Lecture 6

Gradient Descent

Last Time:

- Machine learning, especially supervised learning
- Bias, variance, and overfitting
- Minimized an objective function, called error or cost or risk

LLN: Expectations -> sample averages

$$E_f[R] = \int R(x) f(x) dx = \lim_{n \to \infty} \frac{1}{N} \sum_{x_i \sim f} R(x_i)$$

Empirical Risk Minimization:

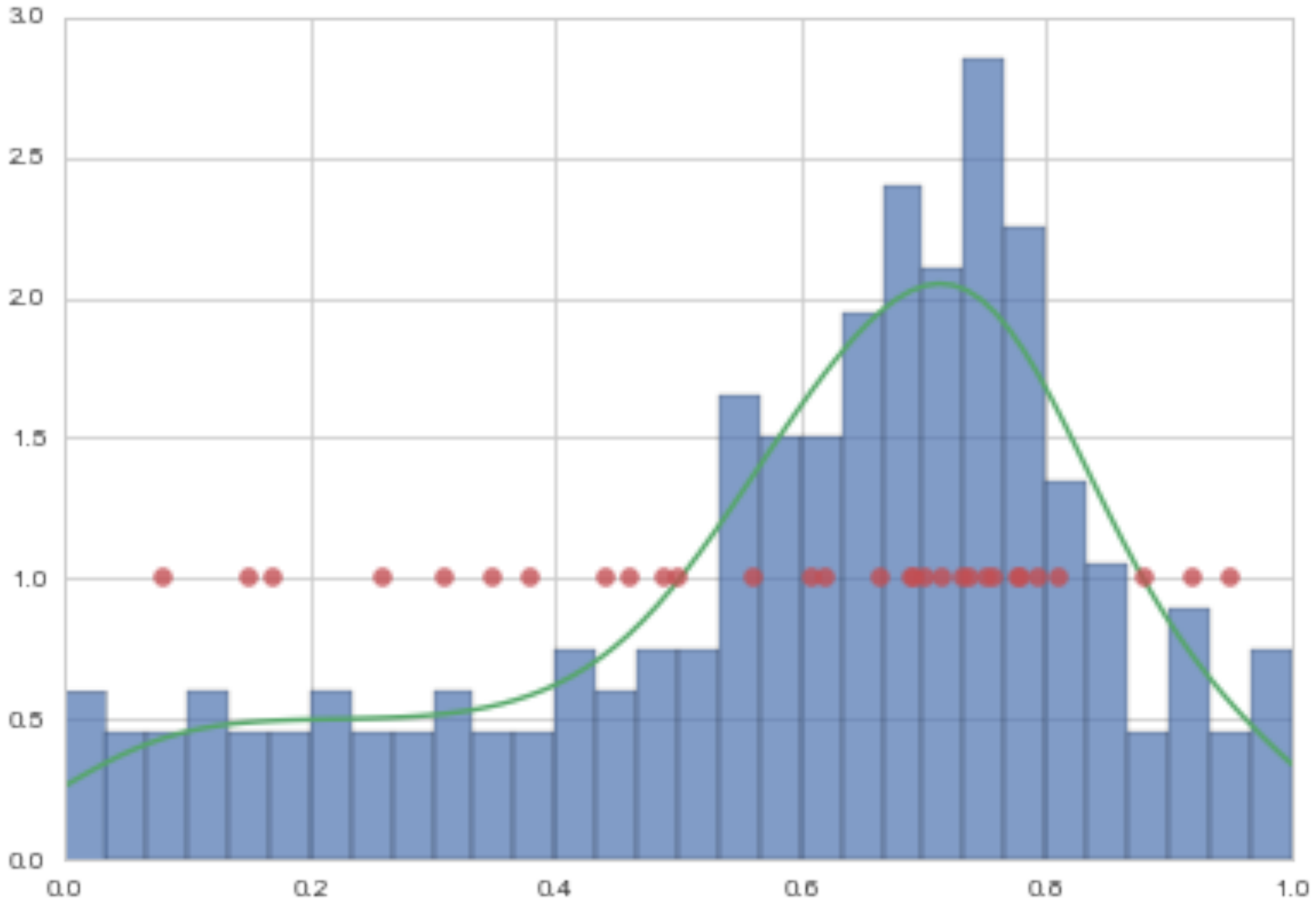
$$R_{\mathcal{D}} = E_f[R] \sim \frac{1}{N} \sum_{x_i \sim f} R(x_i)$$

on training set(sample) $\mathcal{D}$.

