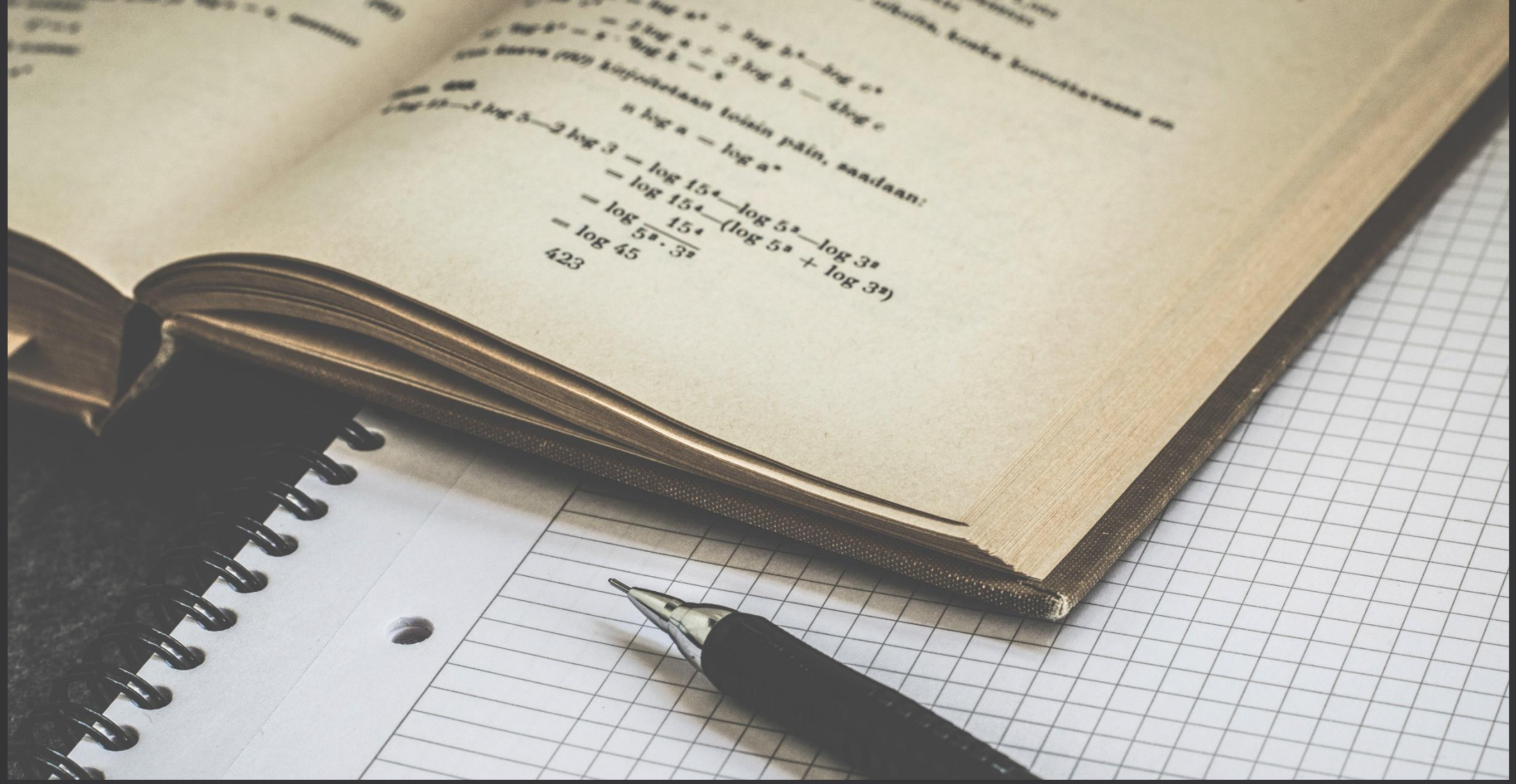


Photo by Lum3n from Pexels (retrieved 2023-04-07)

Introduction to Machine Learning

Machine Learning is a form of statistics

Photo by Tima Miroshnichenko from Pexels
(retrieved 2023-04-02)

Machine Learning is a form of statistics

Work with data

Machine Learning is a form of statistics

Use applied mathematics to understand the data

Machine Learning is a form of statistics

Measure your understanding quantitatively

Machine Learning is a form of **automated** statistics

Teach your computer to do it all for you

Photo by Tima Miroshnichenko from Pexels
(retrieved 2023-04-02)

