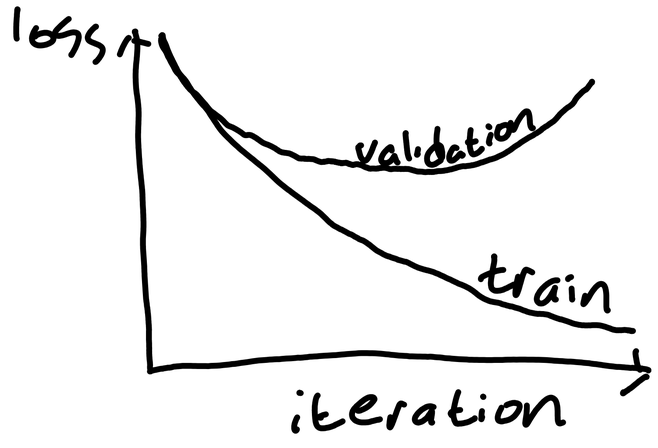
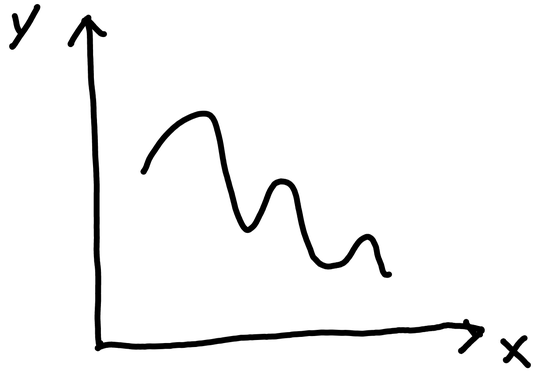
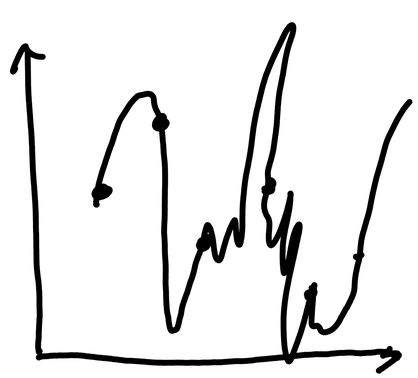
Overfitting and Regularization