Today: machine learning (contd), optimization using gradient descent

- overfitting. complexity, and test sets
- gradient descent
- stochastic gradient descent

Remember Convex (bowl) like functions have 1 global minimum



Statement of the Learning Problem

The sample must be representative of the population!

$$A : R_{\mathcal{D}}(g)\ \ smallest\, on\, \mathcal{H}$$
$$B : R_{out}(g) \approx R_{\mathcal{D}}(g)$$

A: Empirical risk estimates in-sample risk.
B: Thus the out of sample risk is also small.

$$R_{out}(h) = E_{p(x)}[(h(x) - y)^2] = \int dx p(x)(h(x) - f(x) - \epsilon)^2.$$

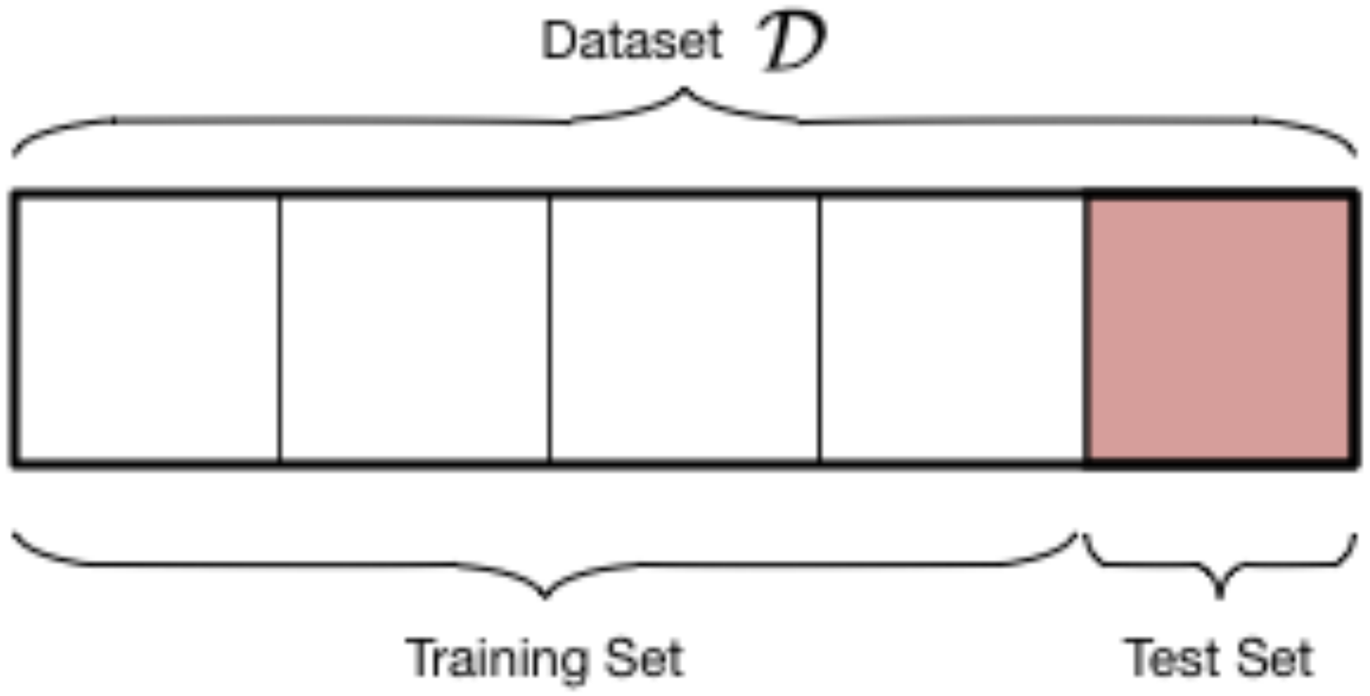
Fit hypothesis $h = g_{\mathcal{D}}$, where $\mathcal{D}$ is our training sample.