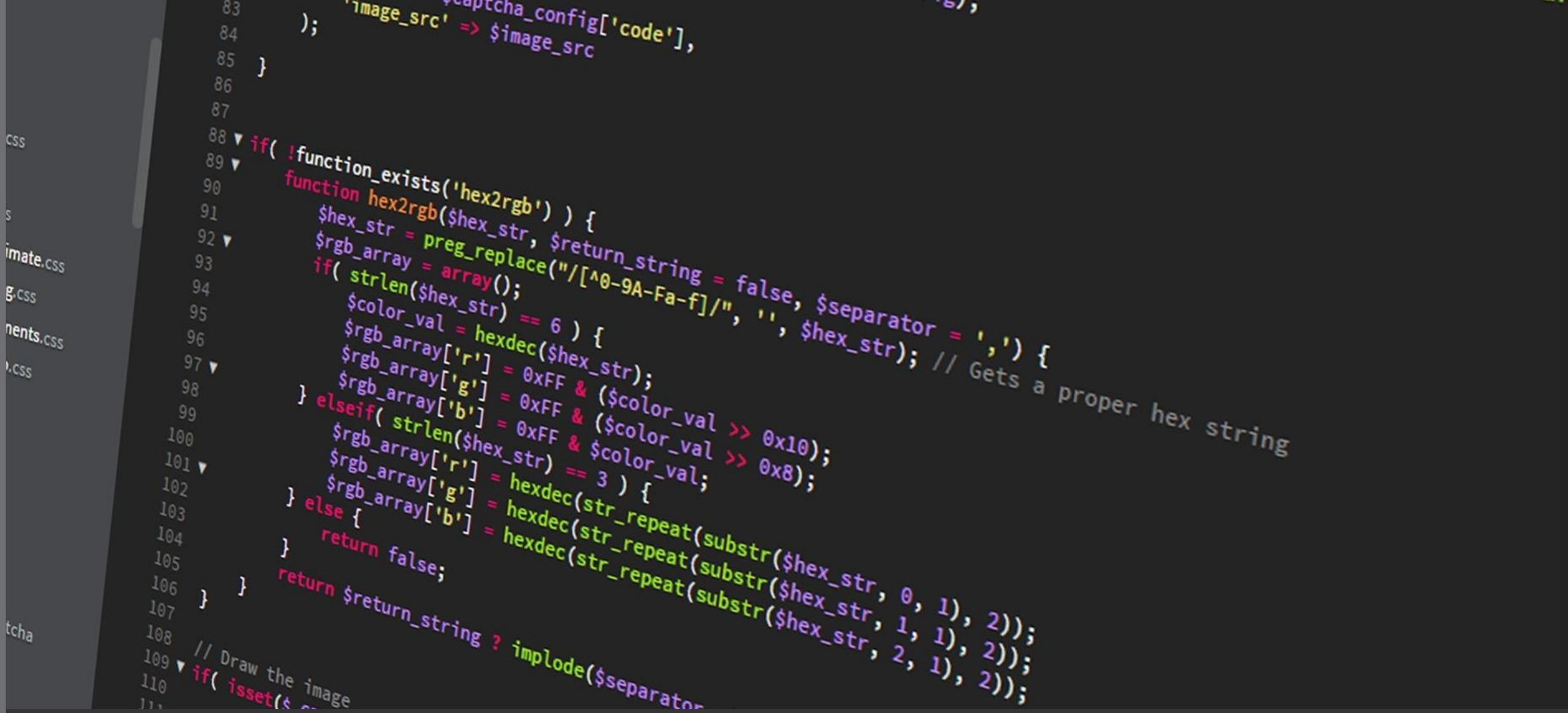


Photo by Pixabay from Pexels (retrieved 2023-04-07)

Introduction to Machine Learning

Setting up an ML and DS problem

Ingredients

Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Ingredients

1. Dataset X
2. Target / label data y

Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Ingredients

1. Dataset X
2. Target / label data y
3. ML model $\hat{y}(w)$
4. Parameters w

Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Ingredients

1. Dataset X
2. Target / label data y
3. ML model $\hat{y}(w)$
4. Parameters w
5. Error function $E(y, \hat{y}(w))$

Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Ingredients

1. Dataset X
2. Target / label data y
3. ML model $\hat{y}(w)$
4. Parameters w
5. Error function $E(y, \hat{y}(w))$

Instructions

1. Analyze X and y
2. Clean / process X and y

Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Ingredients

1. Dataset X
2. Target / label data y
3. ML model $\hat{y}(w)$
4. Parameters w
5. Error function $E(y, \hat{y}(w))$

Instructions

1. Analyze X and y
2. Clean / process X and y
3. Engineer features of X
4. Split X and y into train, test and validation sets

Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Ingredients

1. Dataset X
2. Target / label data y
3. ML model $\hat{y}(w)$
4. Parameters w
5. Error function $E(y, \hat{y}(w))$

Instructions

1. Analyze X and y
2. Clean / process X and y
3. Engineer features of X
4. Split X and y into train, test and validation sets
5. Set up and train $\hat{y}(w)$
6. Measure $E(y, \hat{y}(w))$ on train and test sets

Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Ingredients

1. Dataset X
2. Target / label data y
3. ML model $\hat{y}(w)$
4. Parameters w
5. Error function $E(y, \hat{y}(w))$

Instructions

1. Analyze X and y
2. Clean / process X and y
3. Engineer features of X
4. Split X and y into train, test and validation sets
5. Set up and train $\hat{y}(w)$
6. Measure $E(y, \hat{y}(w))$ on train and test sets
7. Tune hyperparameters of $\hat{y}(w)$
8. Best $\hat{y}(w)$ minimizes $E(y, \hat{y}(w))$

Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Ingredients

1. Dataset X
2. Target / label data y
3. ML model $\hat{y}(w)$
4. Parameters w
5. Error function $E(y, \hat{y}(w))$

Instructions

1. Analyze X and y
2. Clean / process X and y
3. Engineer features of X
4. Split X and y into train, test and validation sets
5. Set up and train $\hat{y}(w)$
6. Measure $E(y, \hat{y}(w))$ on train and test sets
7. Tune hyperparameters of $\hat{y}(w)$
8. Best $\hat{y}(w)$ minimizes $E(y, \hat{y}(w))$
9. Run final measurement on the validation set

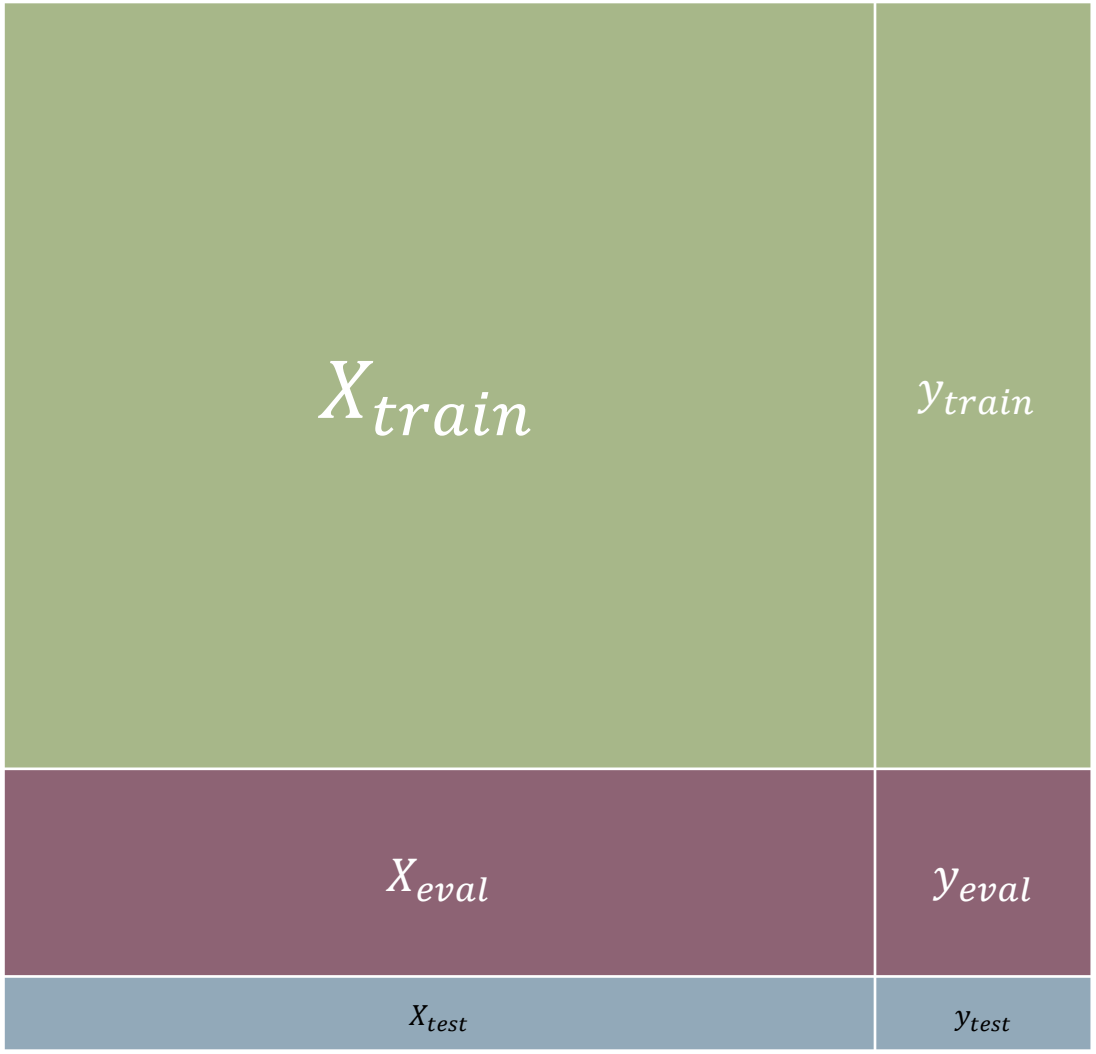
Photo by Calum Lewis on Unsplash (retrieved 2023-04-02)

