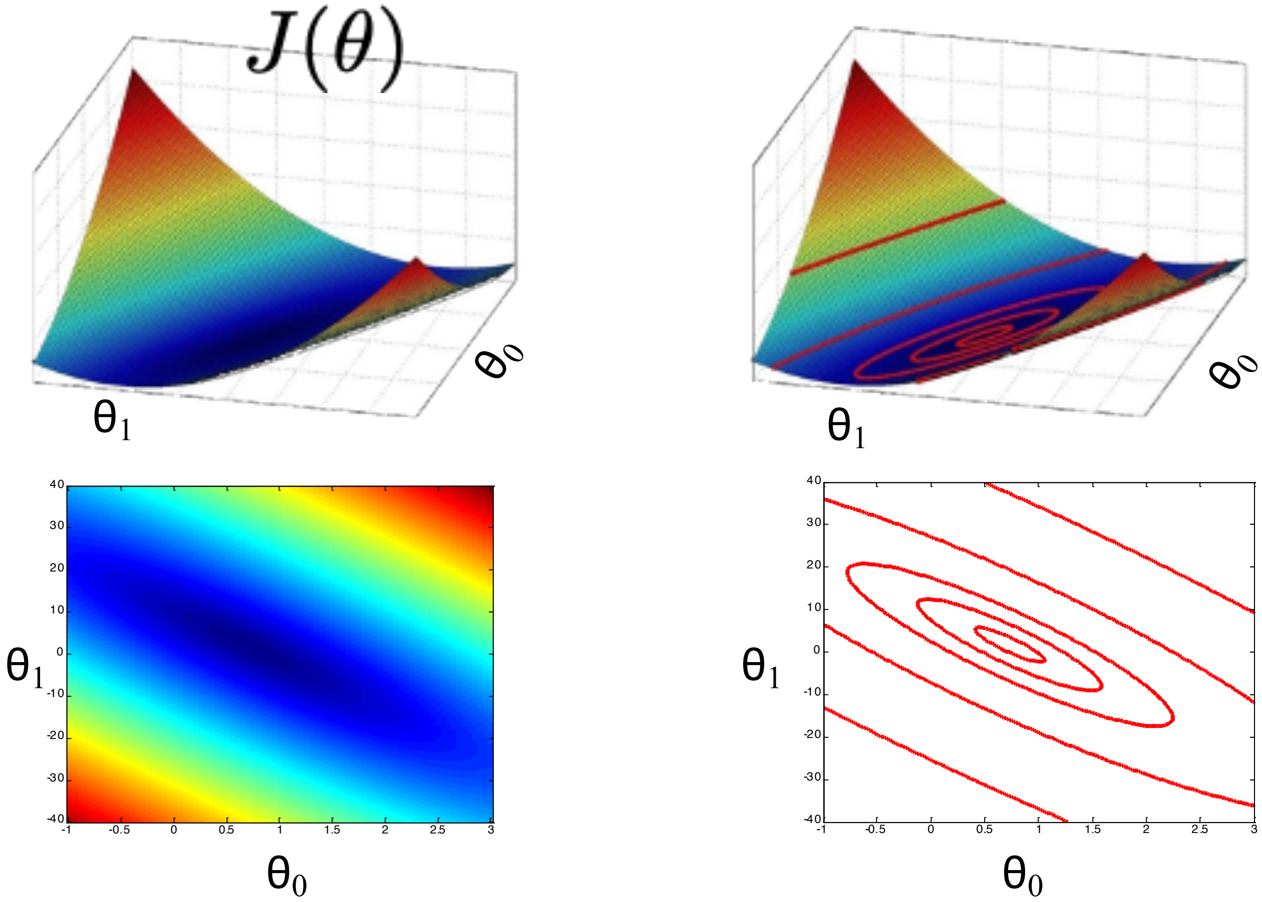
Supervised learning

Notation

- Features x
- Targets y
- Predictions $\hat{y}$
- Parameters θ



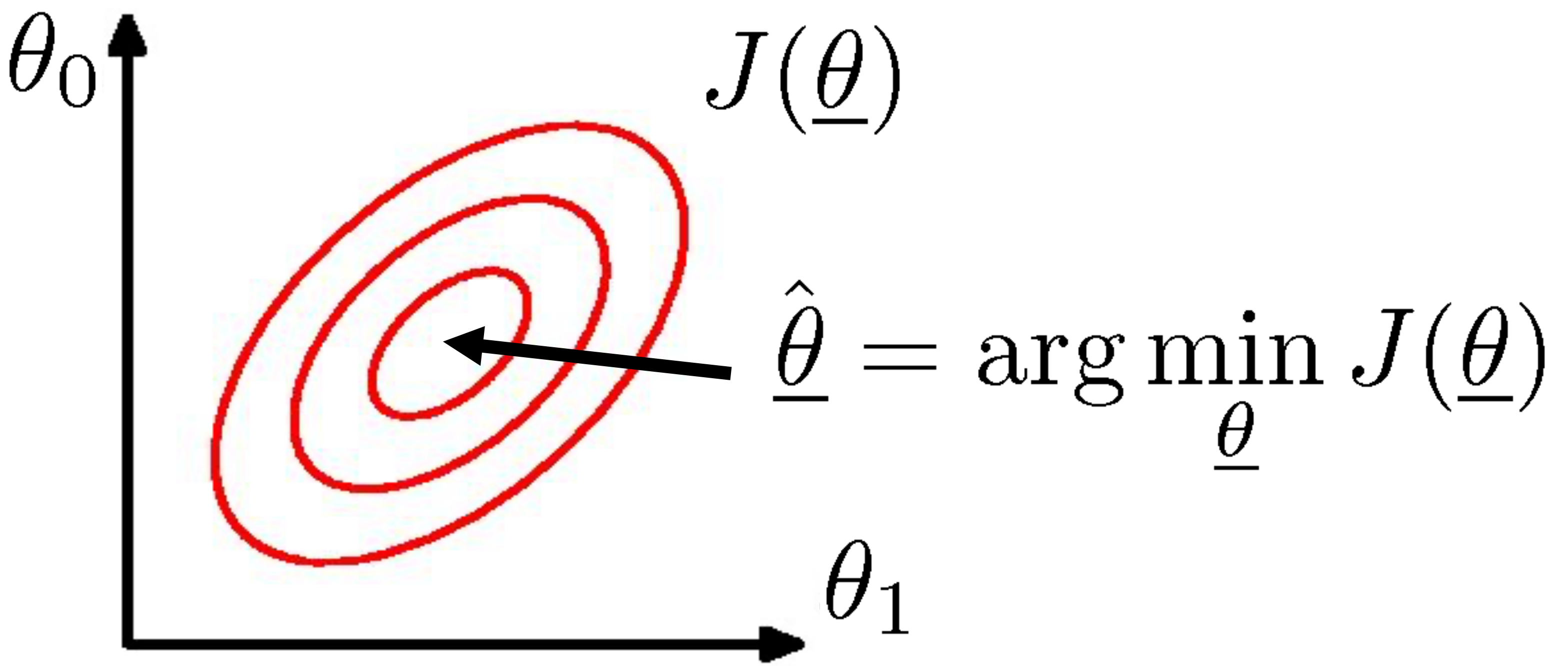