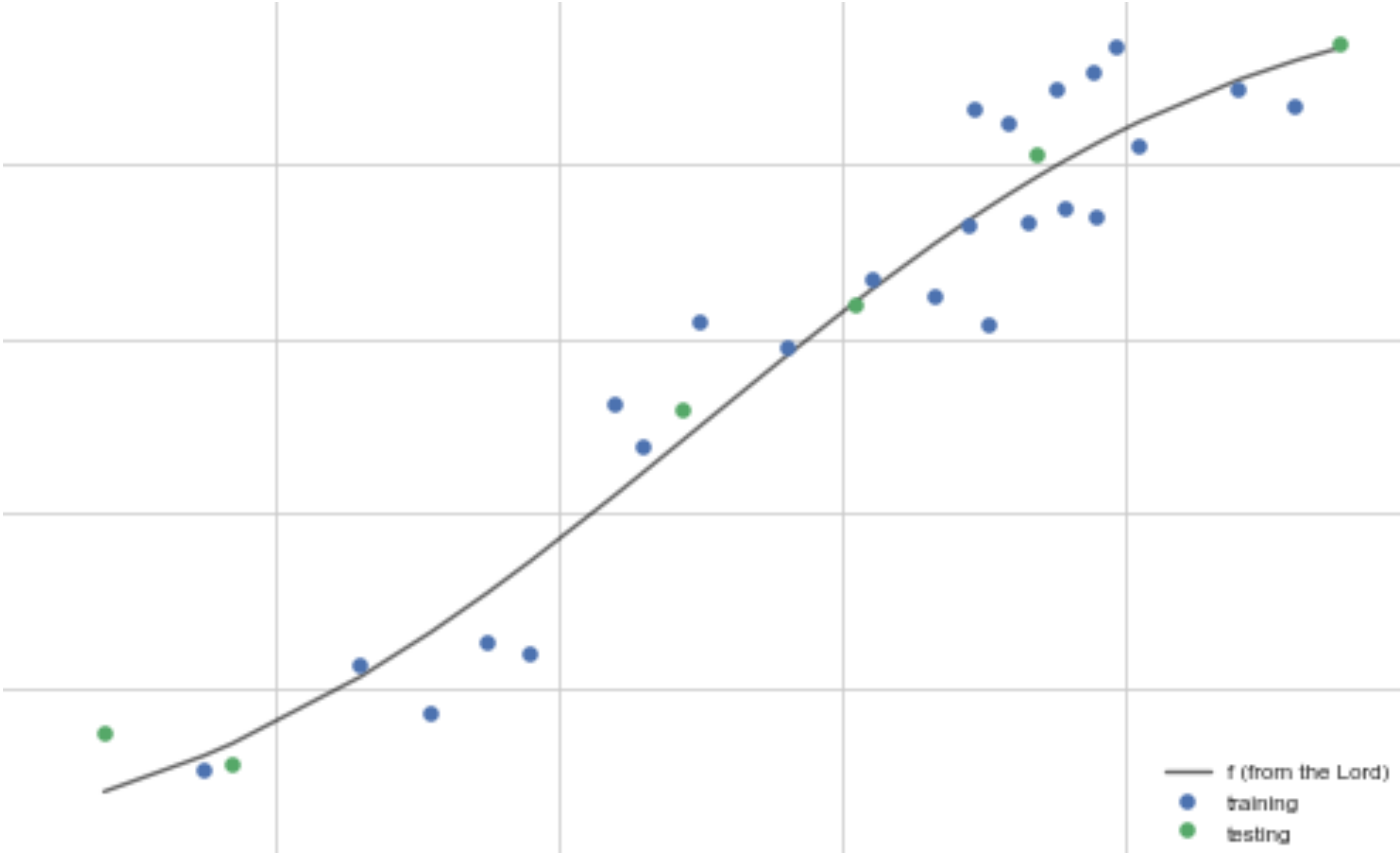
Define:

$$\langle R \rangle = \int dy dx \, p(x, y)(h(x) - y)^2 = \int dy dx p(y \mid x) p(x)(h(x) - y)^2.$$

