FUNCTION PARAMETERIZATION

Function transformations

Transformation Rules for Functions		
Function Notation	Type of Transformation	Change to Coordinate Point
f(x) + d	Vertical translation up d units	(x, y) → (x, y + d)
f(x) – d	Vertical translation down d units	(x, y) → (x, y – d)
f(x + c)	Horizontal translation left c units	(x, y) → (x – c, y)
f(x – c)	Horizontal translation right c units	(x, y) → (x + c, y)
–f(x)	Reflection over x-axis	(x, y) → (x, –y)
f(–x)	Reflection over y-axis	(x, y) → (–x, y)
af(x)	Vertical stretch for |a|> 1	(x, y) → (x, ay)
	Vertical compression for 0 < |a| < 1	
f(bx)	Horizontal compression for |b| > 1	$(x, y) \rightarrow \left(\frac{x}{b}, y\right)$
	Horizontal stretch for 0 < |b| < 1	

Rigid

Non-Rigid

https://www.onlinemathlearning.com/parent-functions.html

Parameterizing a function: $y = f(x \mid \mathbf{p}) = Af\left(\frac{x - x_o}{w}\right) + S$

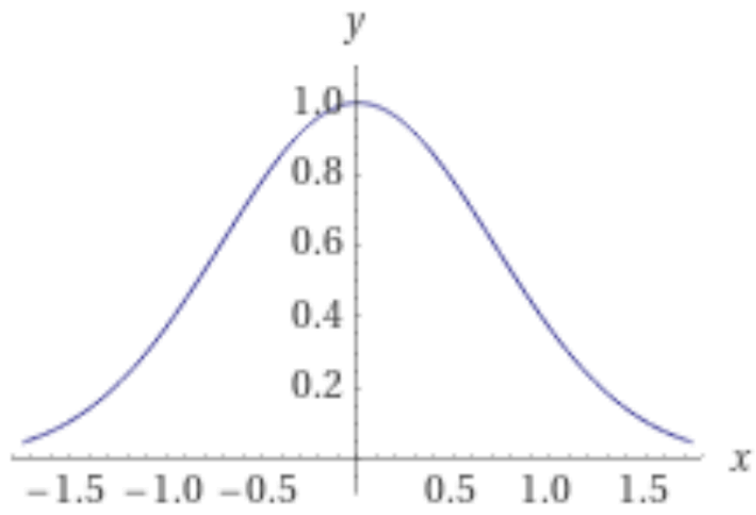
Parameters and Function transformations

PARAMETRIC MODEL = Functional form + Parameterization

(underlying shape) (stretching and shifting)

$y = f(x) \rightarrow$ **"parent function"**

$$f(x) = e^{-x^2}$$



Parameterized form-1:

$$y = f(x \mid \mathbf{p}) = Af\left(\frac{x - x_o}{w}\right) + S$$

$\mathbf{p} = (A, x_o, w, S)$

$A =$ **verticle stretch (amplitude)**

$x_o =$ **horizontal shift (recentering parameter)**

$w =$ **horizontal stretch (width parameter)**

$S =$ **verticle shift (shift parameter)**

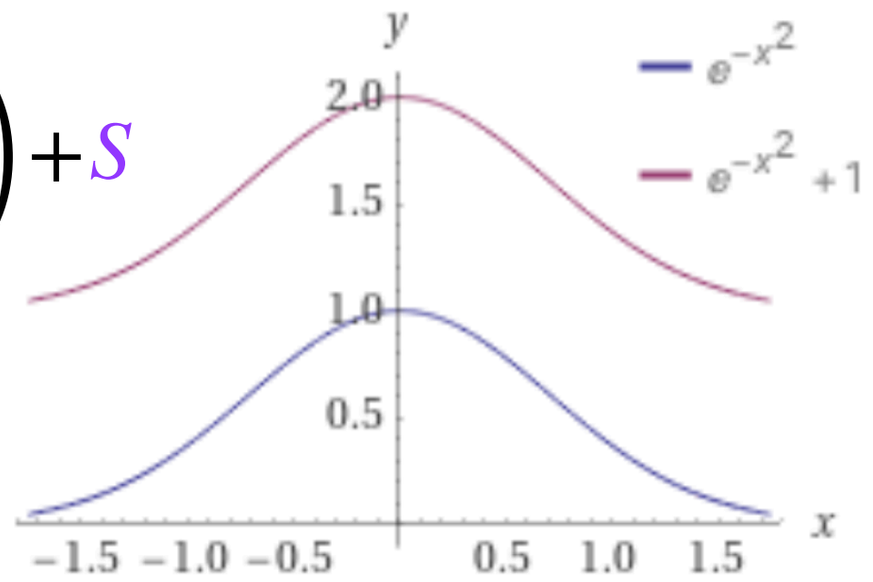
Parameterized form-2: $y = f(x \mid \mathbf{p}) = Af(w_1 x + b) + S$

(weights and bias) $w_1 = \frac{1}{w}$ **and** $b = -\frac{x_o}{w}$

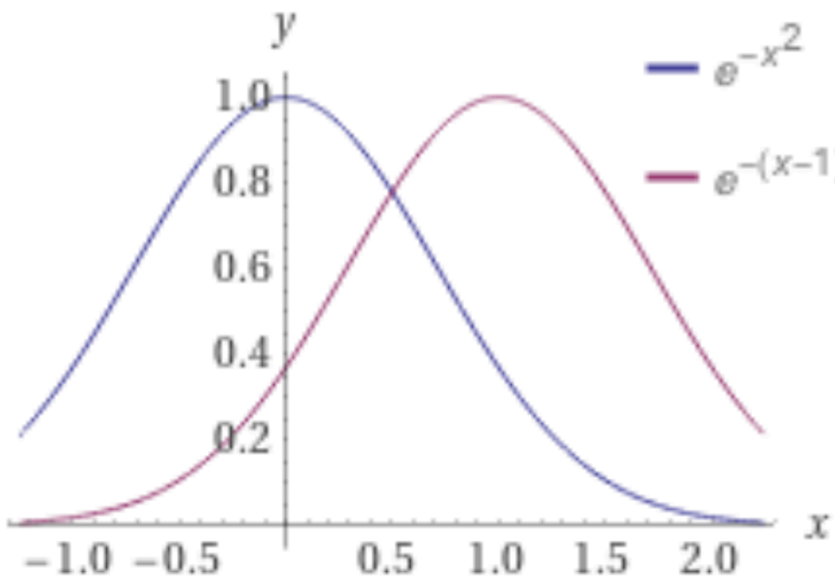
Example: Gaussian

$$y = f(x\,|\,\mathbf{p}) = Af\left(\frac{x - x_o}{w}\right) + S$$

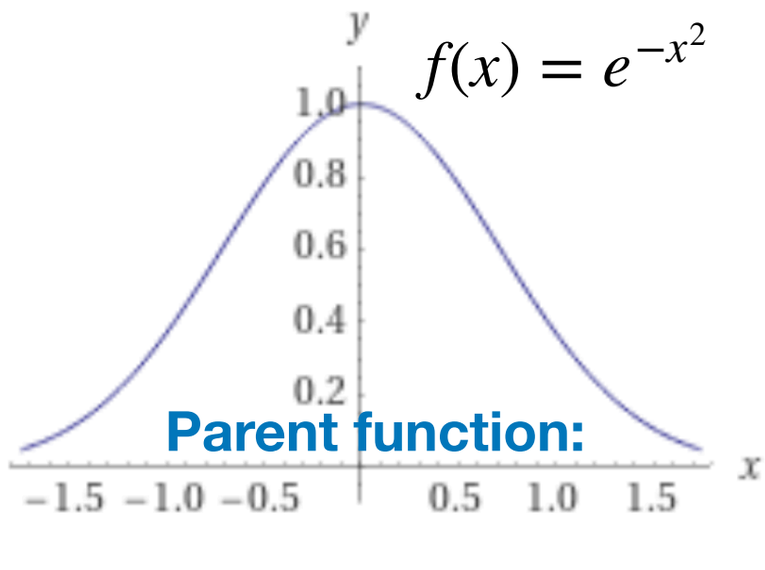
addition outside (rigid verticle shift): $f(x) + a$



addition inside (rigid horizontal shift): $f(x - a)$

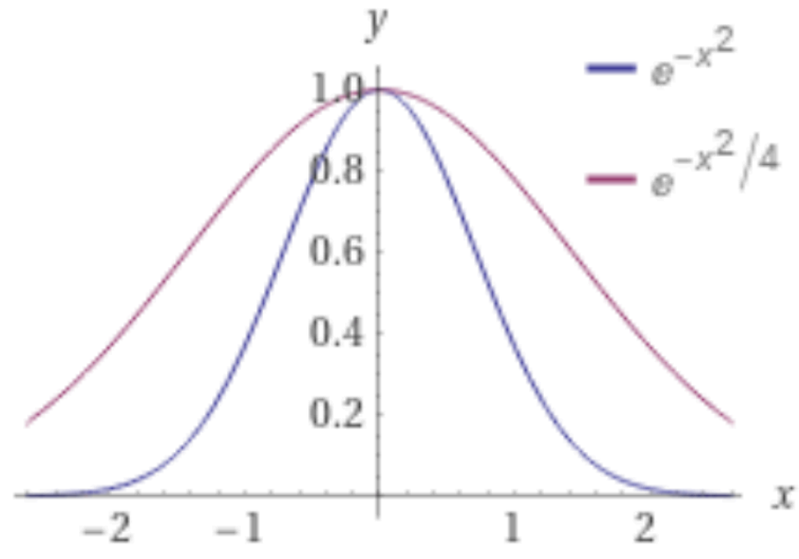


$f(x) = e^{-x^2}$

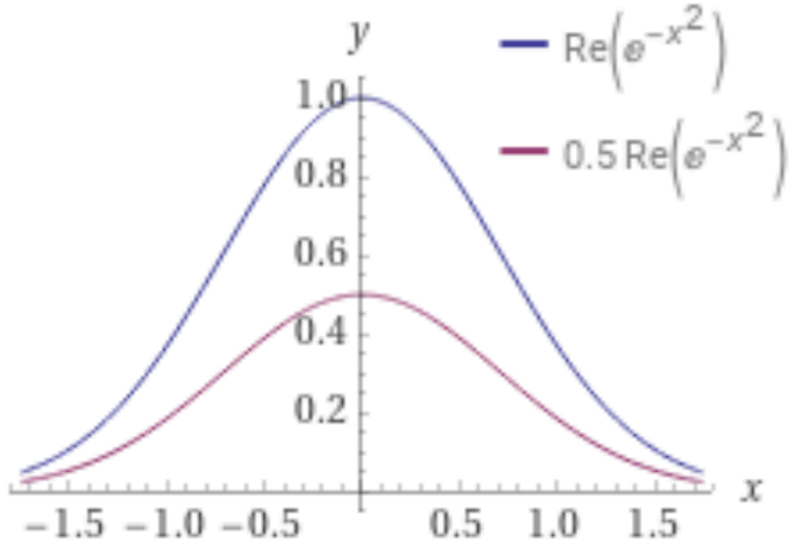


Parent function:

multiplication inside (non-rigid horizontal scaling): $f(ax)$

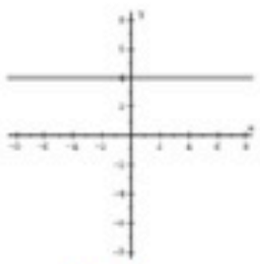
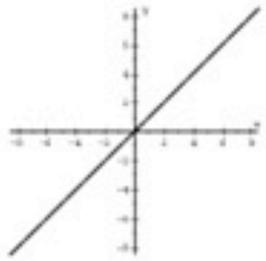
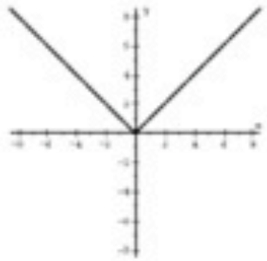
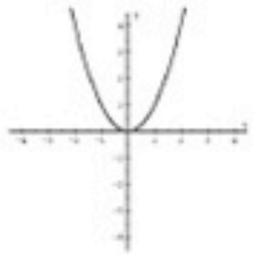
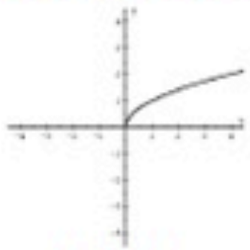
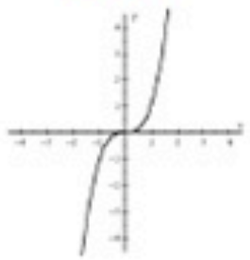
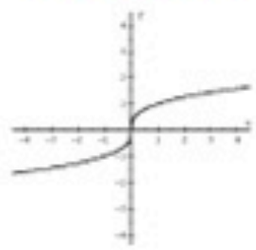
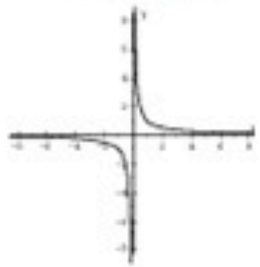
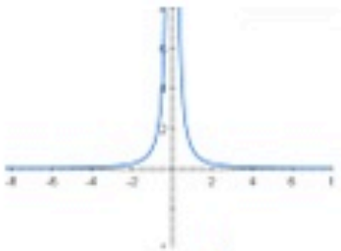
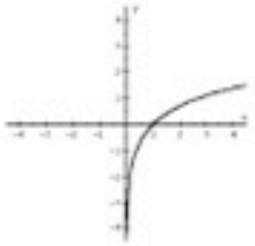
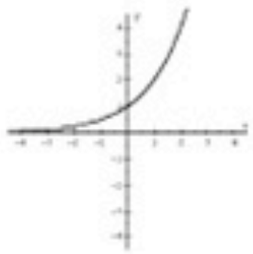
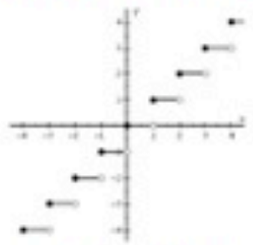
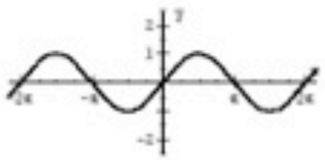
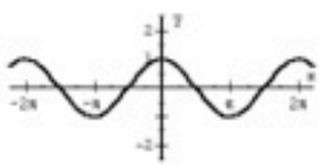
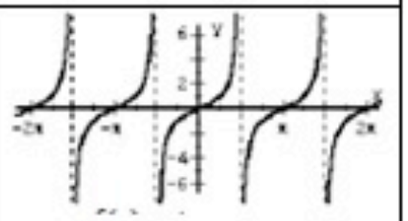


multiplication outside (non-rigid verticle scaling): $af(x)$



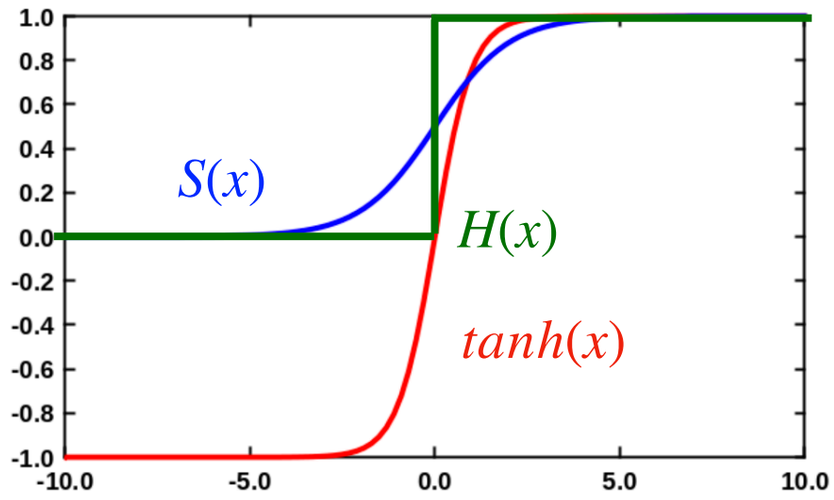
Common parent functions

Model=Functional form + Parameterization

Constant  $f(x) = c$	Linear  $f(x) = x$	Absolute Value  $f(x) = \lvert x \rvert$	Quadratic  $f(x) = x^2$
Square Root  $f(x) = \sqrt{x}$	Cubic  $f(x) = x^3$	Cube Root  $f(x) = \sqrt[3]{x}$	Reciprocal/Inverse/ Rational  $f(x) = \frac{1}{x}$
Rational  $f(x) = \frac{1}{x^2}$	Logarithmic  $f(x) = \ln(x)$	Exponential  $f(x) = e^x$	Greatest Integer (Step Function)  $f(x) = [[x]]$
Trigonometric Functions →	 $f(x) = \sin(x)$	 $f(x) = \cos(x)$	 $f(x) = \tan(x)$

http://calcwithtully.weebly.com/parent-functions.html

IMPORTANT FUNCTIONS IN DEEP LEARNING



Sigmoid (aka. logistic) function

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

Hyperbolic Tangent function

$$\tanh(x) = 2\left(S(2x) - 0.5\right)$$

"Heaviside (aka. step) function"

$$H(x) := \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases}$$

Basic models

MODEL = Functional form + Parameterization

(underlying shape) (stretching and shifting) $\mathbf{p} = (A, x_o, w, S)$

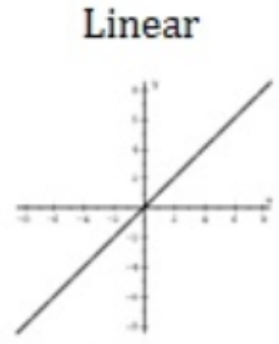
$y = f(x) \rightarrow$ **"parent function"**

$$y = f(x \mid \mathbf{p}) = A f\left(\frac{x - x_o}{w}\right) + S$$

Examples:

$$y = f(x \mid \mathbf{p}) = mx + b$$

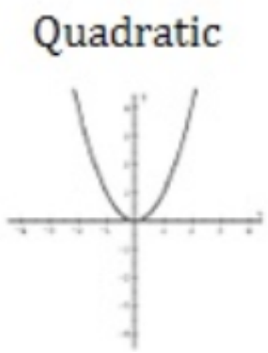
Linear



$\mathbf{p} = (m, b)$

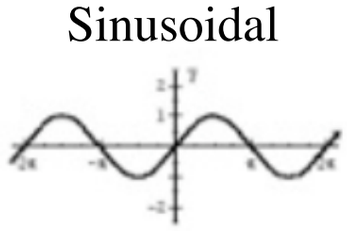
$$y = f(x \mid \mathbf{p}) = Ax^2 + Bx + C$$