Setting up an ML and DS problem

Describing vs Predicting

Setting up an ML and DS problem

Describing vs **Predicting**



Error function

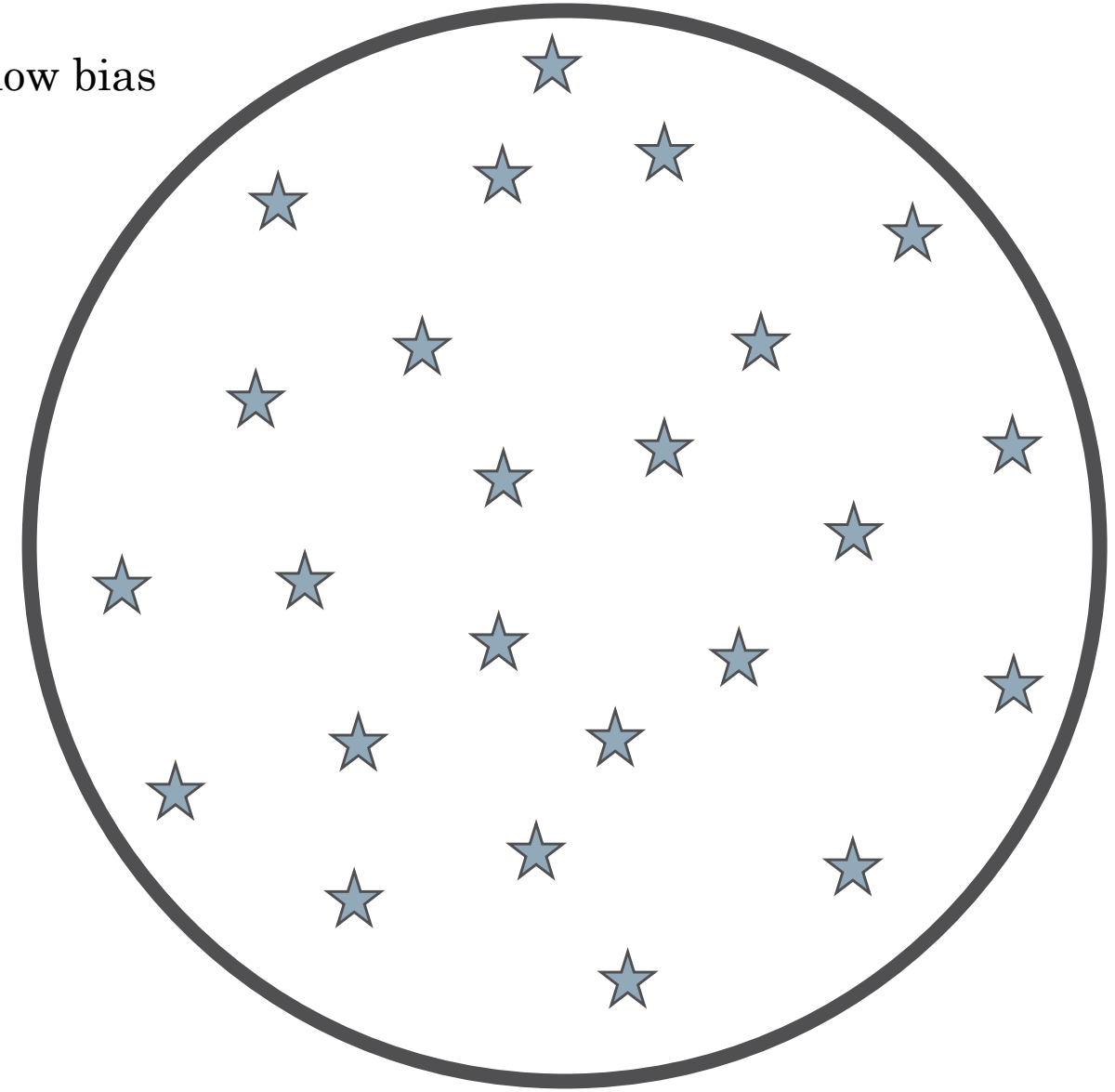