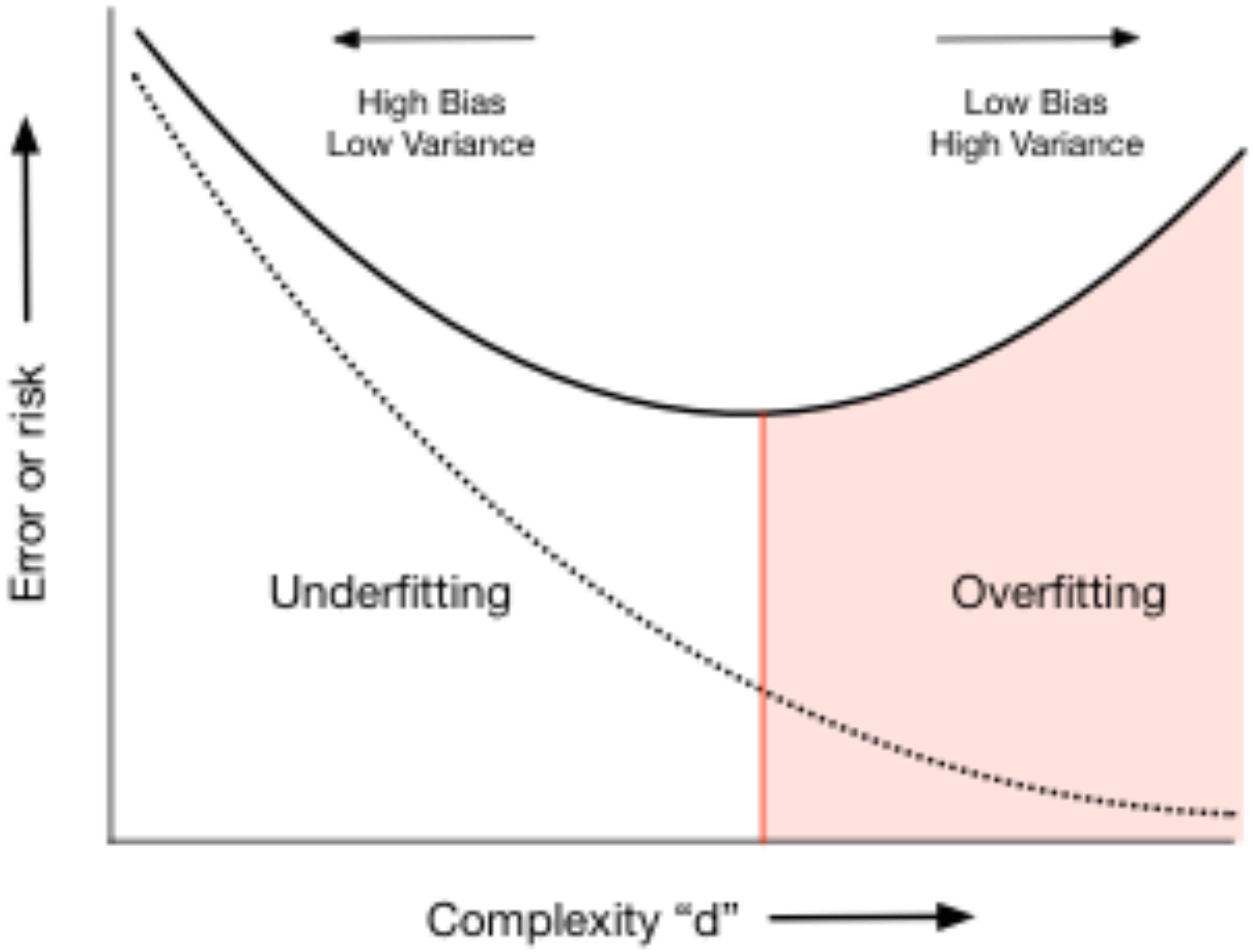
TODO: What is Empirical Risk Minimization?