Quadratic regressions



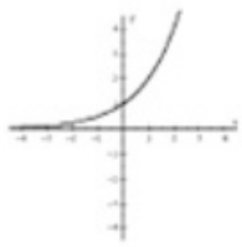
$\mathbf{p} = (A, B, C)$

$$y = f(x \mid \mathbf{p}) = A_o \sin(\omega t + \phi) + c$$



$\mathbf{p} = (A_o, \omega, \phi, c)$

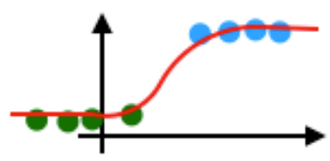
$$y = f(x \mid \mathbf{p}) = ae^{\frac{(x - x_o)}{w}} + d$$



$\mathbf{p} = (a, w, x_o, d)$

$$y = f(x \mid \mathbf{p}) = \frac{A}{1 + e^{-\frac{(x - x_o)}{w}}} + s$$

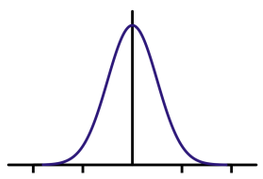
Logistic



$\mathbf{p} = (A, w, x_o, S)$

$$y = f(x \mid \mathbf{p}) = A \cdot \exp^{-\left(\frac{x - x_o}{w}\right)^2} + S$$

Gaussian



$\mathbf{p} = (A, x_o, w, S)$

Higher order models

INPUTS $\rightarrow x$ **(variable)** $\rightarrow$ Model=Functional form + Parameterization

OUTPUTS $\rightarrow y$ **(resposne)**

Polynomial regression:

$$y = f(x) = \sum_{n=0}^{N} A_n x^n$$

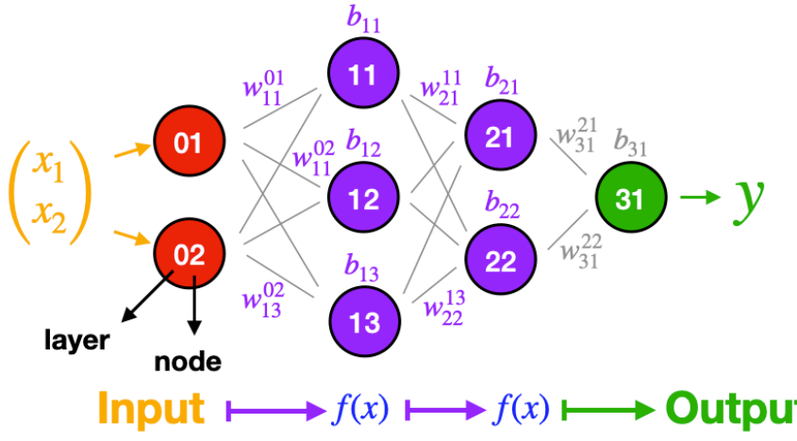
$$\mathbf{P} = (A_0, A_1, \ \dots \ , A_N)$$

Fourier series

$$y = f(x) = \frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi}{P} n x\right) + b_n \sin\left(\frac{2\pi}{P} n x\right) \right)$$

$$\mathbf{P} = (a_0, b_0, a_1, b_1 \ \dots \ a_N,)$$

ANNs



Neural network model

$$\mathbf{y} = ANN(\mathbf{x} \,|\, \mathbf{p})$$

fitting parameters

Module-3

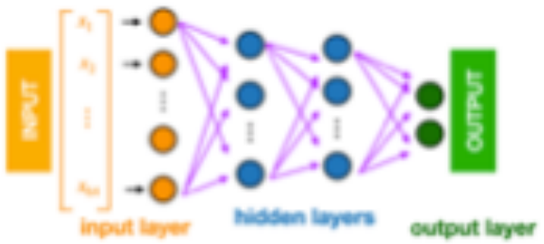
More parameters → more complex model (more flexible)

Taylor series

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \approx f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

PARAMETRIC MODELING

polynomial regression
logistic regression
mixture models, k-means
hidden Markov models
factor analysis / pPCA / PMF
Linear regression
Artificial Neural networks

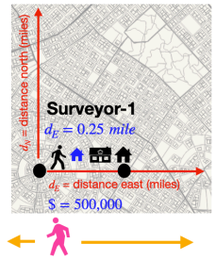


Auto-regression algorithms (timeseries)
General curve fitting

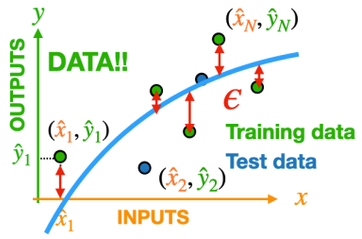
Common ML workflow summary

Source and recommended reading: Chollet p111-p116

- **Defining the problem and assembling a dataset**



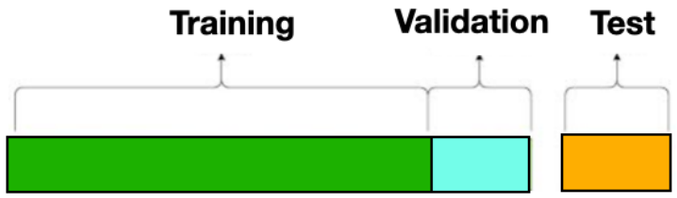
- **Choosing a measure of success**
  - **Choosing the accuracy/error metrics and a loss function**

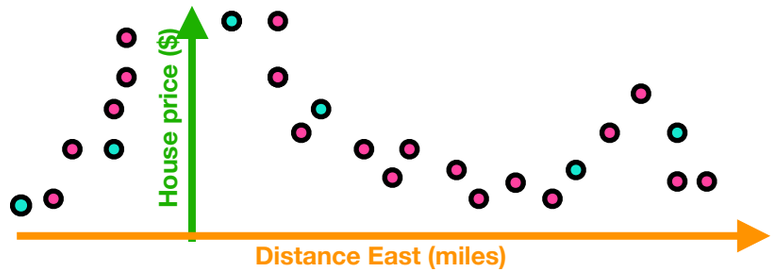


RMSE:

$$\epsilon(\mathbf{p}) = \left(\frac{\sum_i^N \left(y(\hat{x}_i \,|\, \mathbf{p}) - \hat{y}_i \right)^2}{N} \right)^{\frac{1}{2}}$$

- **Deciding on an evaluation protocol**





- **Preparing your data**
  Makes it easier to choice parameter initial conditions (IC)

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.25 \\ 0.30 \\ \vdots \\ 3.14 \end{pmatrix} \quad x_i \to \tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x} \approx [-1.0\,,1.0] \qquad \hat{\mathbf{y}} = \begin{pmatrix} 500{,}000 \\ 410{,}000 \\ \vdots \\ 210{,}000 \end{pmatrix} \quad y_i \to \tilde{y}_i = \frac{y_i - \mu_y}{\sigma_y} \approx [-1.0\,,1.0]$$

$\sigma_x \;\; \mu_x$ $\qquad$ $\sigma_y \;\; \mu_y$

- **Developing a model that does better than some baseline**

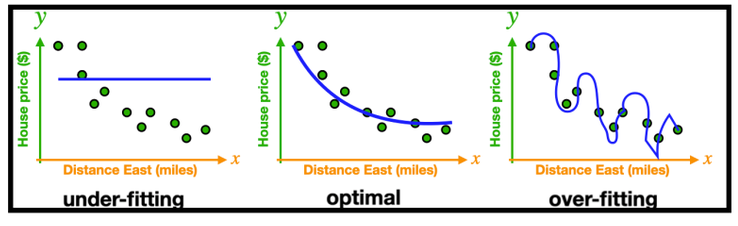
Select a model

$$M(x\,|\,\mathbf{p}) = \text{Model} = \text{Functional form} + \text{Parameterization}$$

$$M(x\,|\,\mathbf{p}) = p_1 e^{-\left(\frac{x-p_2}{p_3}\right)^2} + p_4 e^{-\left(\frac{x-p_5}{p_6}\right)^2} + p_7$$

$$\mathbf{p} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)$$

- **Choosing an optimizer, learning rate, and other hyper-parameters**

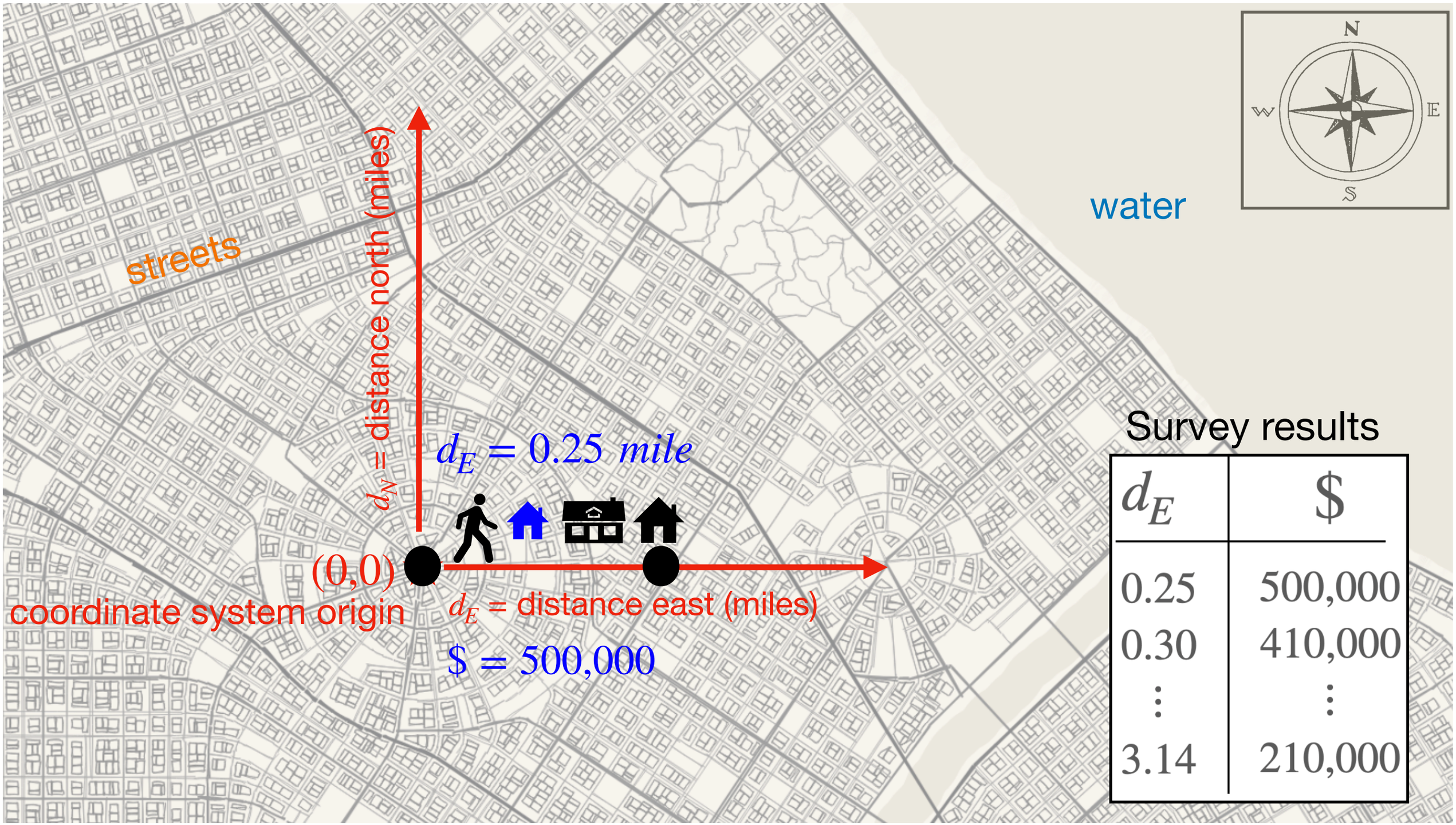


★**Scaling up: developing a model that overfits**

★**Regularizing your model and tuning your hyper-parameters**

Real Estate Price example

City map



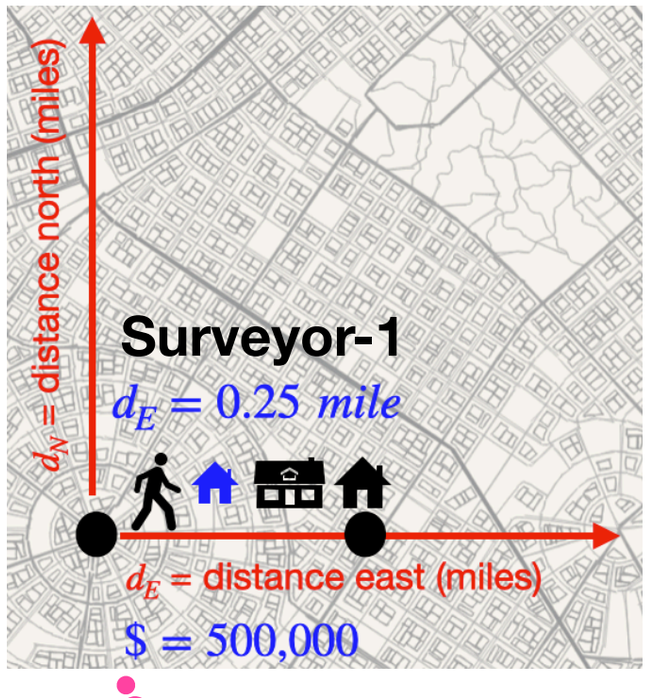
Survey results

d_E	$
0.25	500,000
0.30	410,000
⋮	⋮
3.14	210,000

https://maps.probabletrain.com/#/

$N = 50$

Data



Spreadsheet:

$N = 50$

d_E	$
0.25	500,000
0.30	410,000
⋮	⋮
3.14	210,000

ordered Pairs

$(\hat{x}_1, \hat{y}_1)$ Sample-1

Sample-2

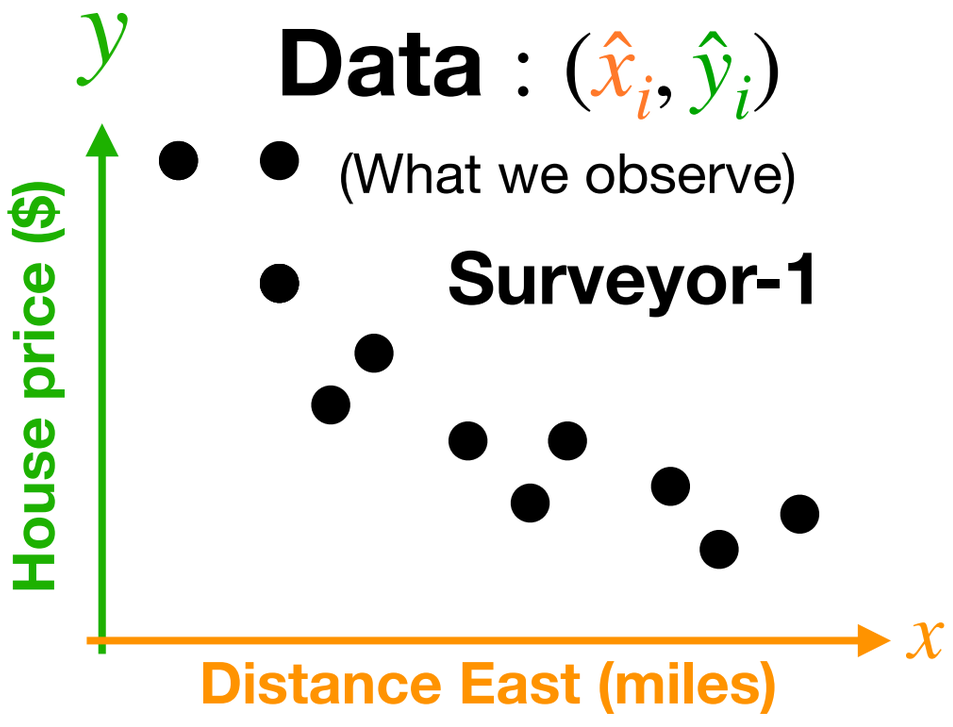
⋮

$(\hat{x}_{50}, \hat{y}_{50})$ Sample-50

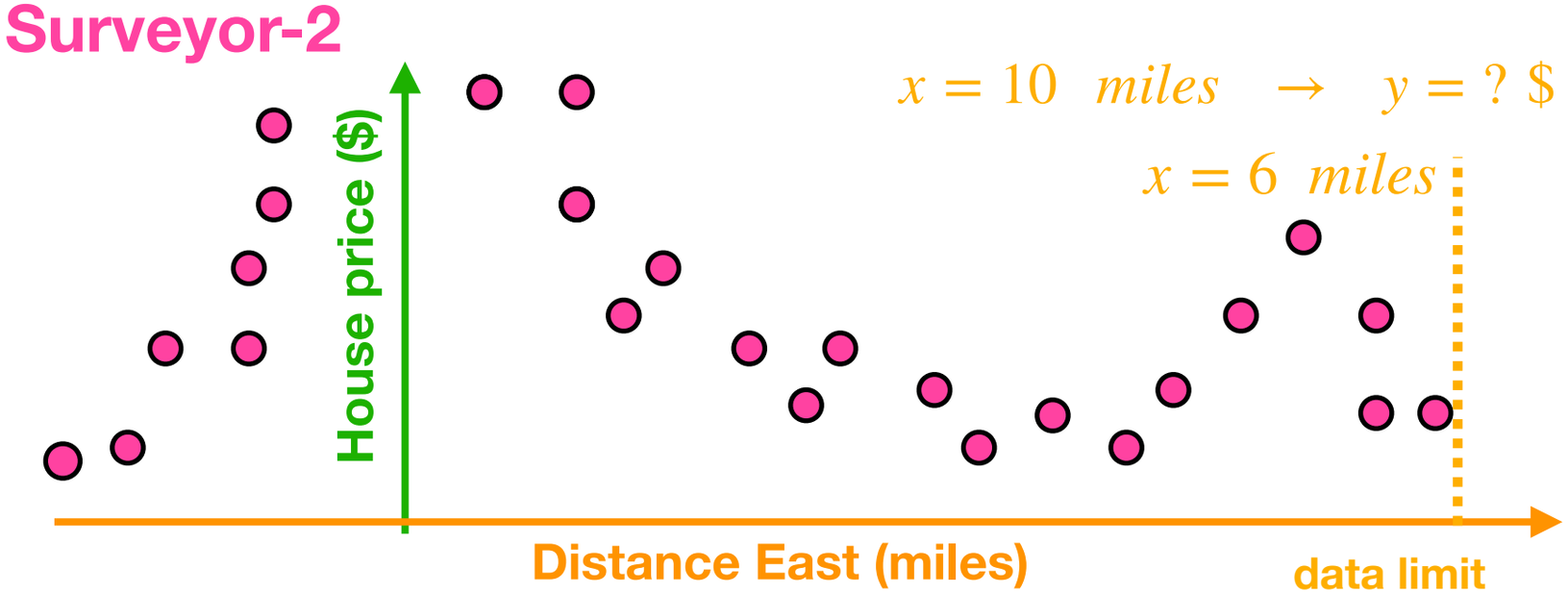
Inputs: $\hat{x}_i$ **Outputs:** $\hat{y}_i$

Hat —> data

No-Hat —> continuous variable



"batch=stacked samples"



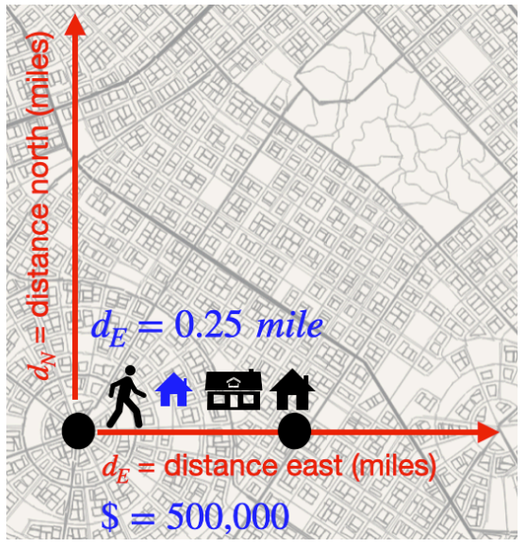
- Data itself generally has little **predictive power**
  - Can only see **local** qualitative trends
    - Are the values big or small
    - Are they going up or down?
    - Are they changing rapidly or slowly?

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.25 \\ 0.30 \\ \vdots \\ 3.14 \end{pmatrix}$$ Sample-1, Sample-2, Sample-50

"Sample" dimension

$$\hat{\mathbf{y}} = \begin{pmatrix} 500{,}000 \\ 410{,}000 \\ \vdots \\ 210{,}000 \end{pmatrix}$$ Sample-1, Sample-2, Sample-50

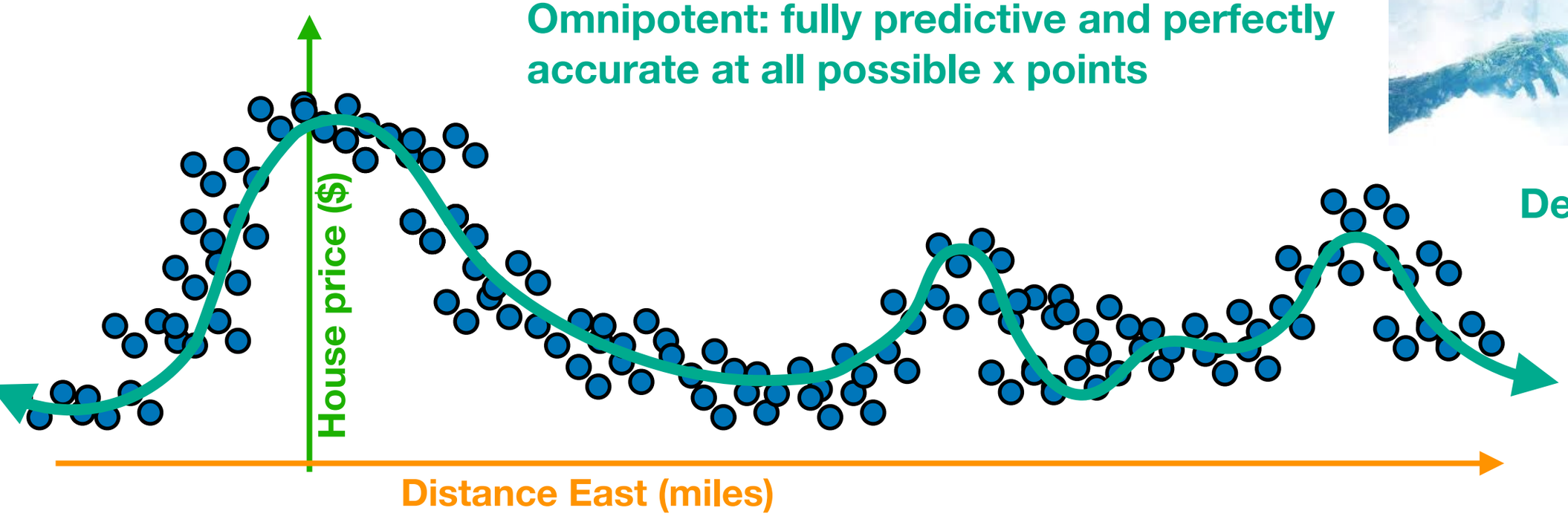
Data and the “ground truth”



Data: what surveyor-1 "sees"

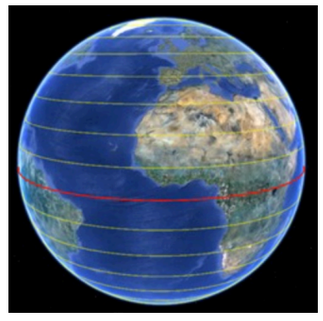
Data: what surveyor-2 "sees"

Data generating function: $G(x)$ → UNKNOWN “ground truth”



Omnipotent: fully predictive and perfectly accurate at all possible x points

Defined everywhere

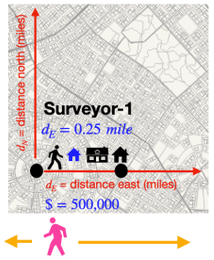


Ideal data -> infinite sampling capability $N \to \infty$

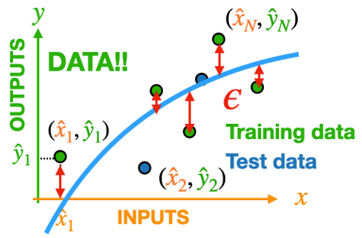
Common ML workflow summary

Source and recommended reading: Chollet p111-p116

- **Defining the problem and assembling a dataset**



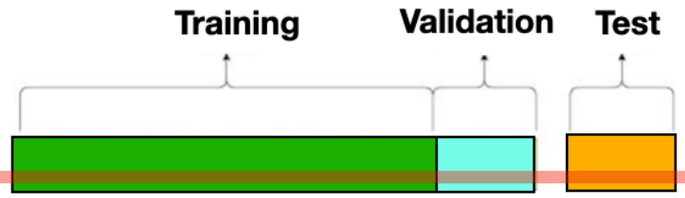
- **Choosing a measure of success**
  - **Choosing the accuracy/error metrics and a loss function**

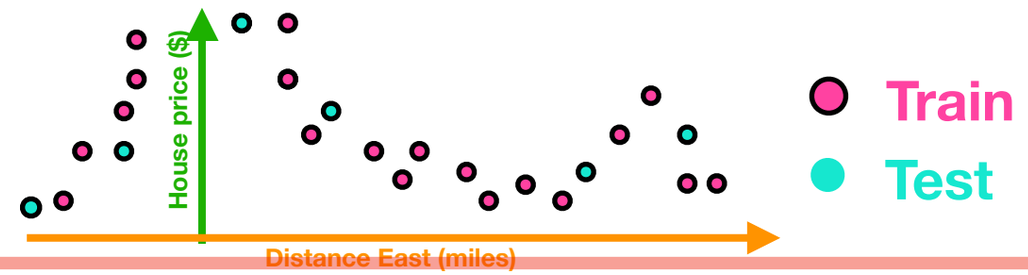


RMSE:

$$\epsilon(\mathbf{p}) = \left(\frac{\sum_i^N \left(y(\hat{x}_i \,|\, \mathbf{p}) - \hat{y}_i \right)^2}{N} \right)^{\frac{1}{2}}$$

- **Deciding on an evaluation protocol**





- **Preparing your data**

  Makes it easier to choice parameter initial conditions (IC)

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.25 \\ 0.30 \\ \vdots \\ 3.14 \end{pmatrix} \quad x_i \rightarrow \tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x} \approx [-1.0\,,\,1.0]$$

$\sigma_x \;\; \mu_x$

$$\hat{\mathbf{y}} = \begin{pmatrix} 500{,}000 \\ 410{,}000 \\ \vdots \\ 210{,}000 \end{pmatrix} \quad y_i \rightarrow \tilde{y}_i = \frac{y_i - \mu_y}{\sigma_y} \approx [-1.0\,,\,1.0]$$

$\sigma_y \;\; \mu_y$

- **Developing a model that does better than some baseline**

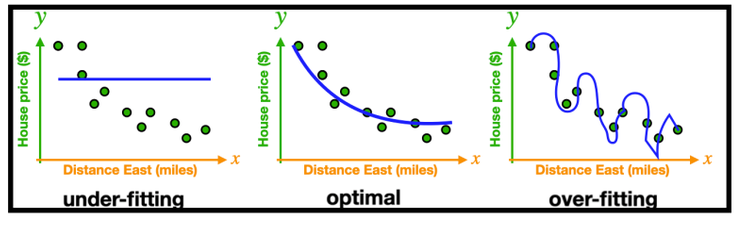
  Select a model

$$M(x \,|\, \mathbf{p}) = \text{Model} = \text{Functional form} + \text{Parameterization}$$

$$M(x \,|\, \mathbf{p}) = p_1 e^{-\left(\frac{x - p_2}{p_3}\right)^2} + p_4 e^{-\left(\frac{x - p_5}{p_6}\right)^2} + p_7$$

$$\mathbf{p} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)$$

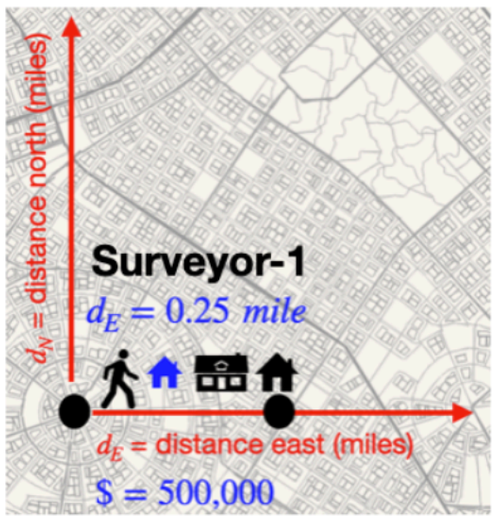
- **Choosing an optimizer, learning rate, and other hyper-parameters**



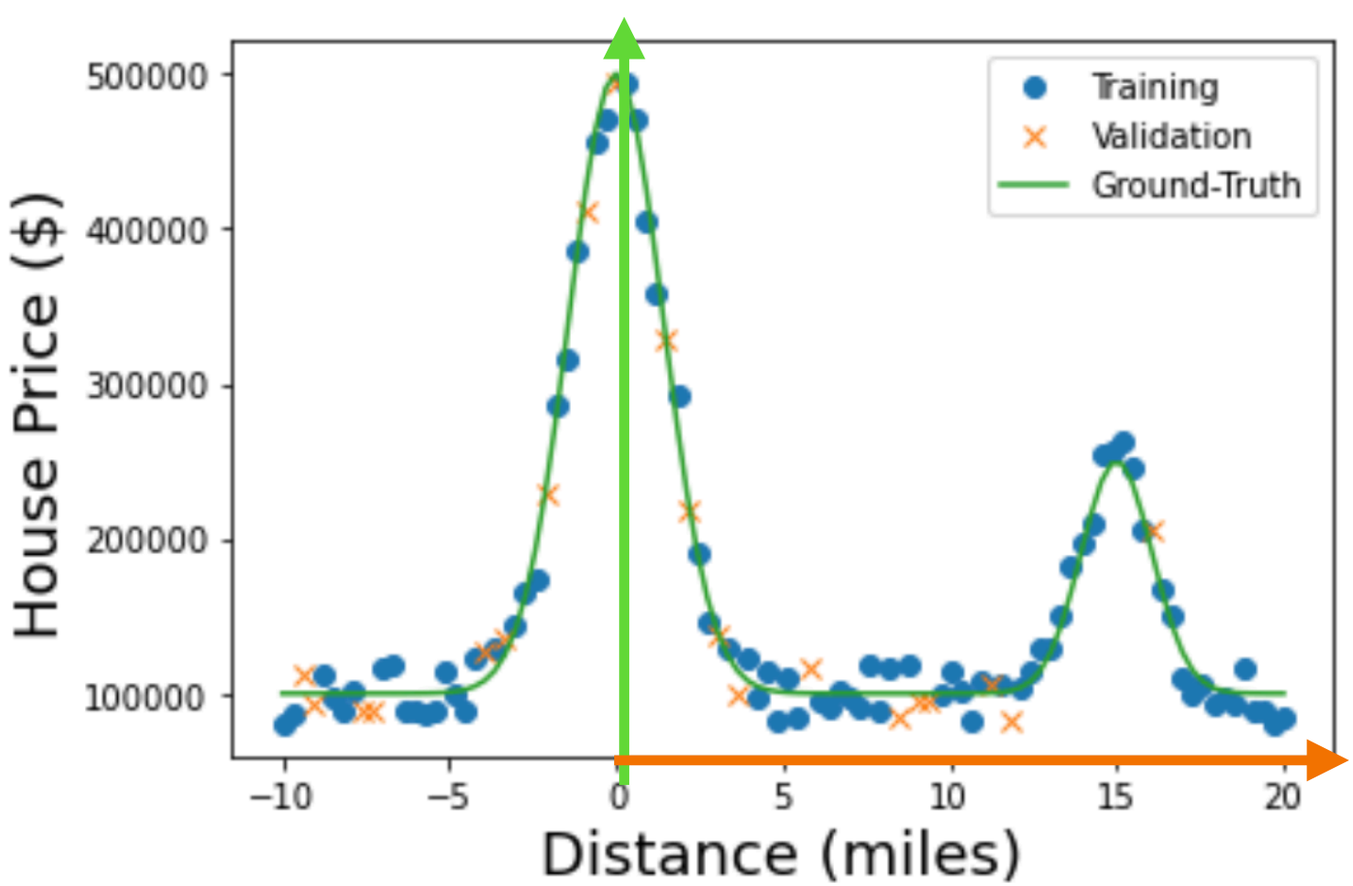
★**Scaling up: developing a model that overfits**

★**Regularizing your model and tuning your hyper-parameters**

Normalize



Gaussian



$$\hat{\mathbf{y}} = \begin{pmatrix} 500{,}000 \\ 410{,}000 \\ \vdots \\ 210{,}000 \end{pmatrix}$$

$\sigma_y \quad \mu_y$

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.25 \\ 0.30 \\ \vdots \\ 3.14 \end{pmatrix}$$