Visualizing the cost function



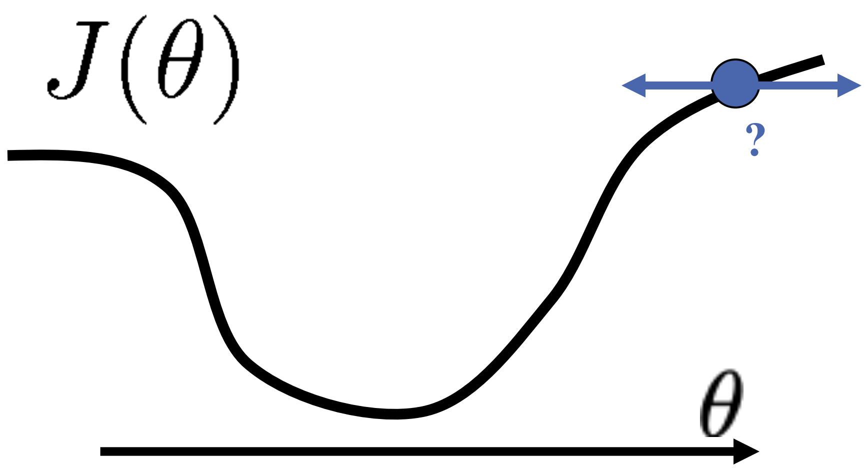
Finding good parameters

Want to find parameters which minimize our error…

Think of a cost “surface”: error residual for that θ…