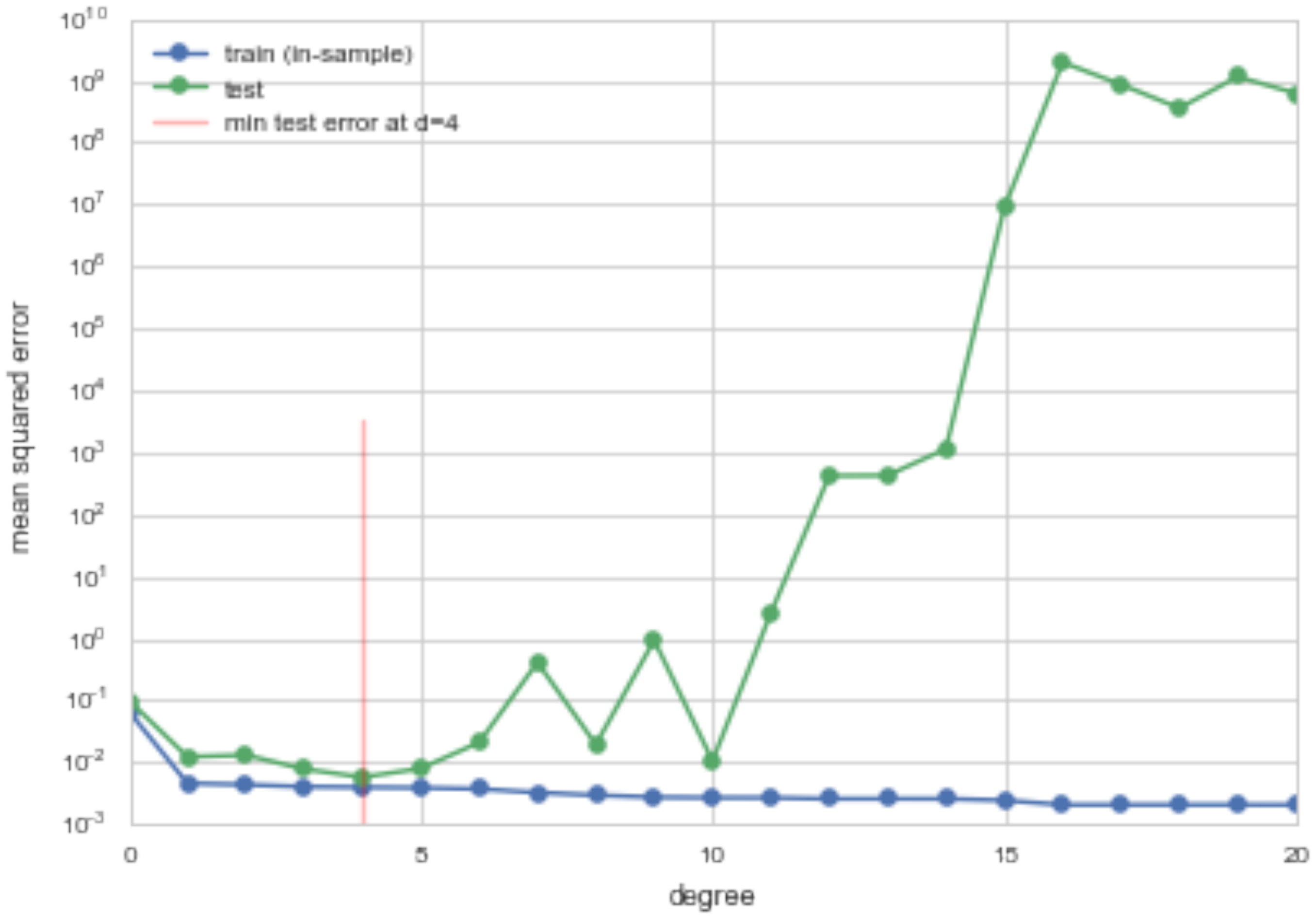
TRAIN AND TEST





BALANCE THE COMPLEXITY





Is this still a test set?

Trouble:

- no discussion on the error bars on our error estimates
- "visually fitting" a value of $d \implies$ contaminated test set.

The moment we **use it in the learning process, it is not a test set**.

Hoeffding's inequality

population fraction μ, sample drawn with replacement, fraction ν:

$$P(|\nu - \mu| > \epsilon) \leq 2e^{-2\epsilon^2 N}$$

For hypothesis h, identify 1 with $h(x_i) \neq f(x_i)$ at sample x_i. Then μ, ν are population/sample error rates. Then,

$$P(|R_{in}(h) - R_{out}(h)| > \epsilon) \leq 2e^{-2\epsilon^2 N}$$

- Hoeffding inequality holds ONCE we have picked a hypothesis h, as we need it to label the 1 and 0s.
- But over the training set we one by one pick all the models in the hypothesis space
- best fit g is among the h in $\mathcal{H}$, g must be h_1 OR h_2 OR....Say **effectively** M such choices:

$$P(|R_{in}(g) - R_{out}(g)| \geq \epsilon) <= \sum_{h_i \in \mathcal{H}} P(|R_{in}(h_i) - R_{out}(h_i)| \geq \epsilon) <= 2\,M\,e^{-2\epsilon^2 N}$$

Hoeffding, repharased:

Now let $\delta = 2\,M\,e^{-2\epsilon^2 N}$.