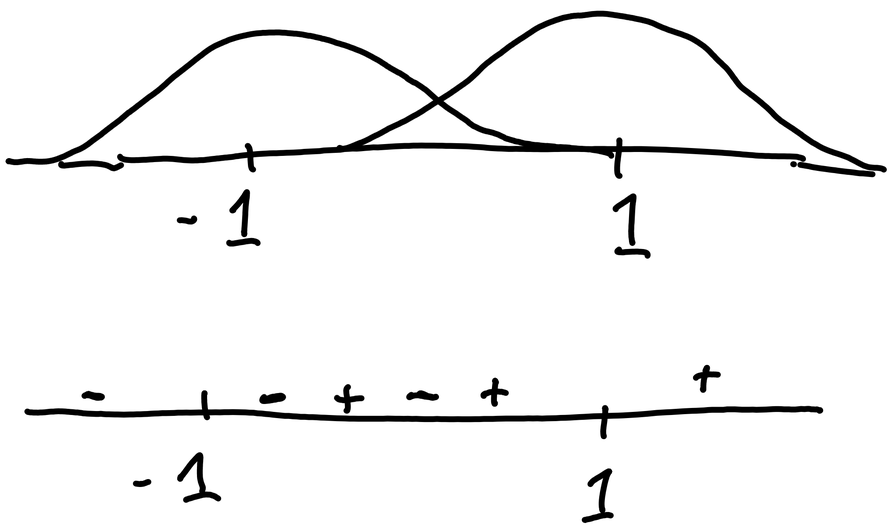


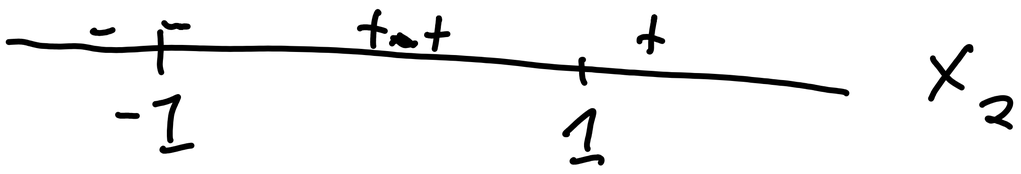


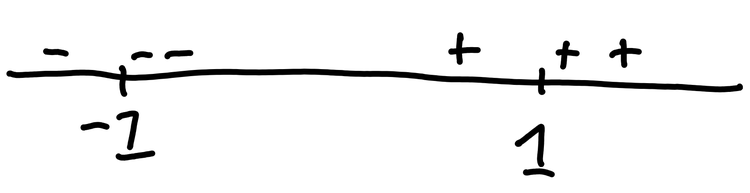
How can we overfit in a linear model?

Say $x_i \sim N(y, \sigma)$ $(y \in \{-1, 1\})$

We want to use $\hat{y} = wx$







As $dim(x)$ grows and the training data shrinks, it becomes increasingly likely that some feature X_i allows overfitting!

How can we avoid over reliance on a feature?

1. minimize $\|w\|_2^2$ by adding $\lambda \|w\|_2^2$ to our loss function
   (λ: hyperparameter)

$$w - \eta \nabla_w (L + \lambda \|w\|_2^2) = \eta \nabla_w L + (1 - \eta 2\lambda) w$$

so we are performing "weight decay"

2. "Dropout" features at random:

During training,

$$x_i' = \begin{cases} 0 & \text{with probability } p \\ \frac{x_i}{1-p} & \text{with probability } 1-p \end{cases}$$

(p: hyperparameter)

$$\mathbb{E}[x_i'] = x_i$$

Mult:-layer Perceptron (MLP)

"linear model" → an increase or decrease in a feature must always increase or decrease the output.

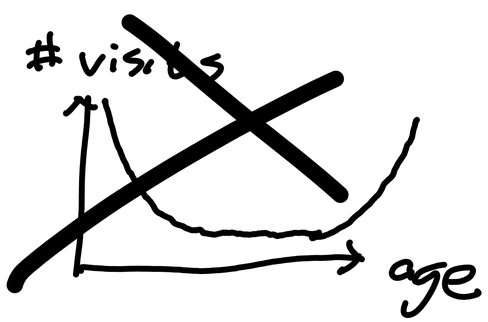
$$\hat{y} = wx + b$$

w positive: $x \uparrow$, $y \uparrow$

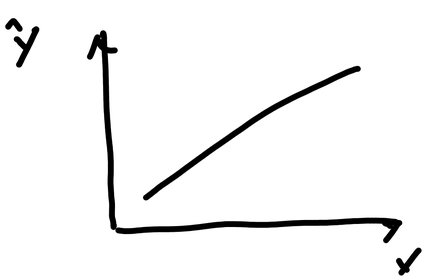
w negative: $x \downarrow$, $y \uparrow$

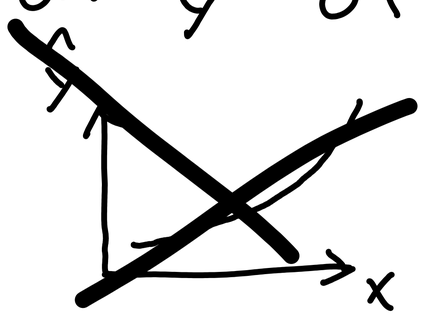
ex/ sqft ↑, value ↑

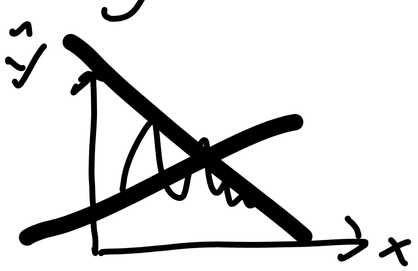
ex/ age ↑, # doctor visits



also, effect on $\hat{y}$ of changes in x is always the same.







How can we get more complex functions?

Stack multiple transformations.
(compose multiple functions)

Let's try:

$$H = XW_1 + b_1$$

hidden features — input

$$O = HW_2 + b_2$$

output

$$O = (XW_1 + b_1)W_2 + b_2$$

$$= XW_1W_2 + b_1W_2 + b_2$$

W = W_1W_2, b = $b_1W_2 + b_2$

$$= XW + b$$

Need nonlinearity! Apply nonlinear functions
(almost always elementwise)

$$H = \sigma(XW_1 + b_1) \qquad O = HW_2 + b_2$$

↳ nonlinear function!