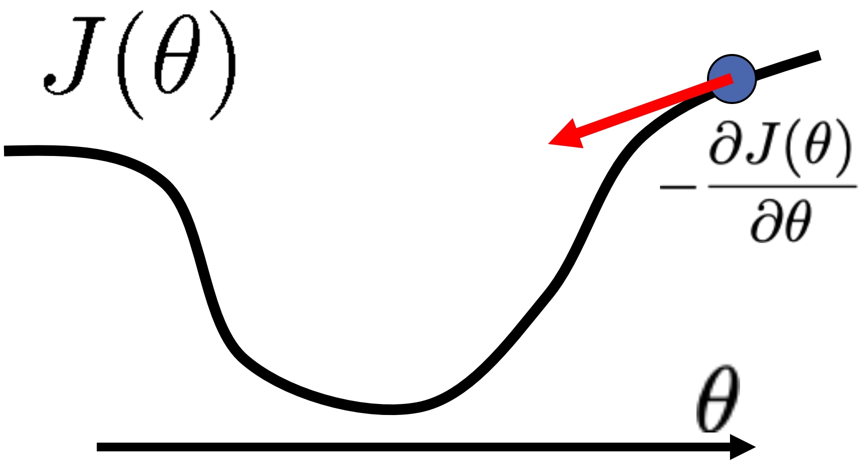


Gradient descent



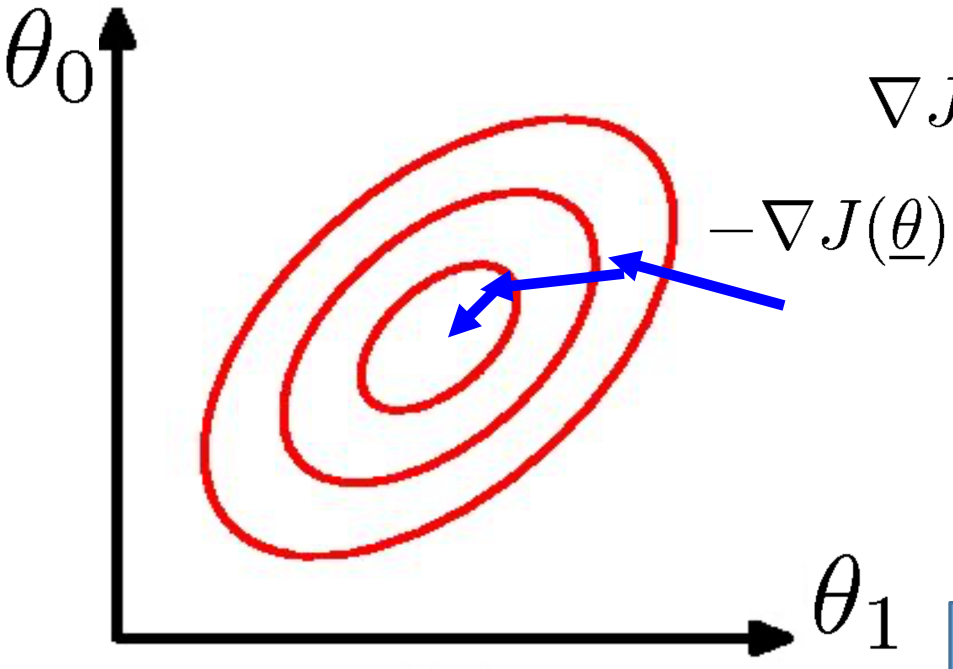
- How to change θ to improve J(θ)?
- Choose a direction in which J(θ) is decreasing