Then, **with probability** $1-\delta$:

$$R_{out} <= R_{in} + \sqrt{\frac{1}{2N}ln(\frac{2M}{\delta})}$$

For finite effective hypothesis set size M, $R_{out} \sim R_{in}$ as N larger..

Training vs Test

- training error approximates out-of-sample error slowly
- is test set just another sample like the training sample?
- key observation: test set is looking at only one hypothesis because the fitting is already done on the training set. So $M = 1$ for this sample!

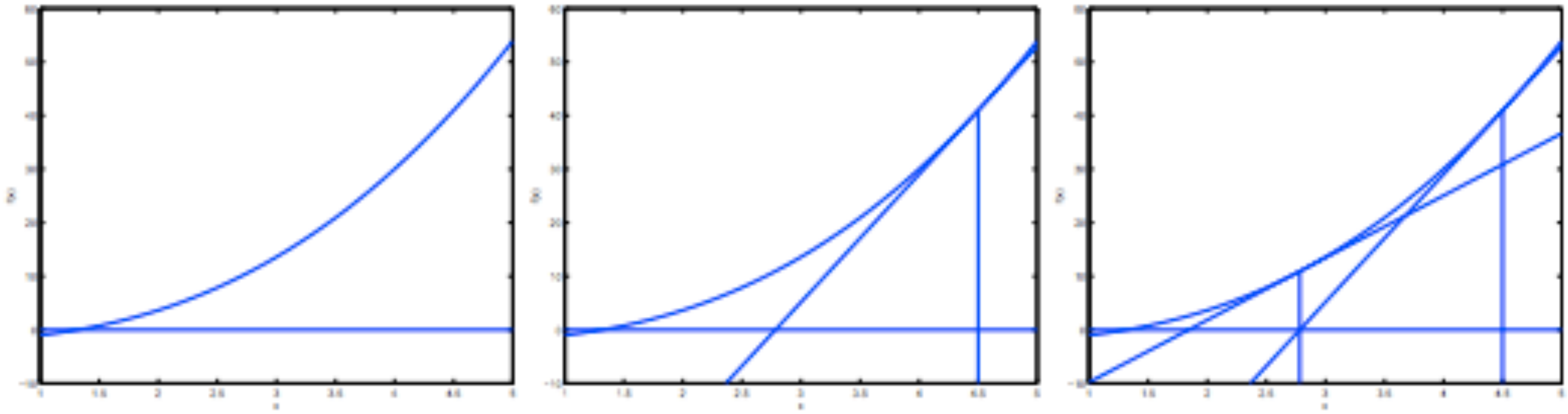
$$R_{out} <= R_{in} + \sqrt{\frac{1}{2N_{test}} ln(\frac{2}{\delta})}$$

Training vs Test

- the test set does not have an optimistic bias like the training set(thats why the larger effective M factor)
- once you start fitting for things like d on the test set, you cant call it a test set any more since we lose tight guarantee.
- test set has a cost of less data in the training set and must thus fit a less complex model.

FINDING DERIVATIVES

Newton's Method



Find a zero of the first derivative.

Gradients and Hessians

$$J(\bar{\theta}) = \theta_1^2 + \theta_2^2$$

Gradient: $\nabla_{\bar{\theta}}(J) = \dfrac{\partial J}{\partial \bar{\theta}} = \begin{pmatrix} 2\theta_1 \\ 2\theta_2 \end{pmatrix}$