"elementwise"

$$h_i = \sigma(x_i)$$

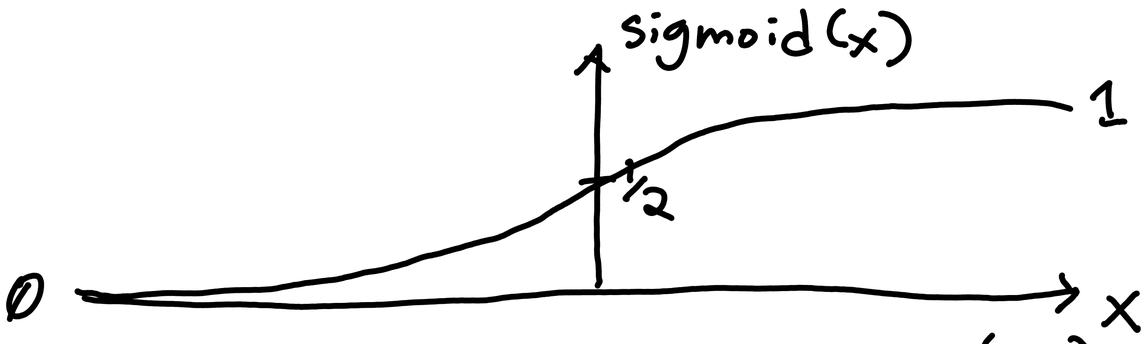
"Neural network"

"Multi-layer perceptron"

"Dense / feed-forward / fully connected"
(layer or network)

What to use for $\sigma(x)$?

$$\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$$



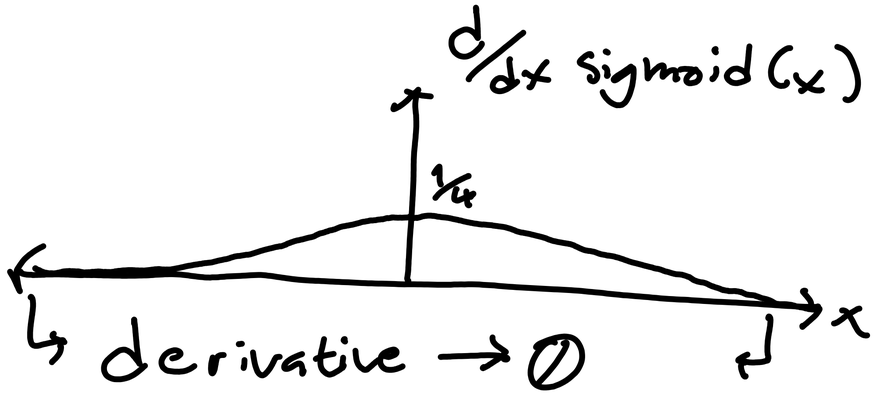
$$\frac{d}{dx}\,\text{sigmoid}(x) = \frac{\exp(-x)}{(1+\exp(-x))^2}$$

$$= \text{sigmoid}(x)\,(1-\text{sigmoid}(x))$$

$$1-\text{sigmoid}(x) = 1 - \frac{1}{1+\exp(-x)}$$

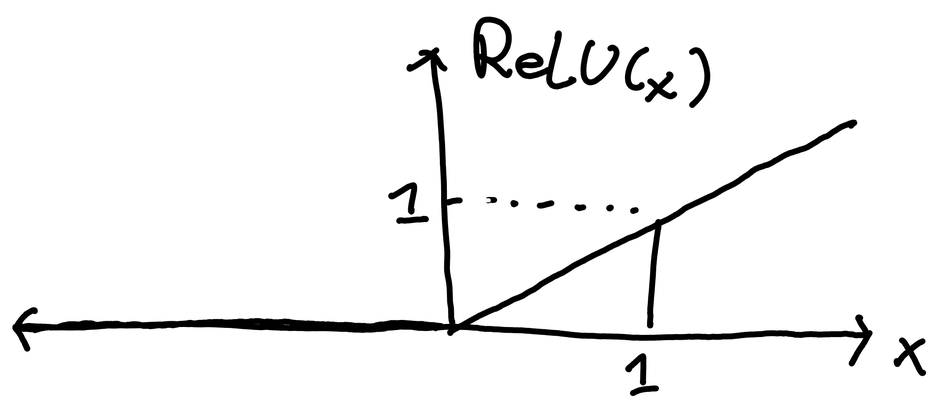
$$= \frac{1+\exp(-x)}{1+\exp(-x)} - \frac{1}{1+\exp(-x)}$$