Gradient descent



- How to change θ to improve J(θ)?
- Choose a direction in which J(θ) is decreasing
- Derivative $\frac{\partial J(\theta)}{\partial \theta}$
- Positive => increasing
- Negative => decreasing

Gradient descent in >2 dimensions



- Gradient vector

$$\nabla J(\underline{\theta}) = \left[\frac{\partial J(\underline{\theta})}{\partial \theta_0} \quad \frac{\partial J(\underline{\theta})}{\partial \theta_1} \quad \ldots \right]$$

Indicates direction of steepest ascent
(negative = steepest descent)

Gradient descent

Initialization

Step size α

- Can change as a function of iteration

Gradient direction

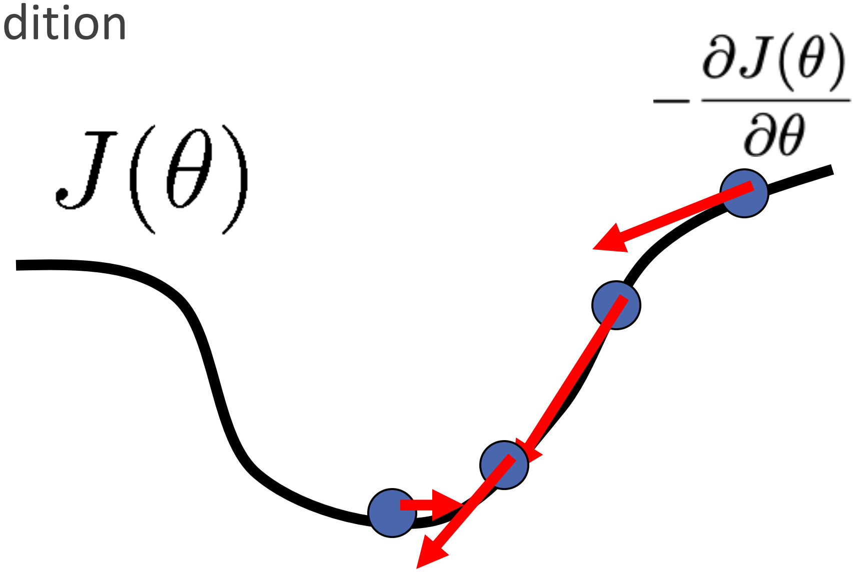
Stopping condition

Initialize θ

Do{

$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$

} while $(\alpha \|\nabla_\theta J\| > \varepsilon)$



Gradient for the MSE

MSE
$$J(\underline{\theta}) = \frac{1}{m}\sum_j (y^{(j)} - \underline{\theta}\cdot \underline{x}^{(j)^T})^2$$