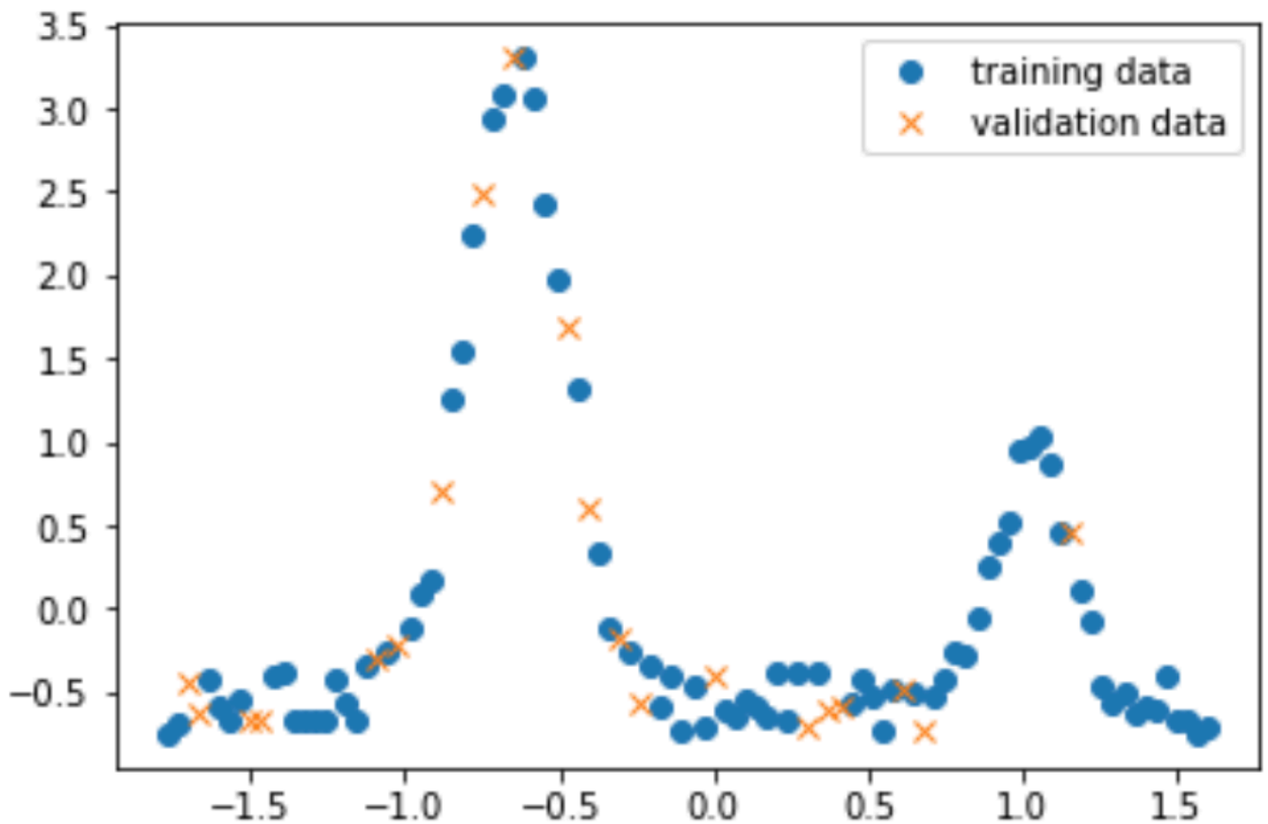
$\sigma_x \quad \mu_x$

```
ux=np.mean(x_train)
sx=np.std(x_train)
x_train = (x_train-ux)/sx
y_train = (y_train-uy)/sy
```

Transformed that data Into a dimensionless space that runs from roughly -3 to 3 (i.e. unit-less space)

Normalize

$$x_i \rightarrow \tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x} \qquad y_i \rightarrow \tilde{y}_i = \frac{y_i - \mu_y}{\sigma_y}$$

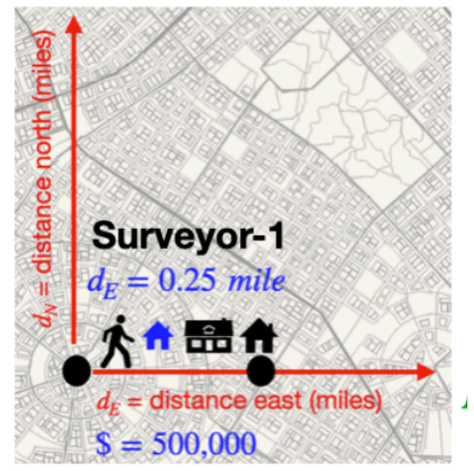


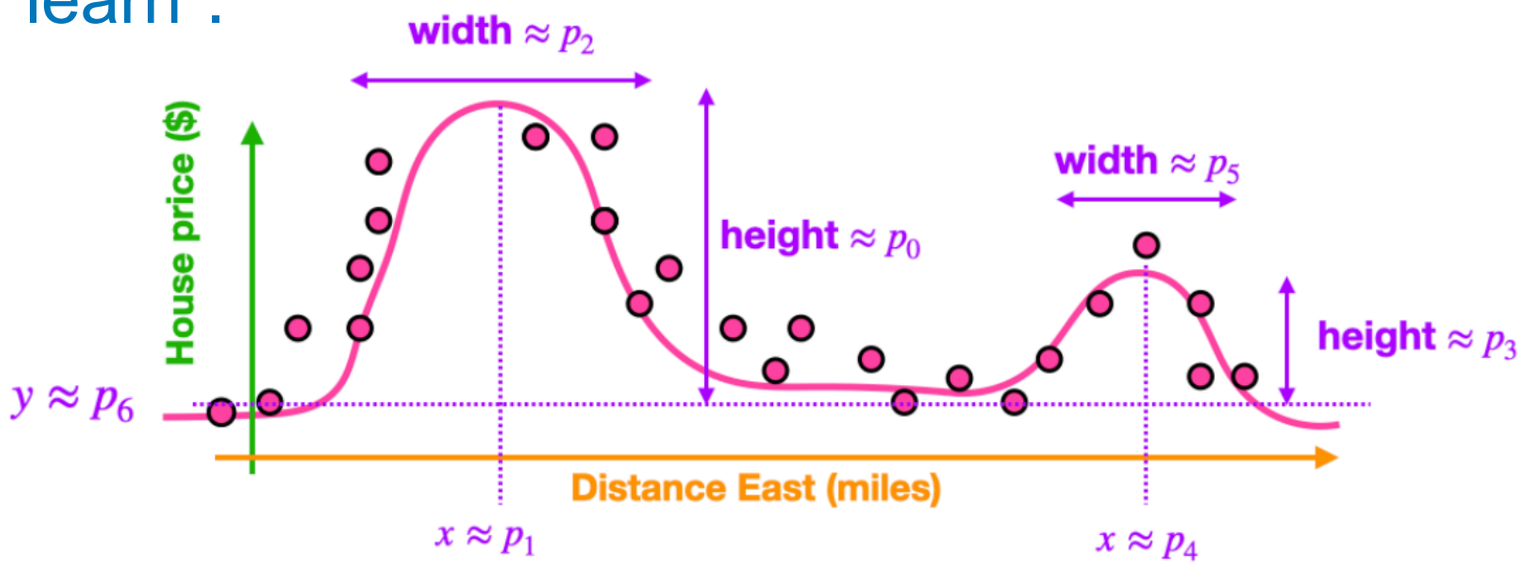
Why do we normalize?

1) Normalization forces the data into a standardized space, centered near the origin (0,0), with a data range ~ (-3 to 3). This effectively "gets rid" of units. However, this range depends on the statistical distribution of the original data

2) Normalization makes visualization more effective, since all numerical variables span roughly the same range

3) In this normalized space it is MUCH easier for machine learning algorithms to "learn" the data. For example, it parametric modeling we try to "learn" a curve that fits the data by adjusting (i.e. learning) the model parameters. In the normalized space, these parameters will also tend to be in the range (-3 to 3), and therefore easier to "learn".





$$M(x \mid \mathbf{p}) = \text{Model} = \text{Functional form} + \text{Parameterization}$$

$$M(x \mid \mathbf{p}) = p_0 e^{-\left(\frac{x-p_1}{p_2}\right)^2} + p_3 e^{-\left(\frac{x-p_4}{p_5}\right)^2} + p_6$$

$$\mathbf{p} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6)$$

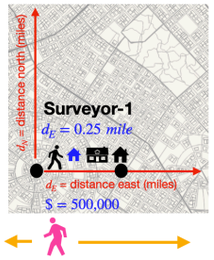
"The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model"

https://developers.google.com/machine-learning/data-prep/transform/normalization

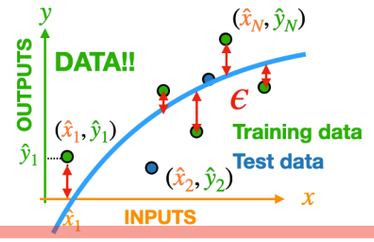
Common ML workflow summary

Source and recommended reading: Chollet p111-p116

- **Defining the problem and assembling a dataset**



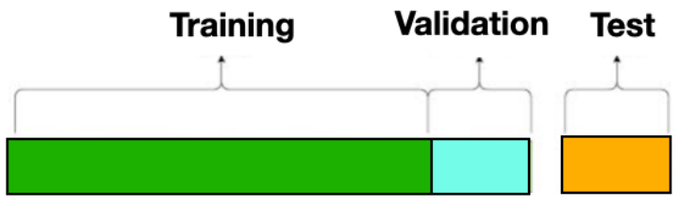
- **Choosing a measure of success**
  - **Choosing the accuracy/error metrics and a loss function**

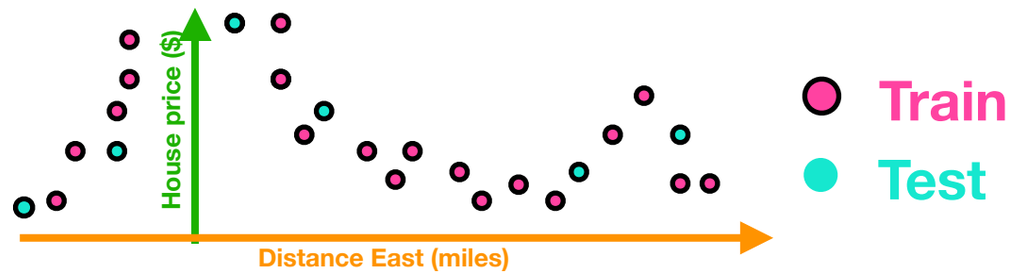


RMSE:

$$\epsilon(\mathbf{p}) = \left(\frac{\sum_i^N \left(y(\hat{x}_i \,|\, \mathbf{p}) - \hat{y}_i \right)^2}{N} \right)^{\frac{1}{2}}$$

- **Deciding on an evaluation protocol**





- **Preparing your data**
  Makes it easier to choice parameter initial conditions (IC)

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.25 \\ 0.30 \\ \vdots \\ 3.14 \end{pmatrix} \quad x_i \rightarrow \tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x} \approx [-1.0, 1.0] \qquad \hat{\mathbf{y}} = \begin{pmatrix} 500{,}000 \\ 410{,}000 \\ \vdots \\ 210{,}000 \end{pmatrix} \quad y_i \rightarrow \tilde{y}_i = \frac{y_i - \mu_y}{\sigma_y} \approx [-1.0, 1.0]$$

$\sigma_x \;\; \mu_x$ $\sigma_y \;\; \mu_y$

- **Developing a model that does better than some baseline**

Select a model

$$M(x \,|\, \mathbf{p}) = \text{Model} = \text{Functional form} + \text{Parameterization}$$

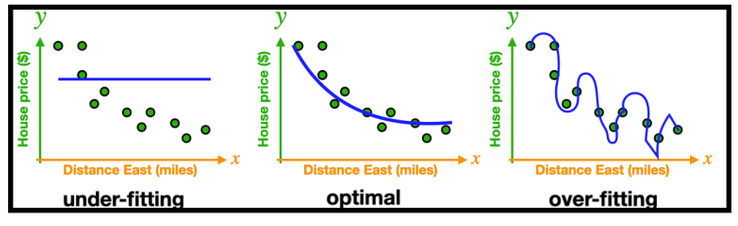
$$M(x \,|\, \mathbf{p}) = p_1 e^{-\left(\frac{x - p_2}{p_3}\right)^2} + p_4 e^{-\left(\frac{x - p_5}{p_6}\right)^2} + p_7$$

$$\mathbf{p} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)$$

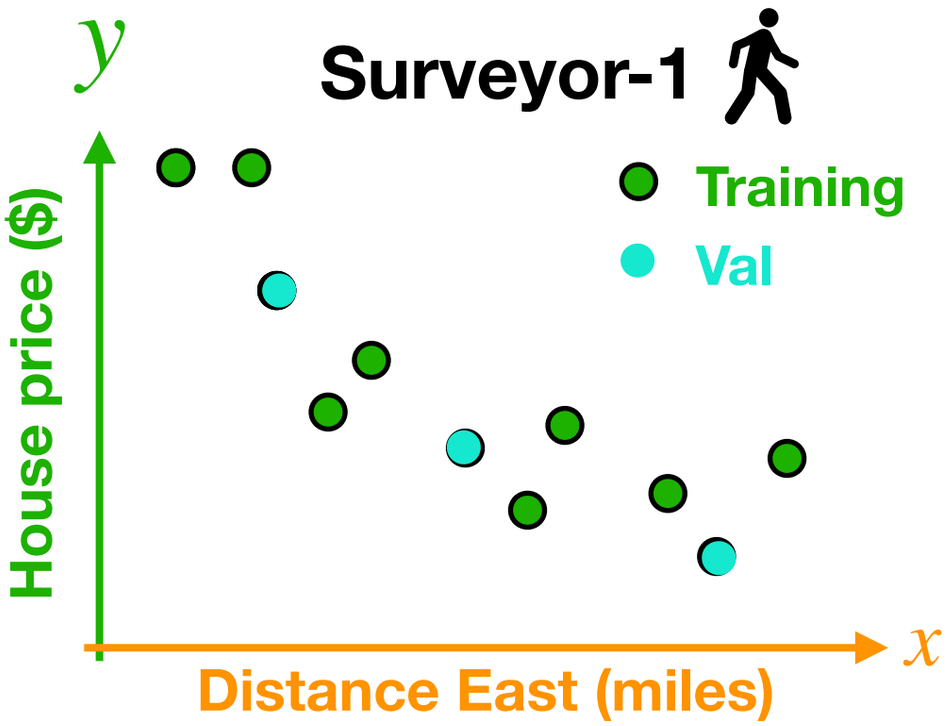
★**Scaling up: developing a model that overfits**

★**Regularizing your model and tuning your hyper-parameters**

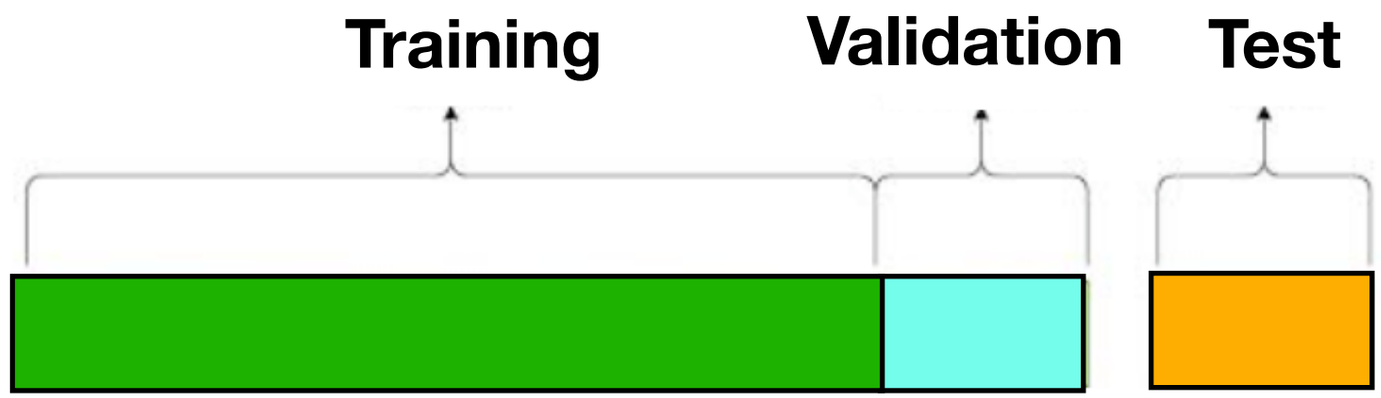
- **Choosing an optimizer, learning rate, and other hyper-parameters**