$$= \frac{\exp(-x)}{1+\exp(-x)}$$

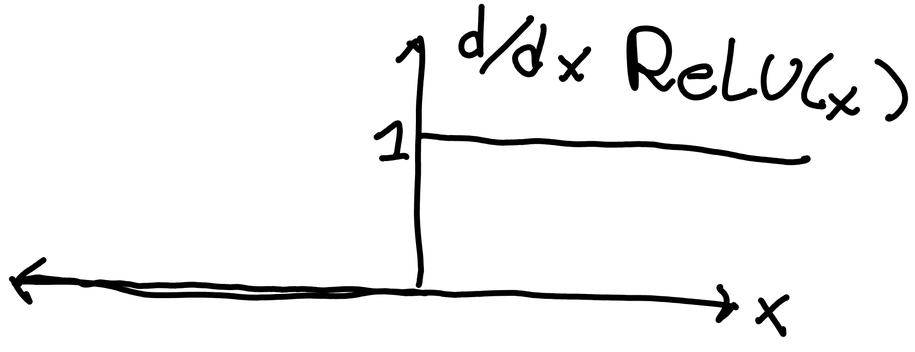


ReLU - "Rectified Linear Unit"

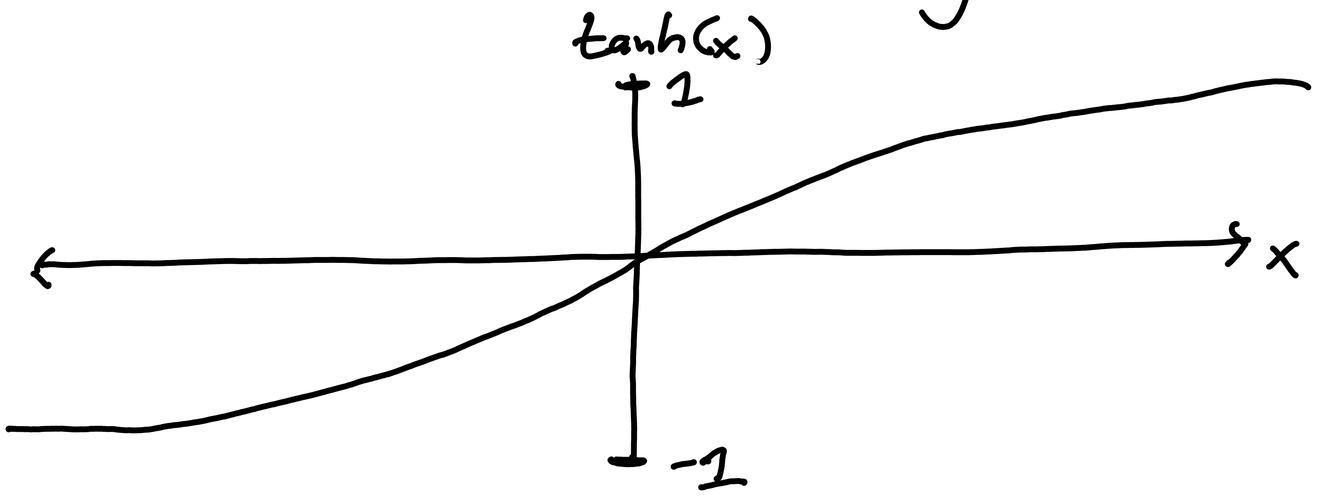
$$ReLU(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} = \max(0, x)$$



$$\frac{d\,ReLU}{dx} = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} = step(x)$$



$$\tanh(x) = 2\,\mathrm{sigmoid}(2x) - 1$$



Universal Approximation Theorem:

A sufficiently "wide" MLP with one hidden layer can approximate any function arbitrarily well.

("width" is the dimensionality of the hidden features)

Is universal approximation useful?