$\nabla J = ?$
$$J(\underline{\theta}) = \frac{1}{m}\sum_j (\overbrace{y^{(j)} - \theta_0 \underline{x}_0^{(j)} - \theta_1 \underline{x}_1^{(j)} - \ldots}^{e_j(\theta)})^2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial}{\partial \theta_0}\frac{1}{m}\sum_j (\ e_j(\theta)\ )^2$$
$$= \frac{1}{m}\sum_j \quad \frac{\partial}{\partial \theta_0}(\ e_j(\theta)\ )^2$$
$$= \frac{1}{m}\sum_j \ 2e_j(\theta)\ \frac{\partial}{\partial \theta_0} e_j(\theta)$$

$$\frac{\partial}{\partial \theta_0} e_j(\theta) = \underbrace{\frac{\partial}{\partial \theta_0} y^{(j)}}_{0} - \frac{\partial}{\partial \theta_0}\theta_0 x_0^{(j)} - \underbrace{\frac{\partial}{\partial \theta_0}\theta_1 x_1^{(j)}}_{0} - \ldots$$
$$= -x_0^{(j)}$$

Gradient for the MSE

MSE
$$J(\underline{\theta}) = \frac{1}{m} \sum_j (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)^T})^2$$

∇J = ?
$$J(\underline{\theta}) = \frac{1}{m} \sum_j (\overbrace{y^{(j)} - \theta_0 \underline{x}_0^{(j)} - \theta_1 \underline{x}_1^{(j)} - \ldots}^{e_j(\theta)})^2$$

$$\nabla J(\underline{\theta}) = \left[\quad \frac{\partial J}{\partial \theta_0} \qquad \frac{\partial J}{\partial \theta_1} \qquad \ldots \right]$$
$$= \left[\frac{2}{m} \sum_j -e_j(\theta) x_0^{(j)} \quad \frac{2}{m} \sum_j -e_j(\theta) x_1^{(j)} \qquad \ldots \right]$$

Gradient descent

Initialization

Step size
- Can change as a function of iteration

Gradient direction

Stopping condition

```
Initialize θ
Do {
  θ ← θ - α∇θJ(θ)
} while (α||∇θJ|| > ε)
```

$$J(\underline{\theta}) = \frac{1}{m} \sum_j (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)T})^2$$

$$\nabla J(\underline{\theta}) = -\frac{2}{m} \sum_j \underbrace{(y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)T})}_{\text{Error magnitude \& direction for datum j}} \cdot \underbrace{[x_0^{(j)} x_1^{(j)} \ldots]}_{\text{Sensitivity to each } \theta_i}$$

Derivative of MSE

Rewrite using matrix form

$$\nabla J(\underline{\theta}) = -\frac{2}{m}\sum_j \underbrace{\left(y^{(j)} - \underline{\theta}\cdot \underline{x}^{(j)\,T}\right)}_{\text{Error magnitude \& direction for datum j}} \cdot \underbrace{\left[x_0^{(j)}\, x_1^{(j)} \ldots\right]}_{\text{Sensitivity to each } \theta_i}$$