Hessian H = $\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$

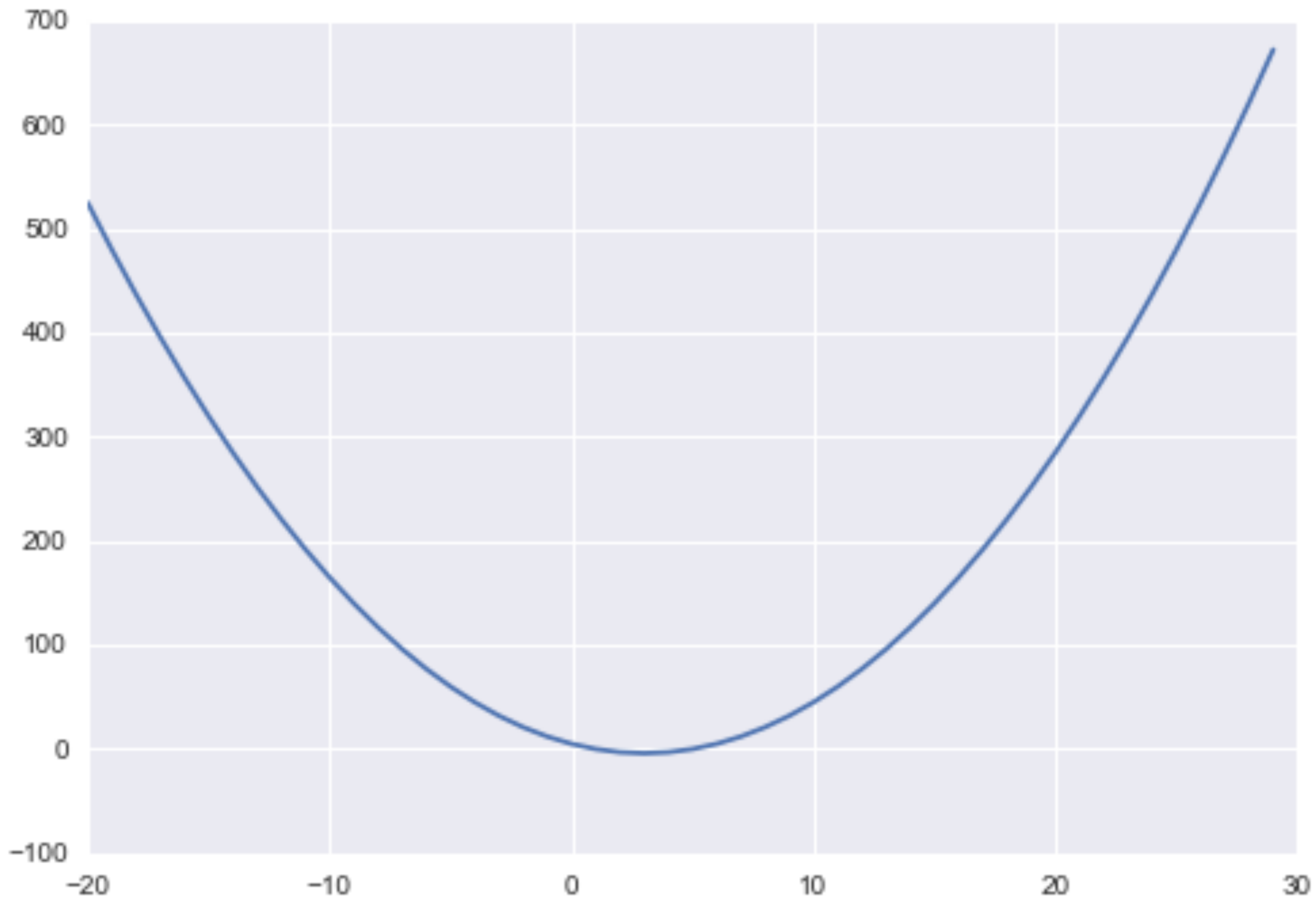
Hessian gives curvature. Why not use it?

Gradient ascent (descent)