1. It means we can overfit. → regularization, pre-training
2. The model might be huge. → different architectures
3. It doesn't tell us how. → gradient descent (via backprop)

Gradient Descent

We want to minimize $f(x)$

$$f(x+\epsilon) = \sum_{n=0}^{\infty} \frac{\epsilon^n f^{(n)}(x)}{n!}$$

$$f(x+\underset{\text{small}}{\epsilon}) = f(x) + \epsilon f'(x) + O(\epsilon^2)$$

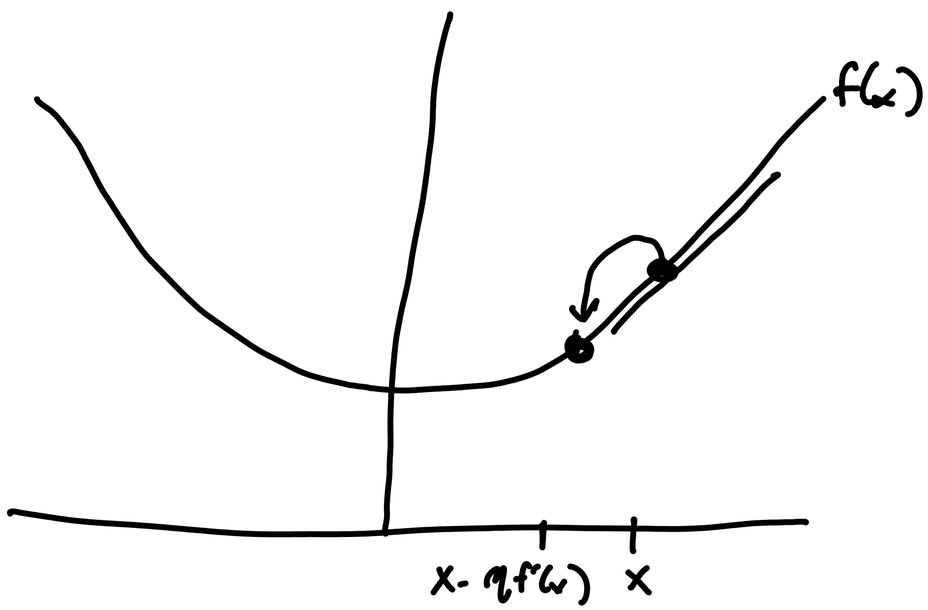
set $\epsilon = -\eta f'(x)$

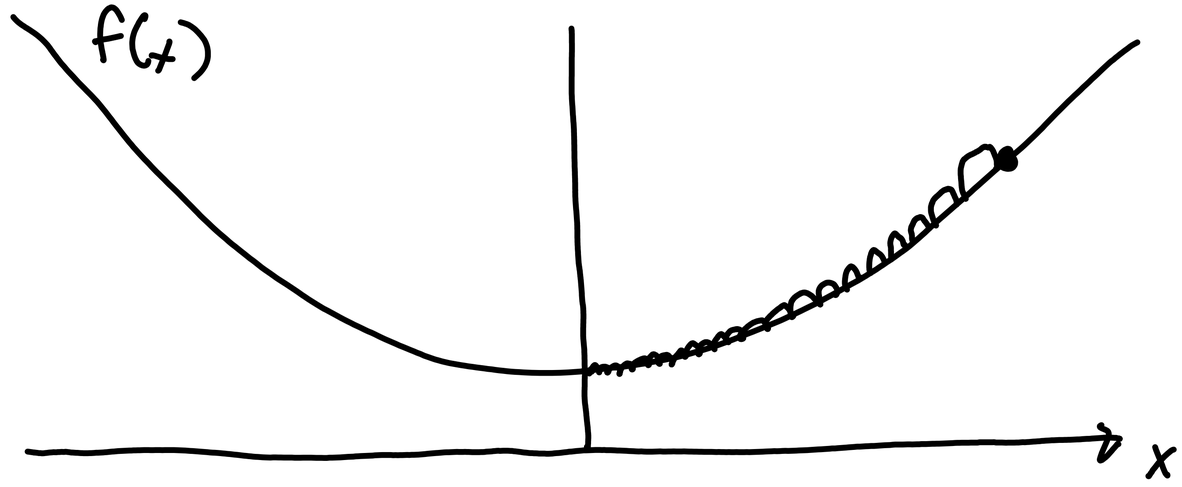
$$f(x-\eta f'(x)) = f(x) - \eta (f'(x))^2 + O(\eta^2 (f'(x))^2)$$