$$\underline{\theta} = [\theta_0, \ldots, \theta_n]$$

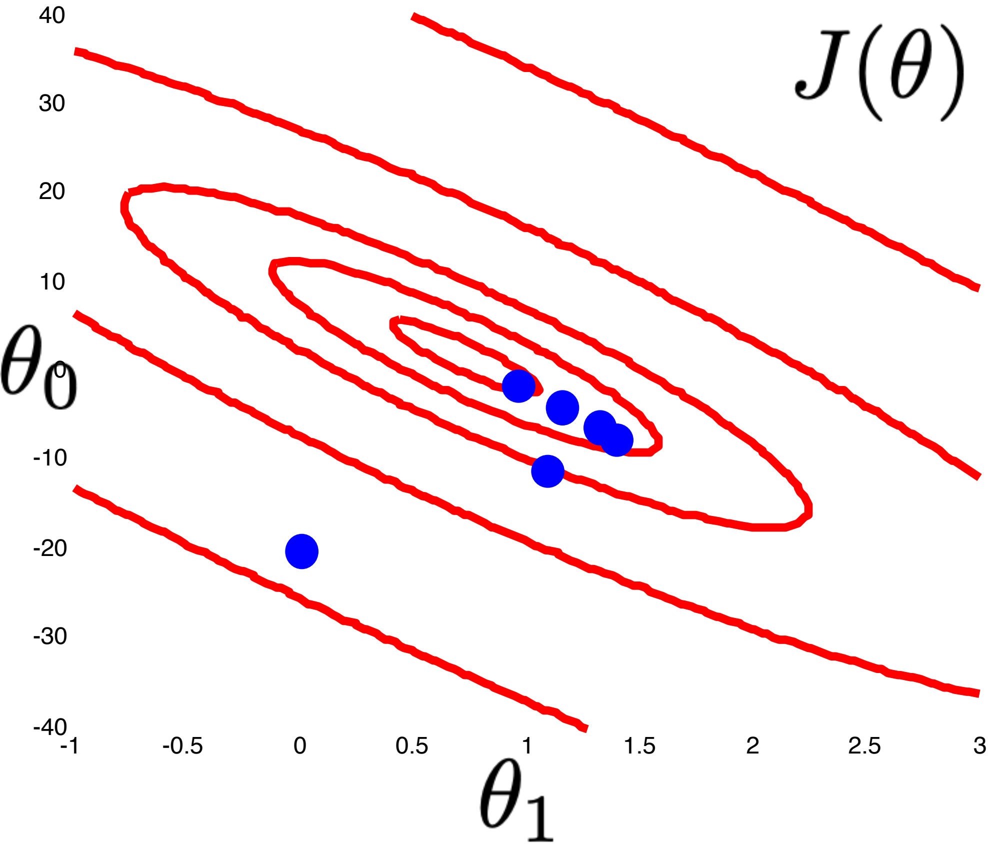
$$\underline{y} = \left[y^{(1)} \ldots, y^{(m)}\right]^T$$

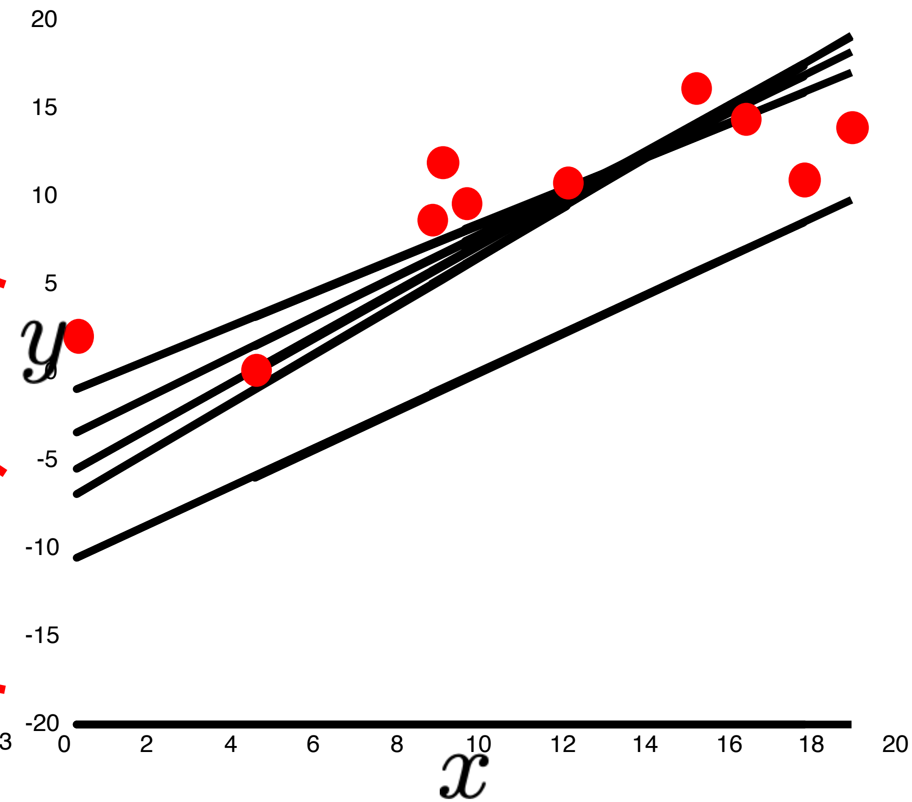
$$\underline{X} = \begin{bmatrix} x_0^{(1)} & \ldots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & \ldots & x_n^{(m)} \end{bmatrix}$$