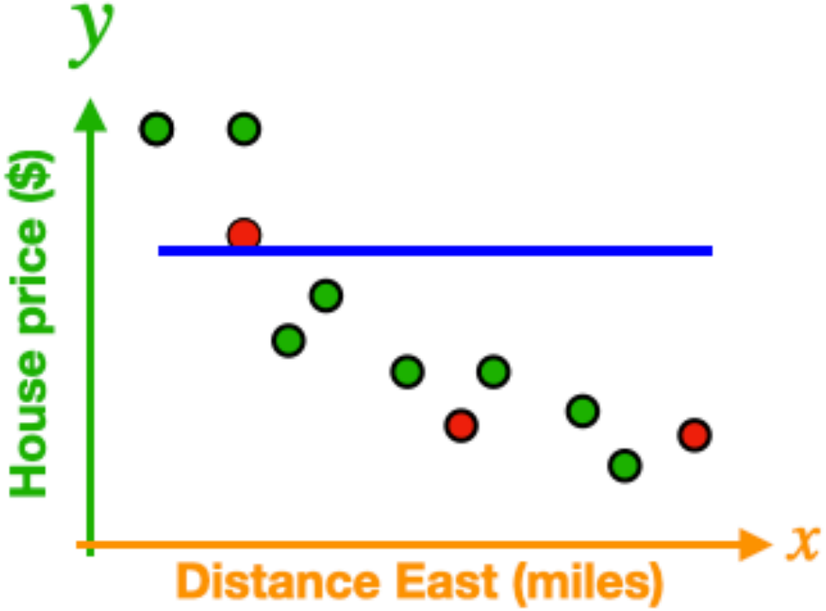


Partition your data

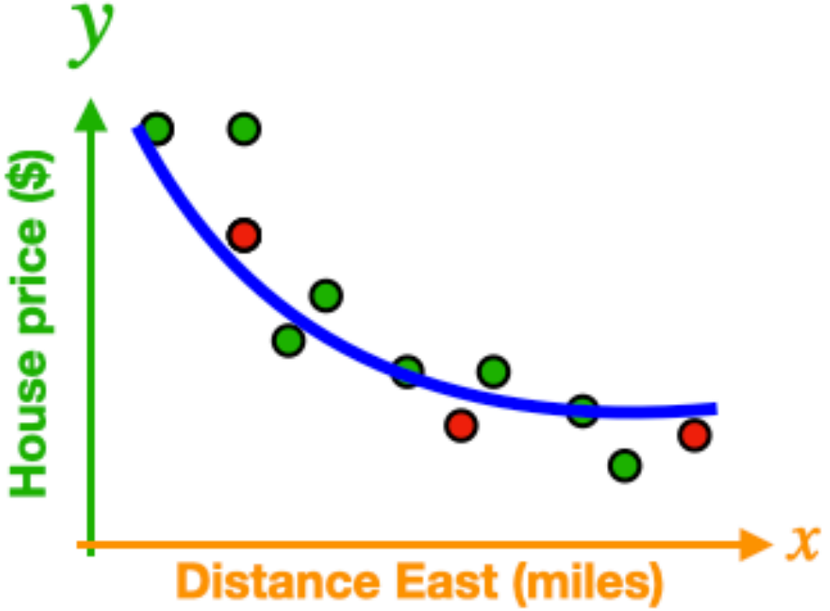


1) Partition data

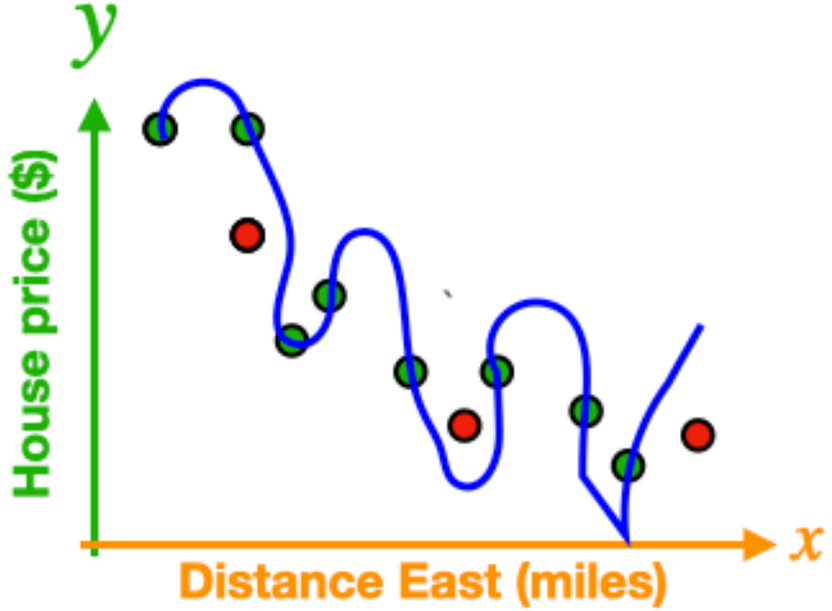




under-fitting
model complexity is too low



optimal
correct level of model complexity

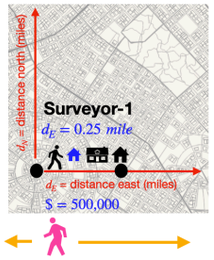


over-fitting
model complexity is too high

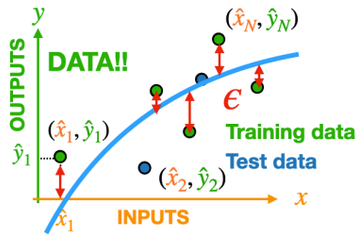
Common ML workflow summary

Source and recommended reading: Chollet p111-p116

- **Defining the problem and assembling a dataset**



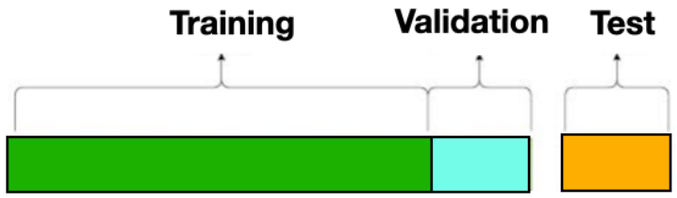
- **Choosing a measure of success**
  - **Choosing the accuracy/error metrics and a loss function**

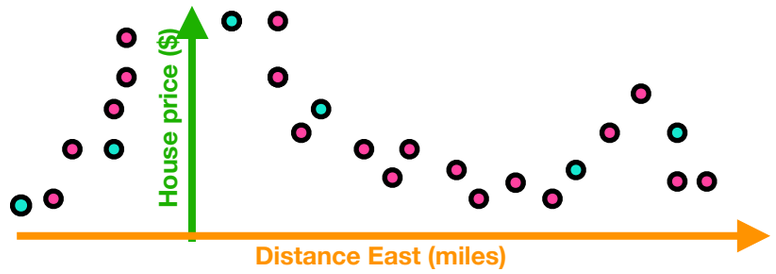


RMSE:

$$\epsilon(\mathbf{p}) = \left(\frac{\sum_i^N \left(y(\hat{x}_i \,|\, \mathbf{p}) - \hat{y}_i \right)^2}{N} \right)^{\frac{1}{2}}$$

- **Deciding on an evaluation protocol**





Train

Test

- **Preparing your data**

  Makes it easier to choice parameter initial conditions (IC)

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.25 \\ 0.30 \\ \vdots \\ 3.14 \end{pmatrix} \quad x_i \to \tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x} \approx [-1.0\,,1.0]$$

$\sigma_x \;\; \mu_x$

$$\hat{\mathbf{y}} = \begin{pmatrix} 500{,}000 \\ 410{,}000 \\ \vdots \\ 210{,}000 \end{pmatrix} \quad y_i \to \tilde{y}_i = \frac{y_i - \mu_y}{\sigma_y} \approx [-1.0\,,1.0]$$

$\sigma_y \;\; \mu_y$

- **Developing a model that does better than some baseline**

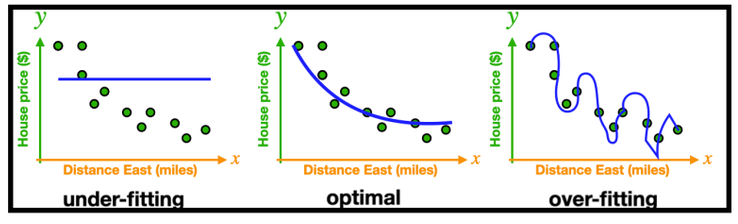
  Select a model

$$M(x \,|\, \mathbf{p}) = \text{Model} = \text{Functional form} + \text{Parameterization}$$

$$M(x \,|\, \mathbf{p}) = p_1 e^{-\left(\frac{x-p_2}{p_3}\right)^2} + p_4 e^{-\left(\frac{x-p_5}{p_6}\right)^2} + p_7$$

$$\mathbf{p} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)$$

- **Choosing an optimizer, learning rate, and other hyper-parameters**



★**Scaling up: developing a model that overfits**

★**Regularizing your model and tuning your hyper-parameters**

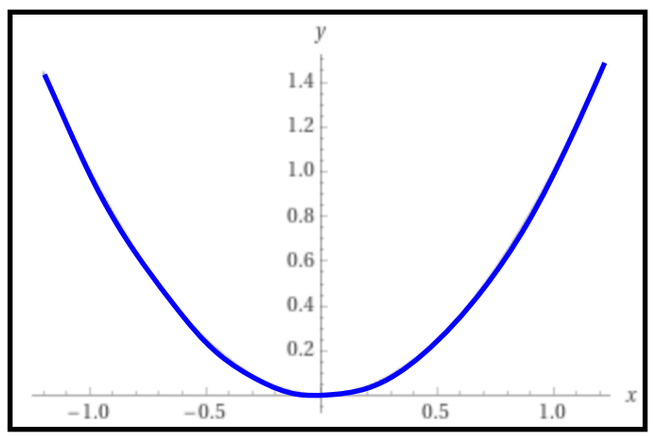
Choosing a model

MODEL = Functional form + Parameterization $\quad y = M(x \mid \mathbf{p})$

(Parametric) (underlying shape) (stretching and shifting)

Surveyor-1

(quadratic)



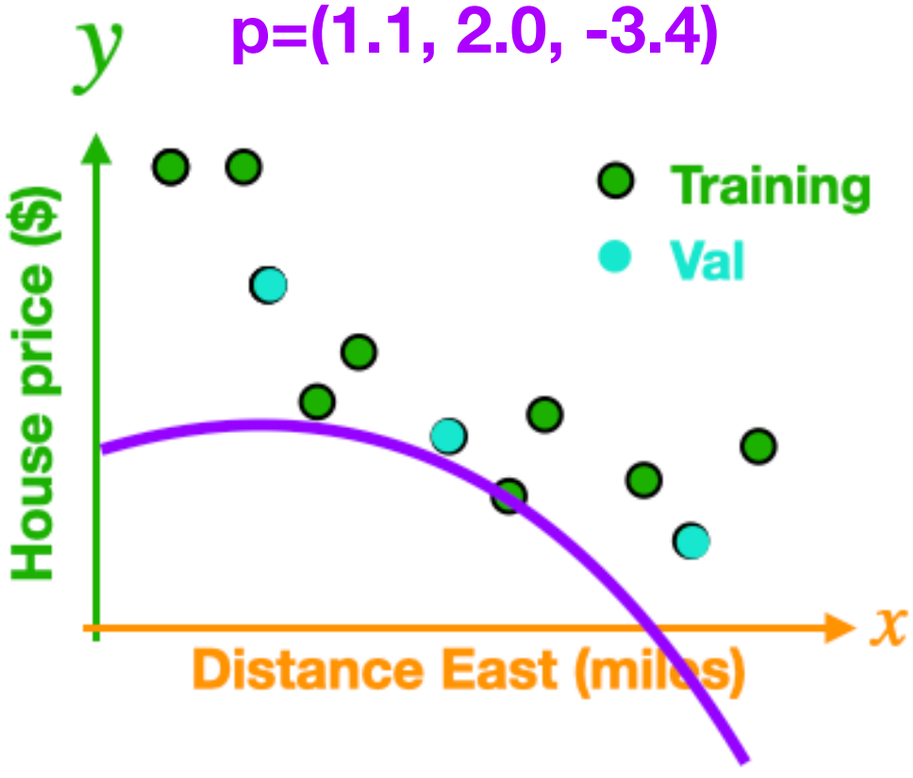
$$M(x \mid \mathbf{p}) = p_0 + p_1 x + p_2 x^2$$

$$\mathbf{p} = (p_0, p_1, p_2)$$

We have three parameters, i.e. "knobs", to "turn" to fit the data (i.e degrees of freedom)

Bad parameterization

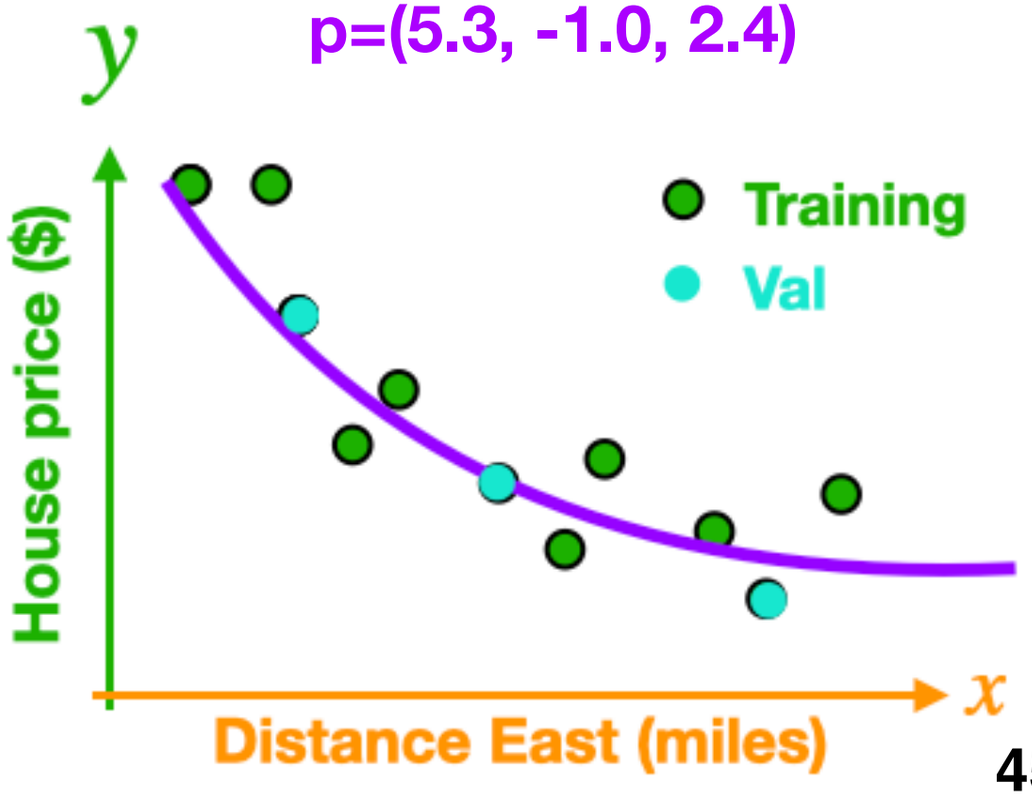
p=(1.1, 2.0, -3.4)



Goal: Find the three PARAMETER numbers $\mathbf{p} = (p_0, p_1, p_2)$ which provide the "best" fit of the data

Good parameterization

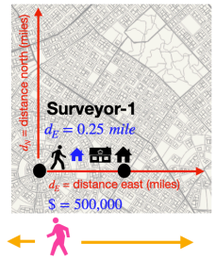
p=(5.3, -1.0, 2.4)



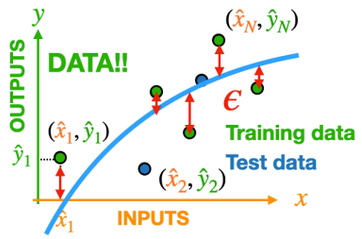
Common ML workflow summary

Source and recommended reading: Chollet p111-p116

- **Defining the problem and assembling a dataset**



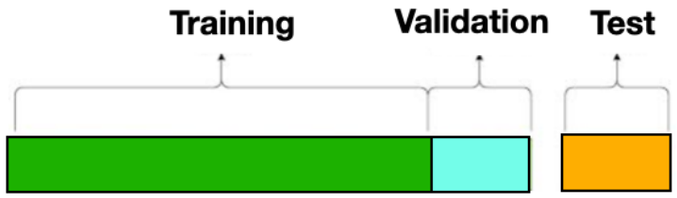
- **Choosing a measure of success**
  - **Choosing the accuracy/error metrics and a loss function**

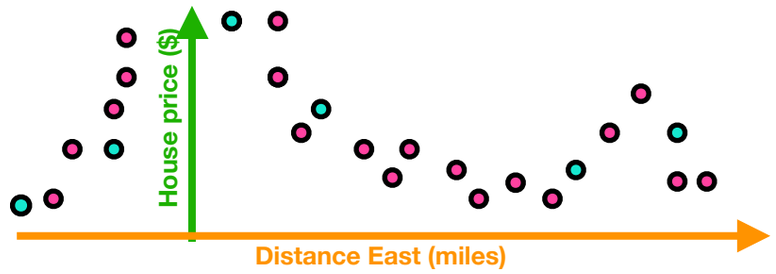


RMSE:

$$\epsilon(\mathbf{p}) = \left(\frac{\sum_i^N \left(y(\hat{x}_i \,|\, \mathbf{p}) - \hat{y}_i \right)^2}{N} \right)^{\frac{1}{2}}$$

- **Deciding on an evaluation protocol**





- **Preparing your data**
  Makes it easier to choice parameter initial conditions (IC)

$$\hat{\mathbf{x}} = \begin{pmatrix} 0.25 \\ 0.30 \\ \vdots \\ 3.14 \end{pmatrix} \quad x_i \to \tilde{x}_i = \frac{x_i - \mu_x}{\sigma_x} \approx [-1.0\,,\,1.0] \qquad \hat{\mathbf{y}} = \begin{pmatrix} 500{,}000 \\ 410{,}000 \\ \vdots \\ 210{,}000 \end{pmatrix} \quad y_i \to \tilde{y}_i = \frac{y_i - \mu_y}{\sigma_y} \approx [-1.0\,,\,1.0]$$