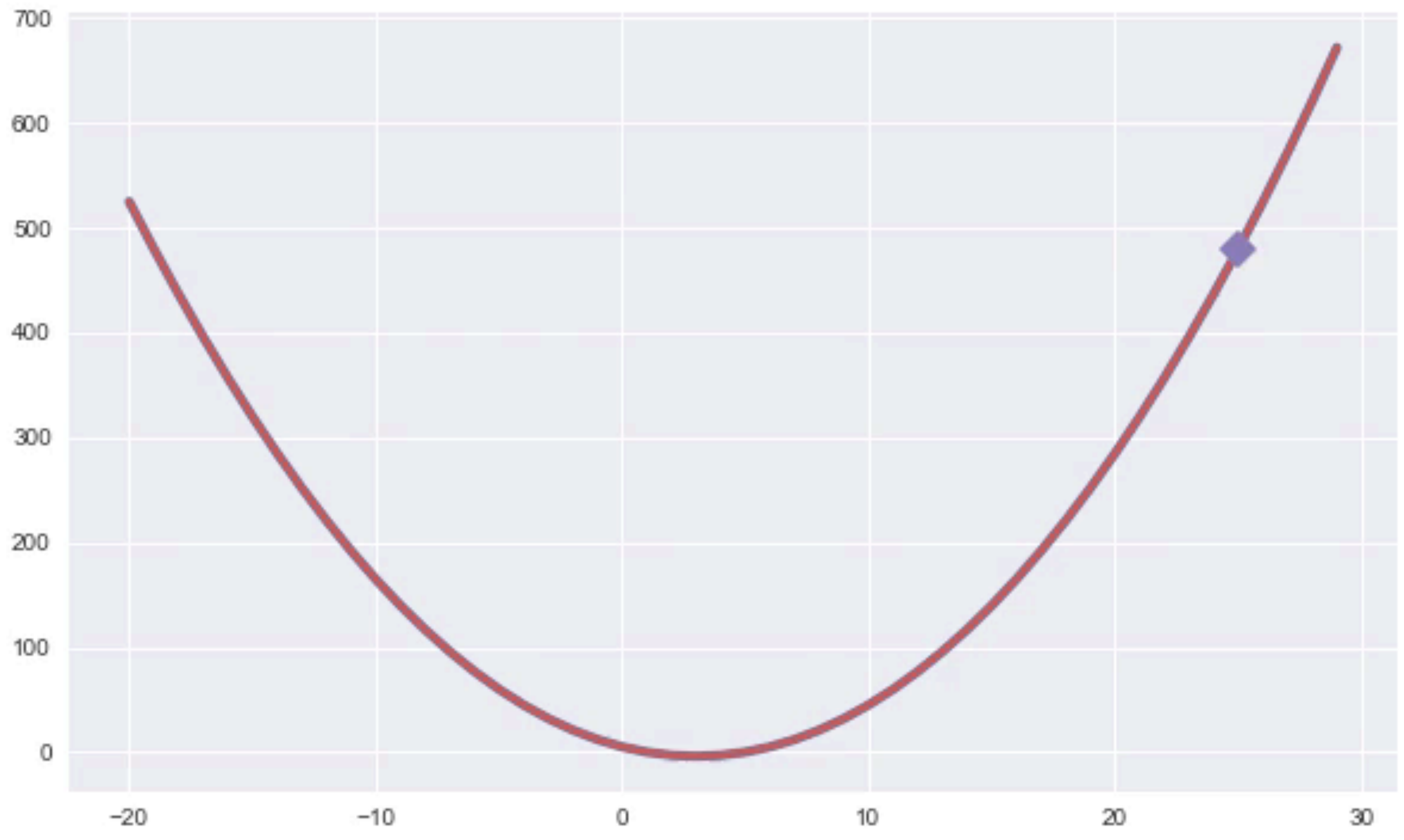
basically go opposite the direction of the derivative.

Consider the objective function:
$J(x) = x^2 - 6x + 5$

```
gradient = fprime(old_x)
move = gradient * step
current_x = old_x - move
```



good step size



too big step size

too small step size

Example: Linear Regression

$$\hat(y) = f_\theta(x) = \theta^T x$$

Cost Function:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (f_\theta(x^{(i)} - y^{(i)})^2$$

Gradient Descent

$$\theta := \theta - \eta \nabla_{\theta} J(\theta) = \theta - \eta \sum_{i=1}^{m} \nabla J_i(\theta)$$

where η is the learning rate.

ENTIRE DATASET NEEDED

```
for i in range(n_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad`
```

Linear Regression: Gradient Descent

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - f_\theta(x^{(i)}))x_j^{(i)}$$

Stochastic Gradient Descent

$$\theta := \theta - \alpha \nabla_\theta J_i(\theta)$$

ONE POINT AT A TIME

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

Mini-Batch: do some at a time

Linear Regression: SGD

$$\theta_j := \theta_j + \alpha(y^{(i)} - f_\theta(x^{(i)}))x_j^{(i)}$$