therefore

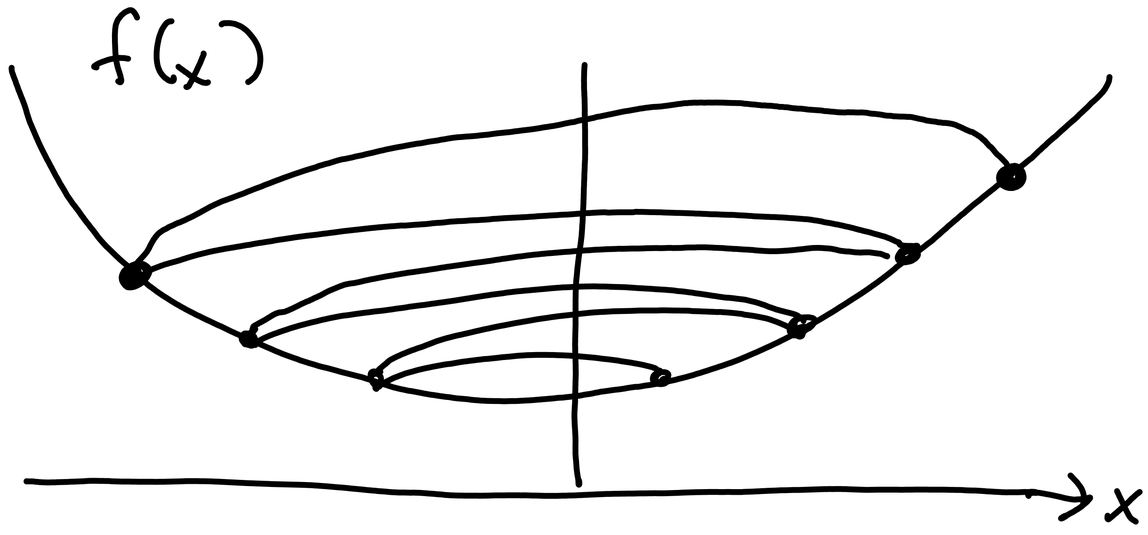
$$f(x-\eta f'(x)) \lesssim f(x)$$

so: $x \leftarrow x - \eta f'(x)$

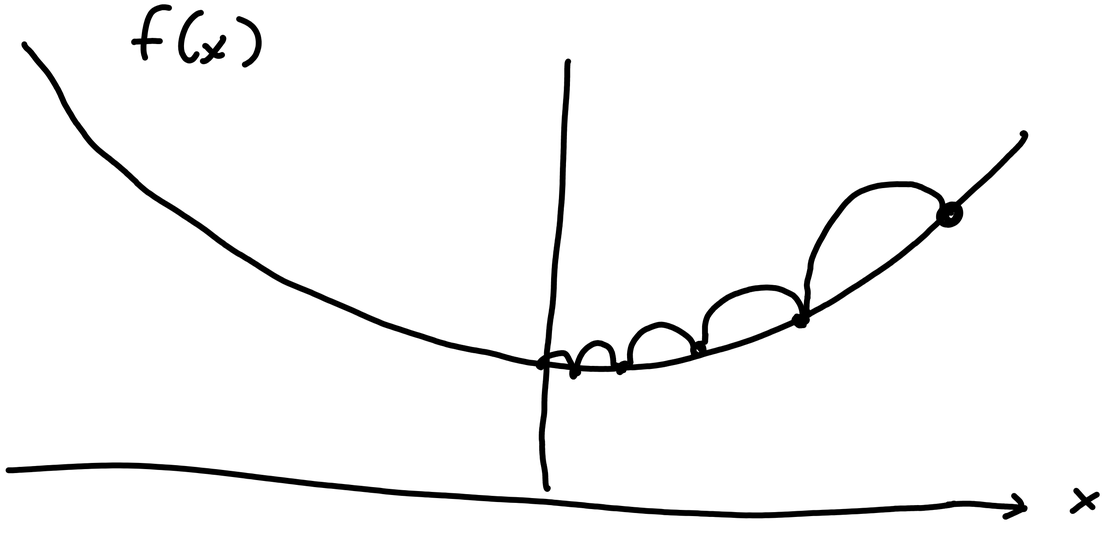




Small η



big η



"just right" η