$\sigma_x \;\; \mu_x$ $\qquad$ $\sigma_y \;\; \mu_y$

- **Developing a model that does better than some baseline**

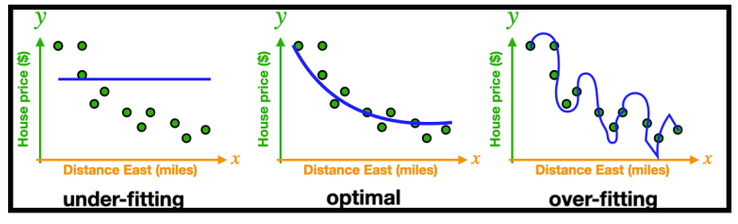
Select a model

$$M(x \,|\, \mathbf{p}) = \text{Model} = \text{Functional form} + \text{Parameterization}$$

$$M(x \,|\, \mathbf{p}) = p_1 e^{-\left(\frac{x - p_2}{p_3}\right)^2} + p_4 e^{-\left(\frac{x - p_5}{p_6}\right)^2} + p_7$$

$$\mathbf{p} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7)$$

- **Choosing an optimizer, learning rate, and other hyper-parameters**



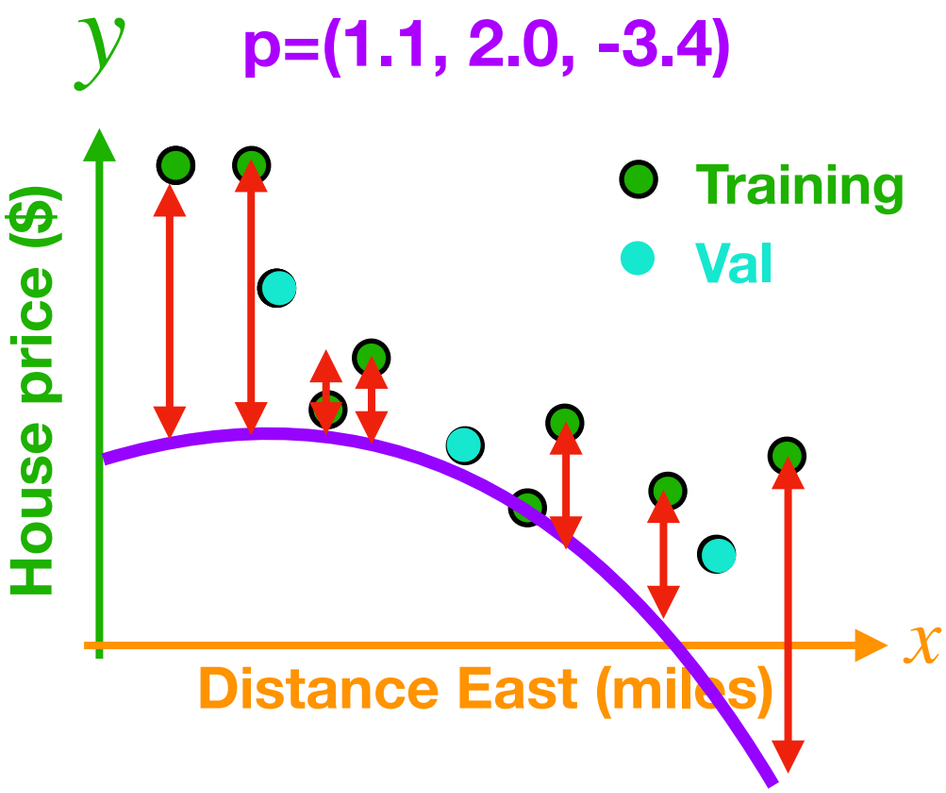
★**Scaling up: developing a model that overfits**

★**Regularizing your model and tuning your hyper-parameters**

Loss function

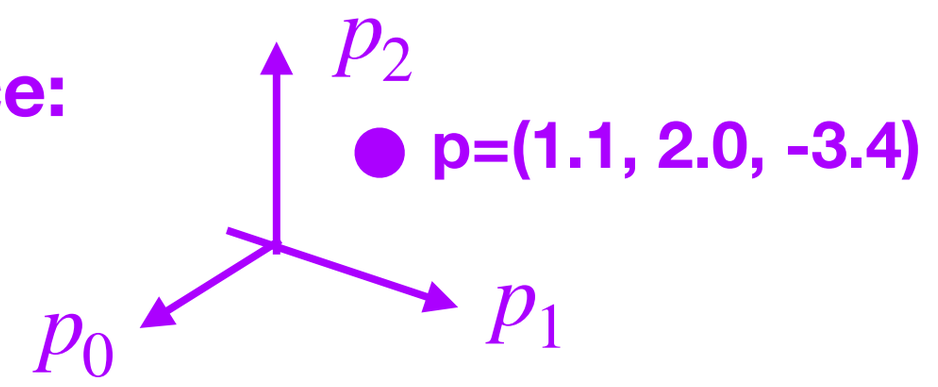
Model: $M(x \mid \mathbf{p}) = p_0 + p_1 x + p_2 x^2$

Bad parameterization



Parameter space:

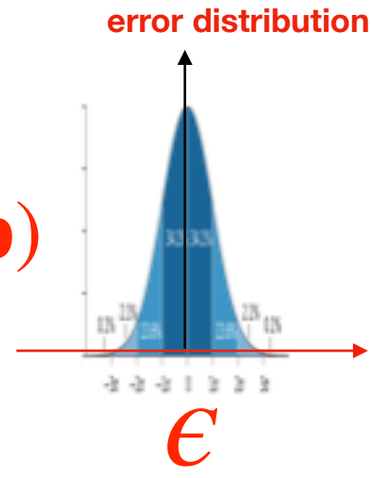
$$\mathbf{p} = (p_0, p_1, p_2)$$



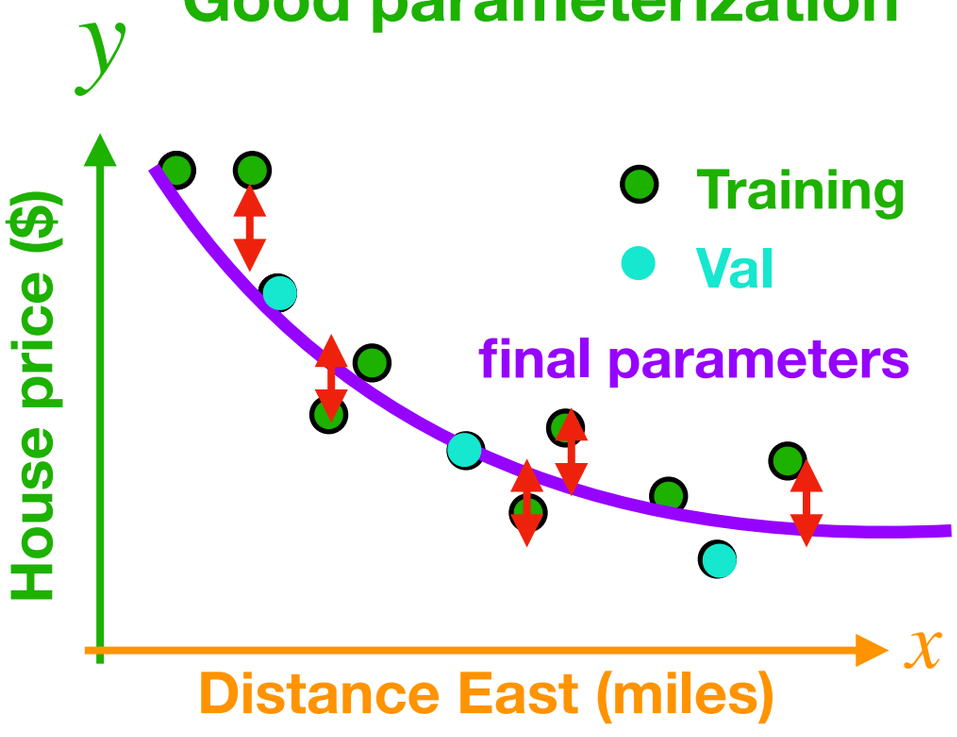
Space of all possible quadratic models

Vector of errors (residuals)

$$\epsilon_i(\mathbf{p}) = y_{data} - y_{pred} = y_i - M(x_i \mid \mathbf{p})$$



Good parameterization



Define an SCALAR error metric (loss function)

Loss function: $L_{train}(\mathbf{p})$

RMSE:

$$L(\mathbf{p}) = \left(\frac{\sum_i^N M(x \mid \mathbf{p}) - \hat{y}_i)^2}{N} \right)^{\frac{1}{2}}$$

AKA: Objective function

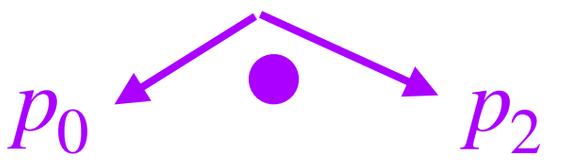
Loss surface-1

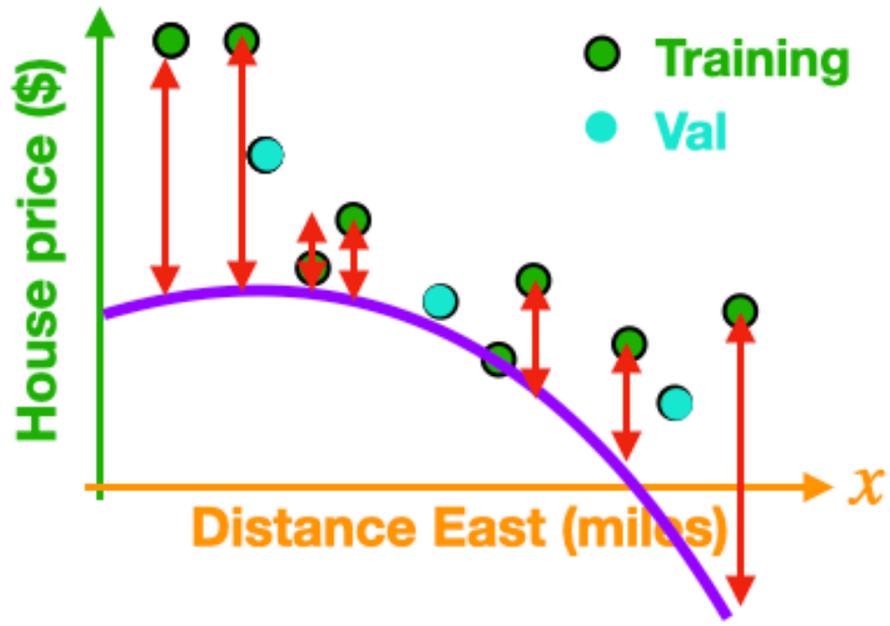
simplified model model:

$$M(x \mid \mathbf{p}) = p_0 + p_2 x^2$$

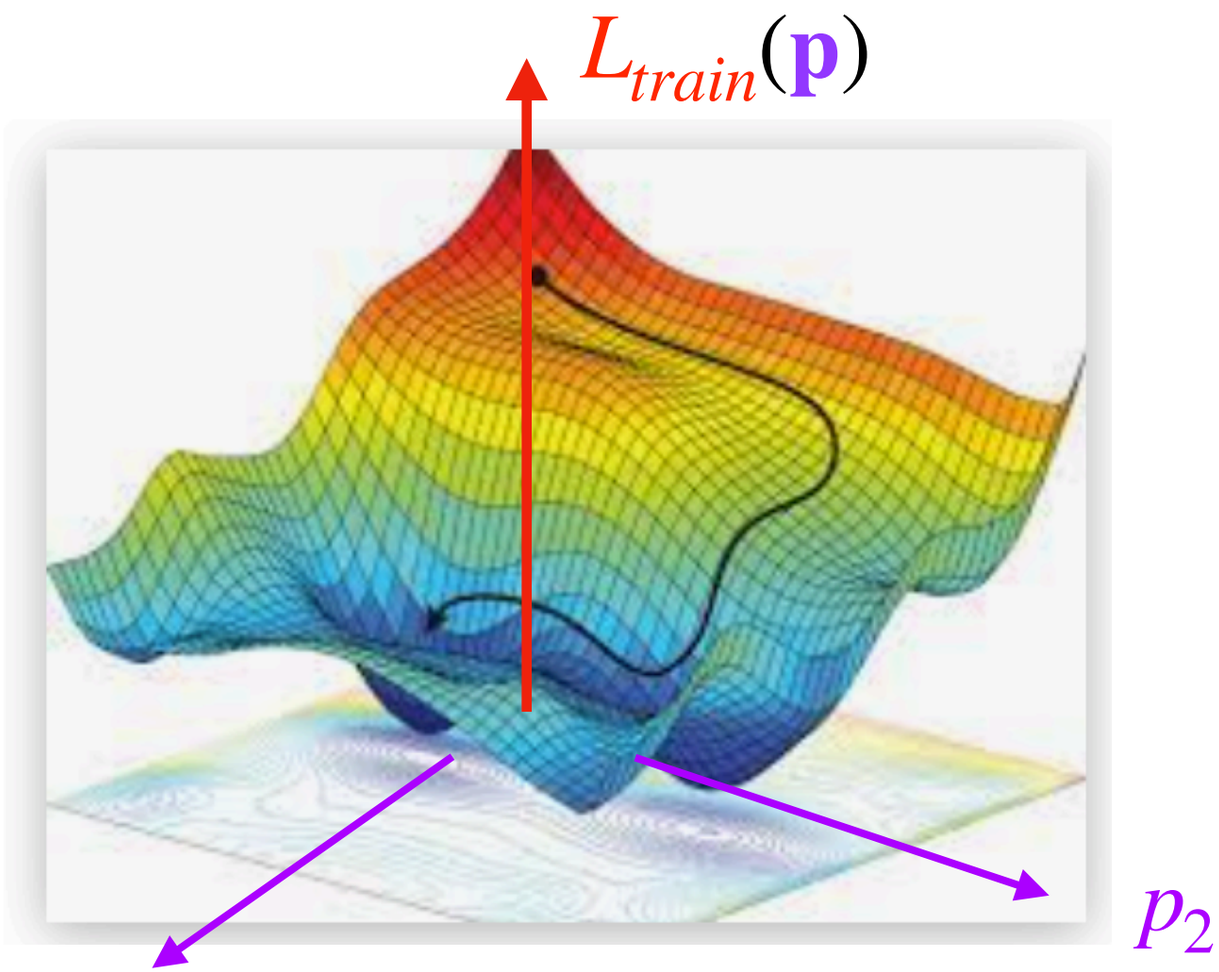
Parameter space:

$$\mathbf{p} = (p_0, p_1)$$





$$\epsilon_i(\mathbf{p}) = y_{data} - y_{pred} = y_i - M(x_i \mid \mathbf{p})$$



Loss surface

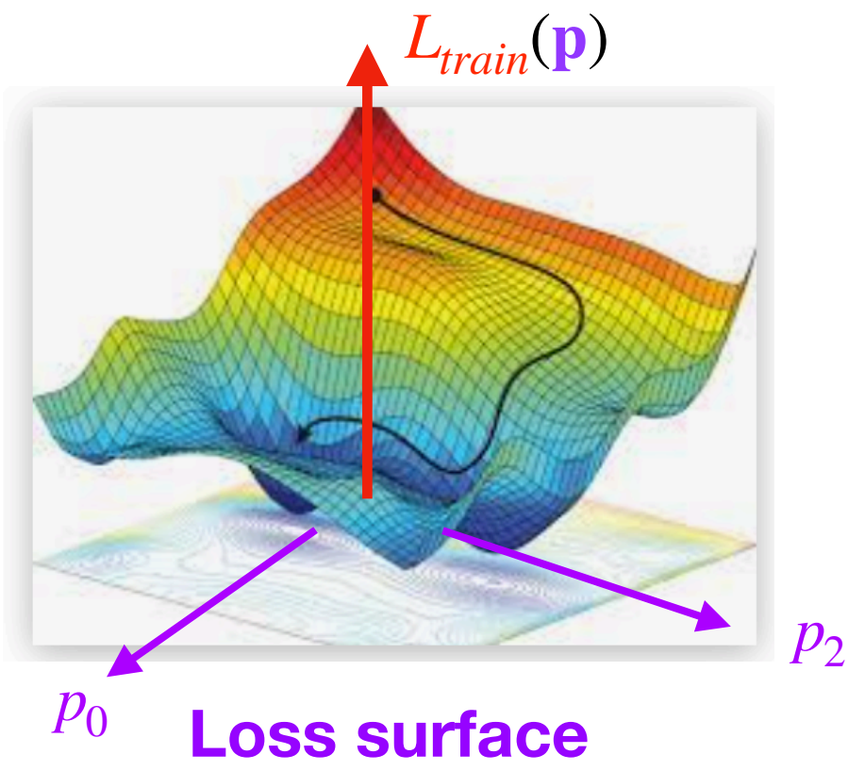
Loss function:

RMSE:

$$L(\mathbf{p}) = \left(\frac{\sum_i^N M(x \mid \mathbf{p}) - \hat{y}_i)^2}{N} \right)^{\frac{1}{2}}$$

AKA: Objective function

Training Numerical optimization problem to find best parameters (best fit)



Loss surface

simplified model model:

$$M(x \mid \mathbf{p}) = p_0 + p_2 x^2$$

Parameter space: $\mathbf{p} = (p_0, p_1)$

Goal find parameters that are local or global minima of the loss surface. The lower the minima the better the model fit