$$\nabla J(\underline{\theta}) = -\frac{2}{m}(\underline{y}^T - \underline{\theta X}^T)\cdot \underline{X}$$

```
e = Y - X.dot( theta.T ); # error residual
DJ = - e.dot(X) * 2.0/m   # compute the gradient
theta -= alpha * DJ       # take a step
```

Gradient descent on cost function





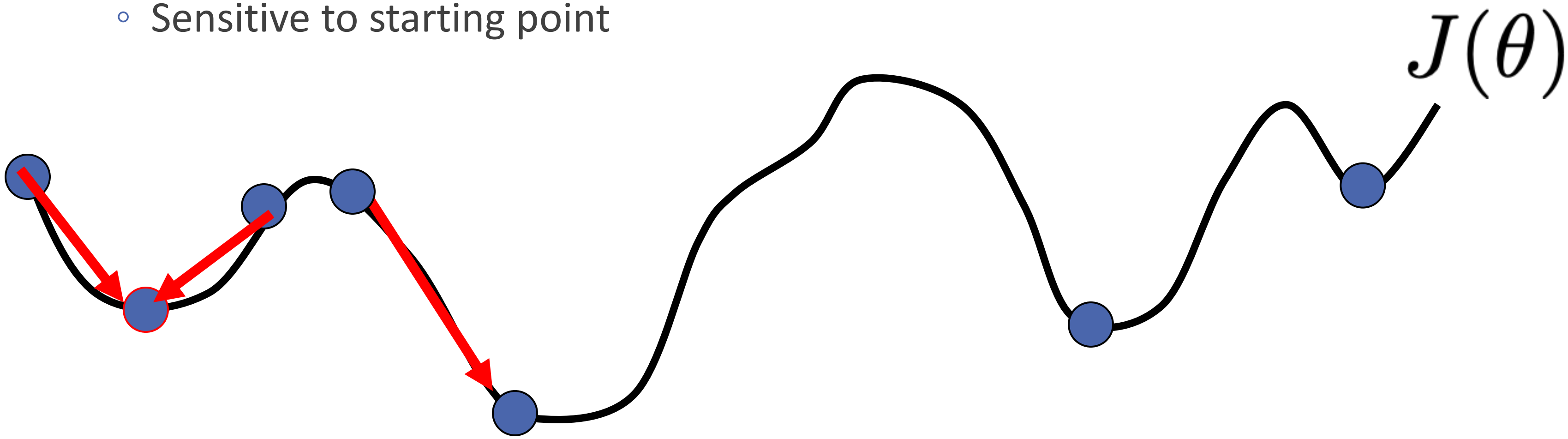
Comments on Gradient Descent

Very general algorithm

- We'll see it many times

Local minima

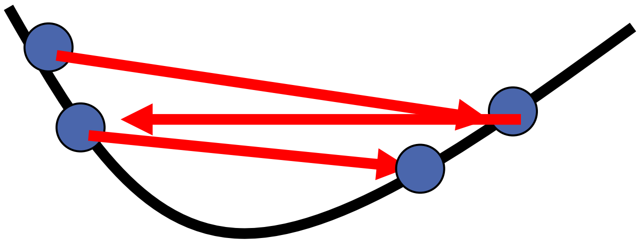
- Sensitive to starting point



Comments on Gradient Descent

Very general algorithm

- We'll see it many times



Local minima

- Sensitive to starting point

Step size

- Too large? Too small? Automatic ways to choose?
- May want step size to decrease with iteration
- Common choices:
  - Fixed
  - Linear: C/(iteration)
  - Line search
  - Newton's method