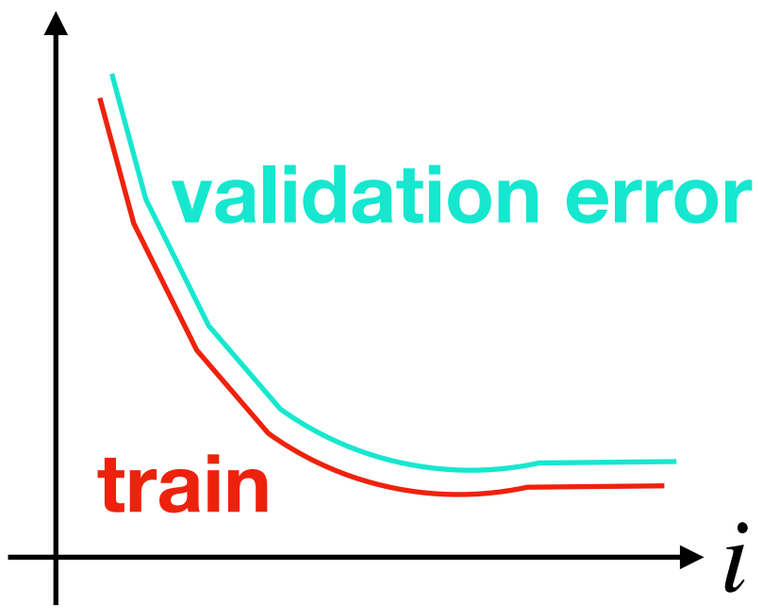
Minimize loss surface with gradient decent $\mathbf{p}_{n+1} = \mathbf{p}_n - \lambda \nabla_w \epsilon(\mathbf{p}_n)$ $\nabla_p \epsilon = \left(\frac{\partial \epsilon}{\partial p_0}, \frac{\partial \epsilon}{\partial p_2} \right)$

Monitor loss+metrics:

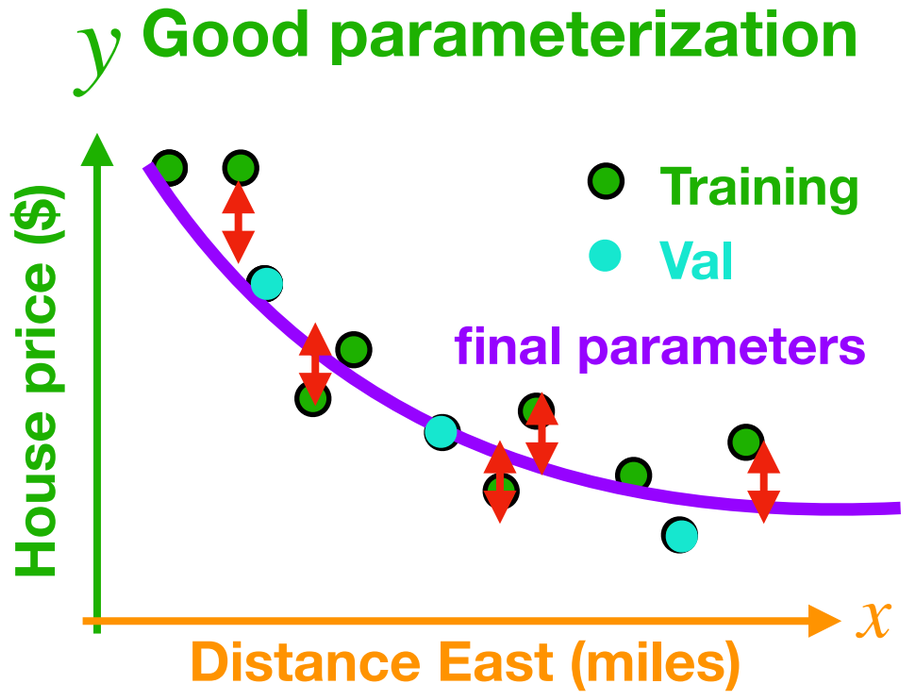
$\epsilon_{test}(\mathbf{p})$

$MAE_{train}(\mathbf{p})$

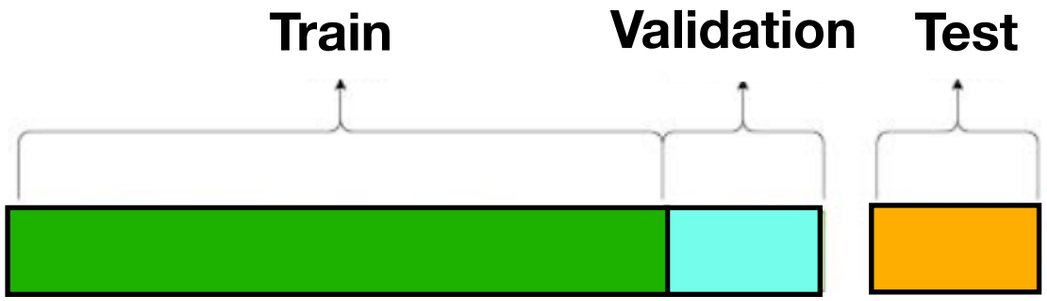
Metrics are monitored during fitting
But **not** seen by optimizer

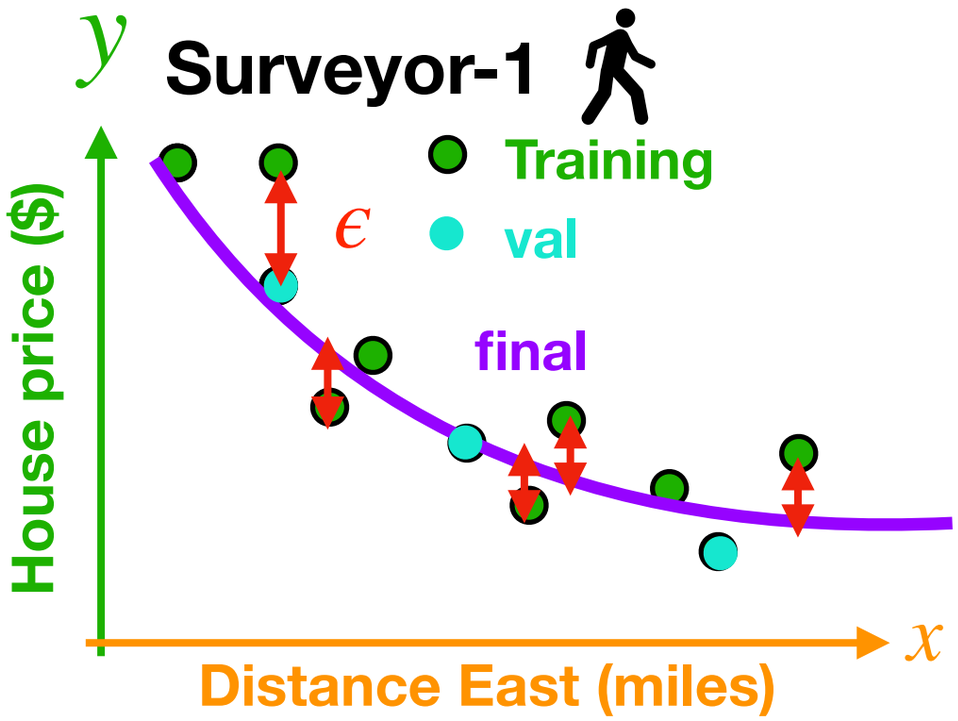


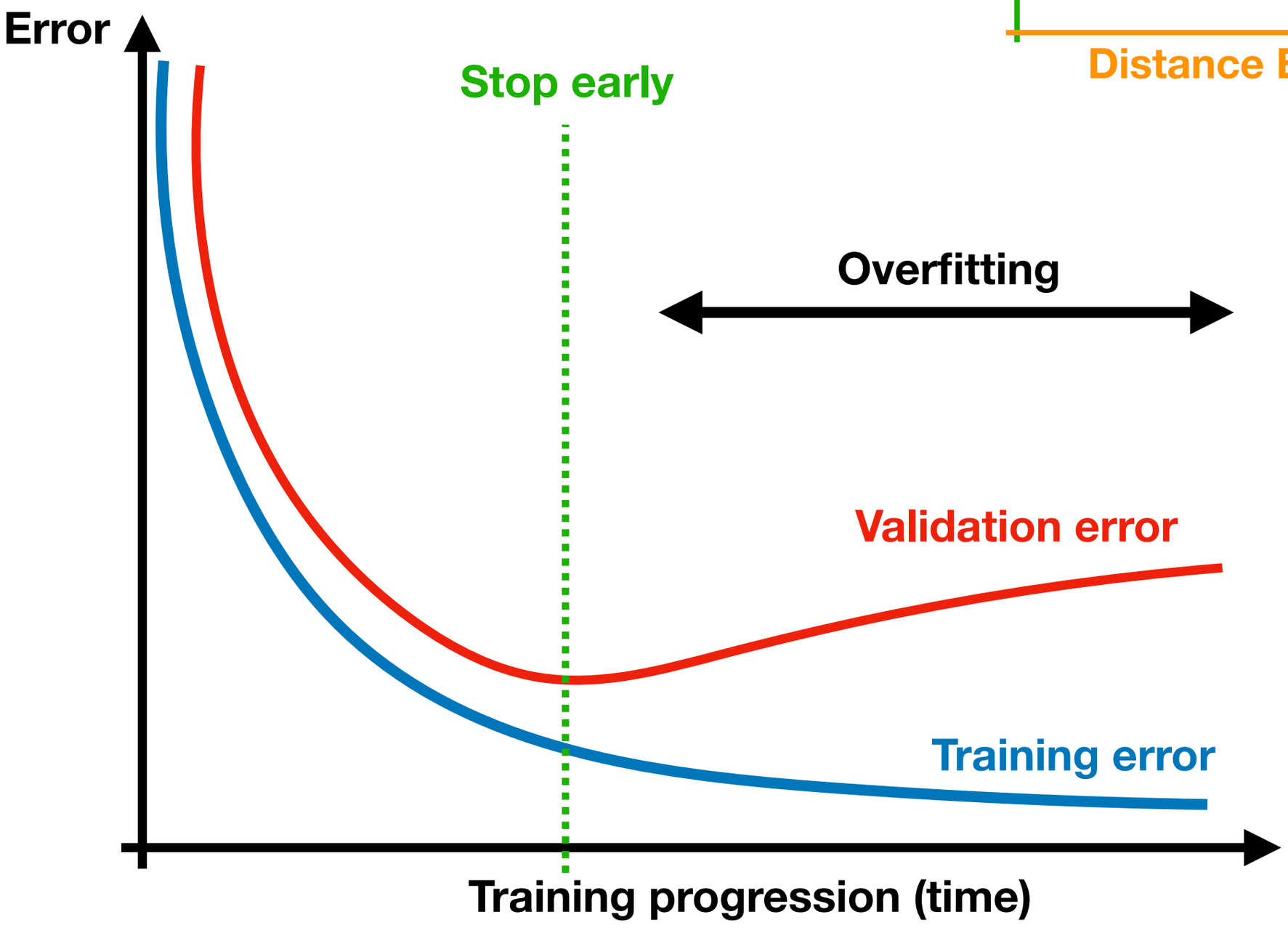
Optimizer iterations



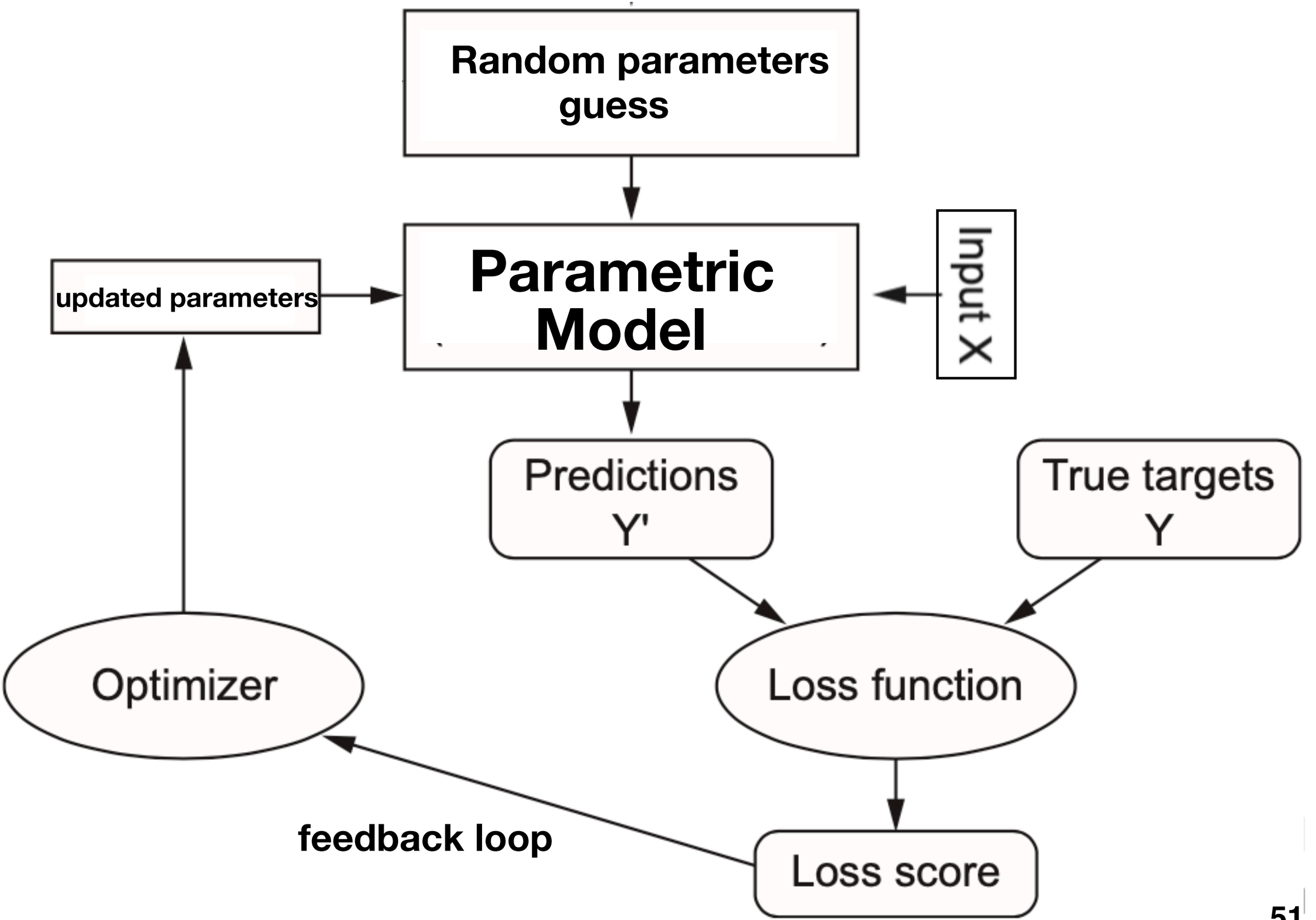
Moitoring overfitting





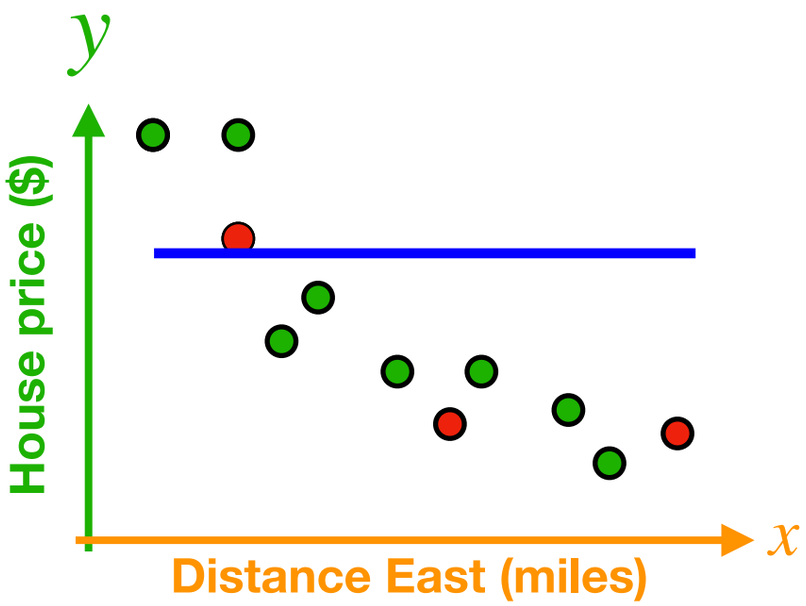


Model training diagram



OVERFITTING

Interpolation concerns



under-fitting

model complexity is too low

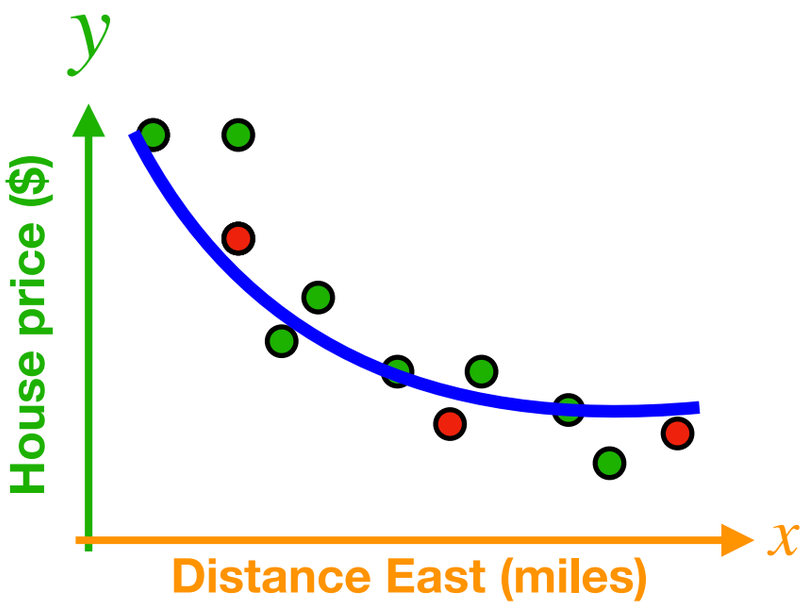
High training error
High test error

high bias

low variance

The inability for a machine learning method (like linear regression) to capture the true relationship is called **bias**.

(Under parameterized)



optimal

correct level of model complexity

Training and Test error are equally low

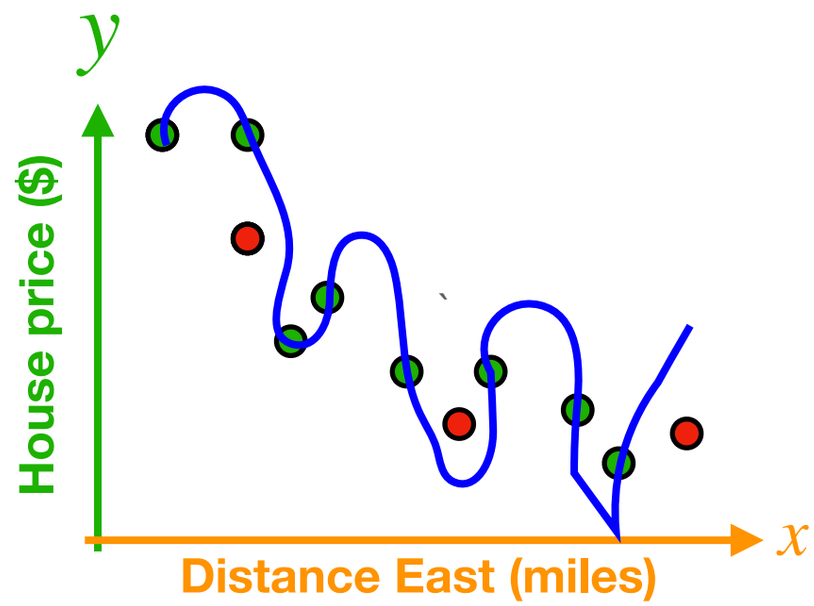
low bias

low variance

In machine learning, the ideal algorithm has **low bias** and can accurately model the true relationship...

...and it has **low variability**, by producing consistent predictions across different datasets.

(Properly parameterized)



over-fitting

model complexity is too high

Very Low training error
Very high test error

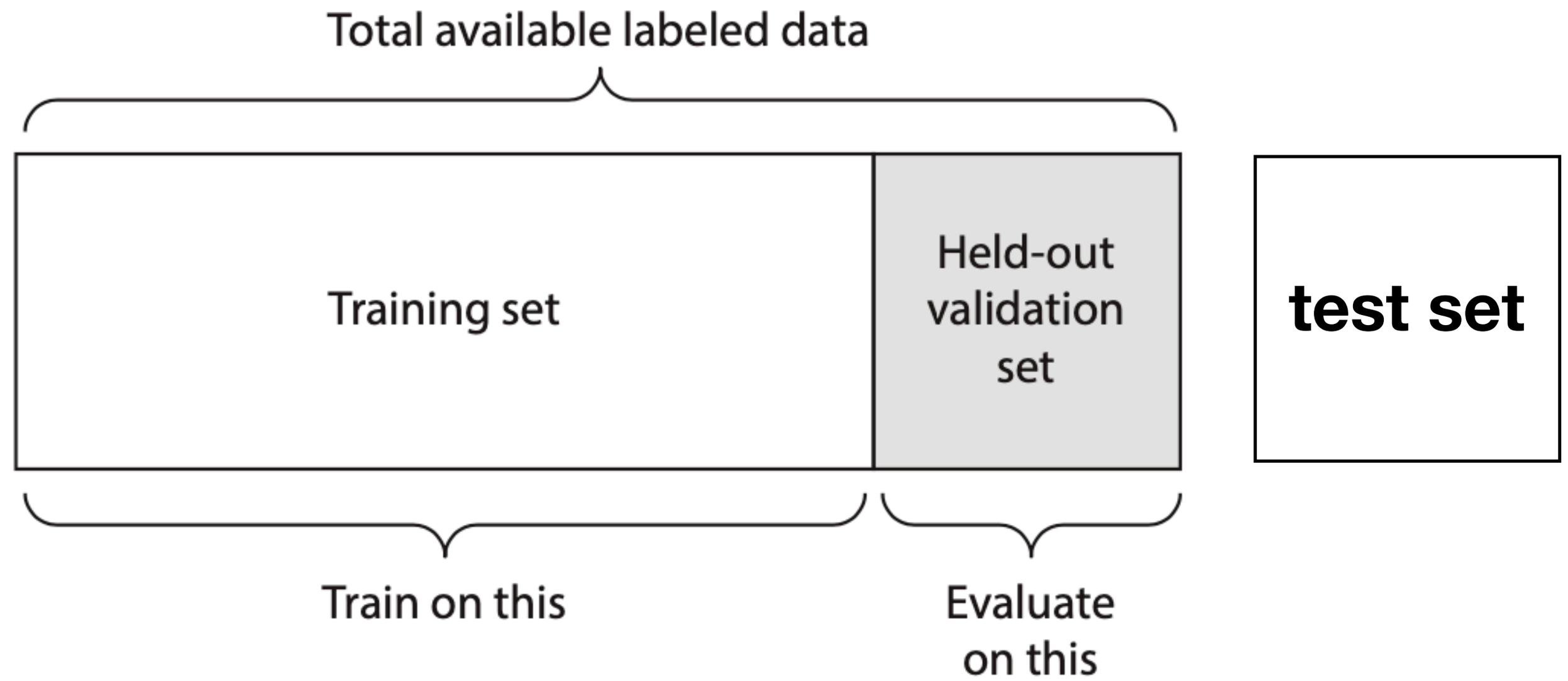
low bias

high variance

In Machine Learning lingo, the difference in fits between data sets is called **Variance**.

(Over parameterized)

Hold out validation



Each is a random subset of total dataset

Chollet, Francois. *Deep learning with Python*. Manning Publications Co., 2018.

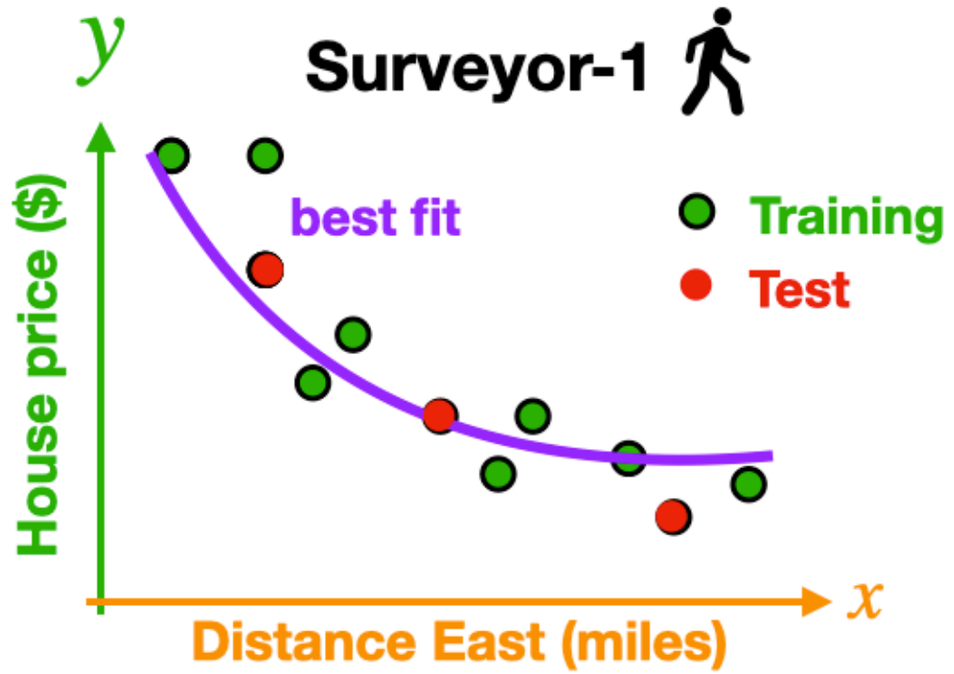
General regression concerns

Extrapolation: Predictions "outside"
(AKA transferability)

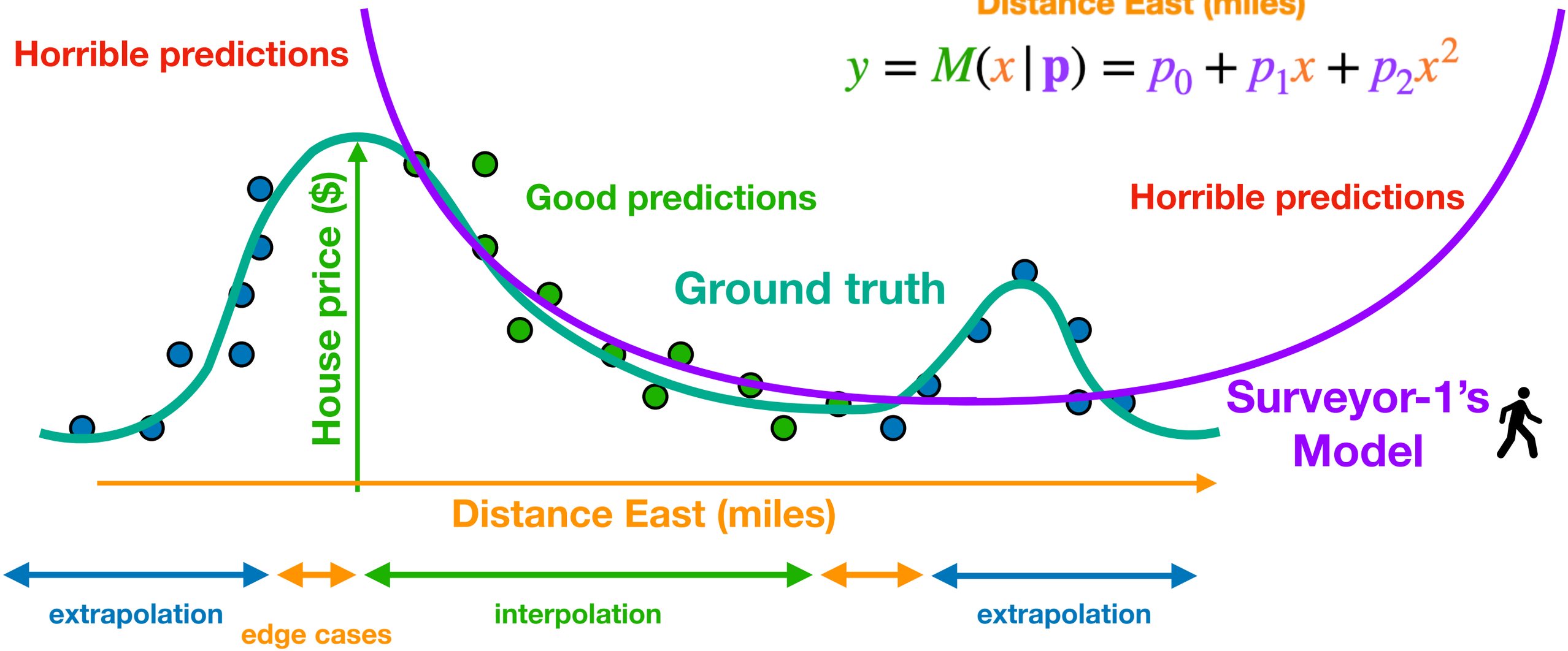
A model's ability to extrapolate depends on;

1) **Quality of the descriptor (inputs)**
2) **Amount and distribution data**
3) **Whether functional form of the model is similar to the ground truth**

To improve transferablity --> MORE DATA or better model



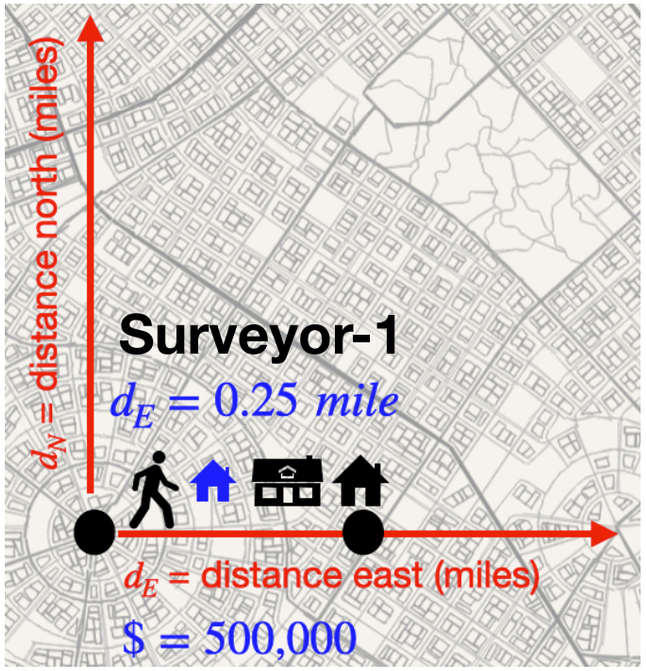
$$y = M(x \mid \mathbf{p}) = p_0 + p_1 x + p_2 x^2$$



Interpolation: Predictions "Inside training region"

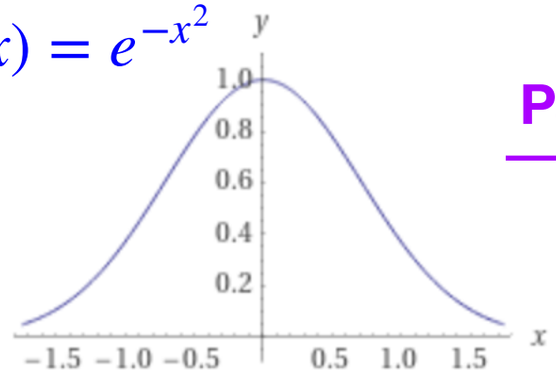
Most WELL TRAINED models can interpolate reasonably well

Surveyor-2 model



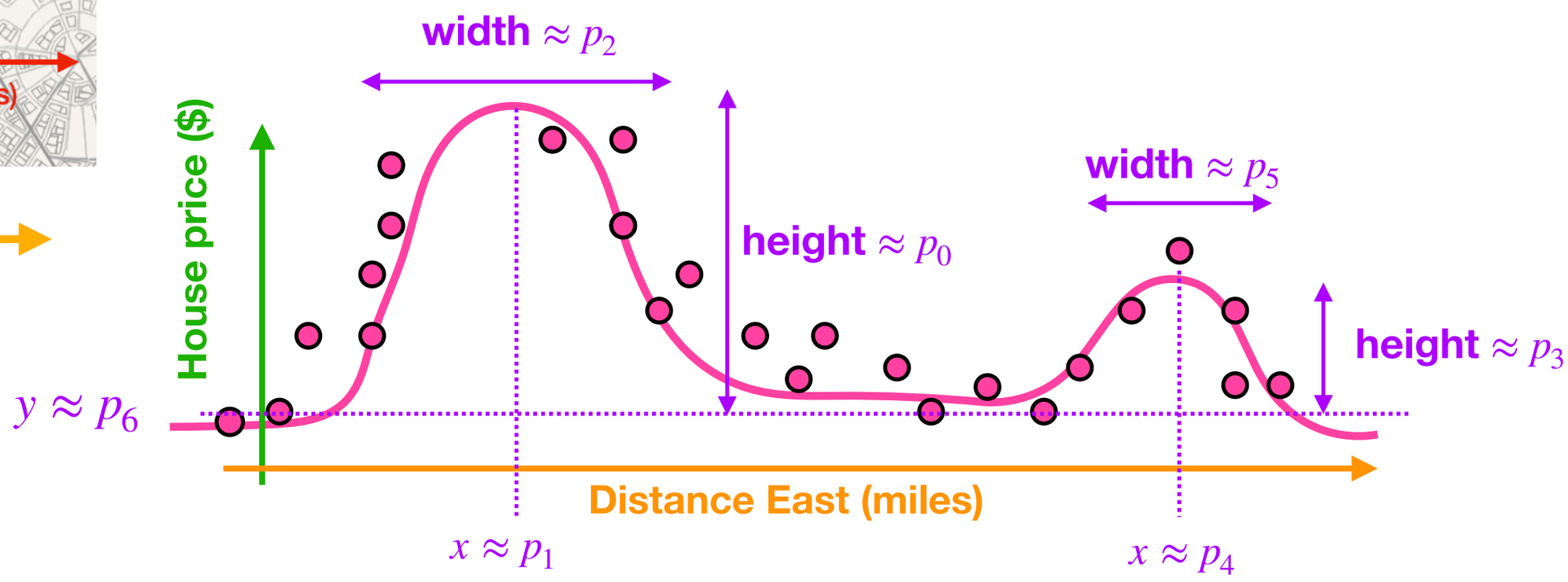
Parent function

$$f(x) = e^{-x^2}$$



Parent function

The purpose of the parameters is to stretch and bend the parent function to match "fit" the data



$$M(x \mid \mathbf{p}) = \text{Model} = \text{Functional form} + \text{Parameterization}$$

$$M(x \mid \mathbf{p}) = p_0 e^{-\left(\frac{x-p_1}{p_2}\right)^2} + p_3 e^{-\left(\frac{x-p_4}{p_5}\right)^2} + p_6$$

$$\mathbf{p} = (p_0, p_1, p_2, p_3, p_4, p_5, p_6)$$