$$E = \sum_i |y_i - \hat{y}_i|$$

Overfitting vs underfitting

This is what **regularization** is for

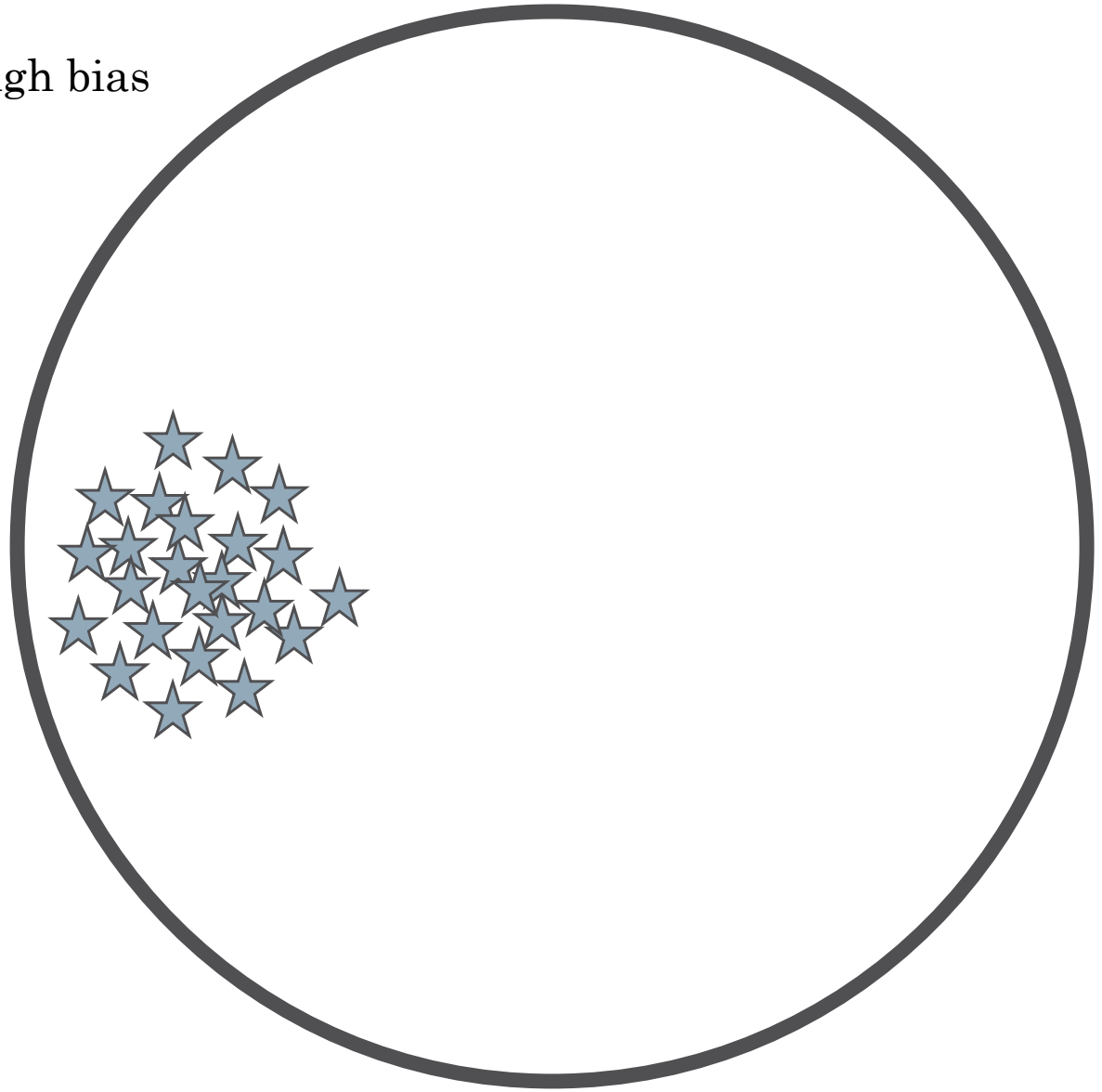
Bias vs variance

High variance, low bias



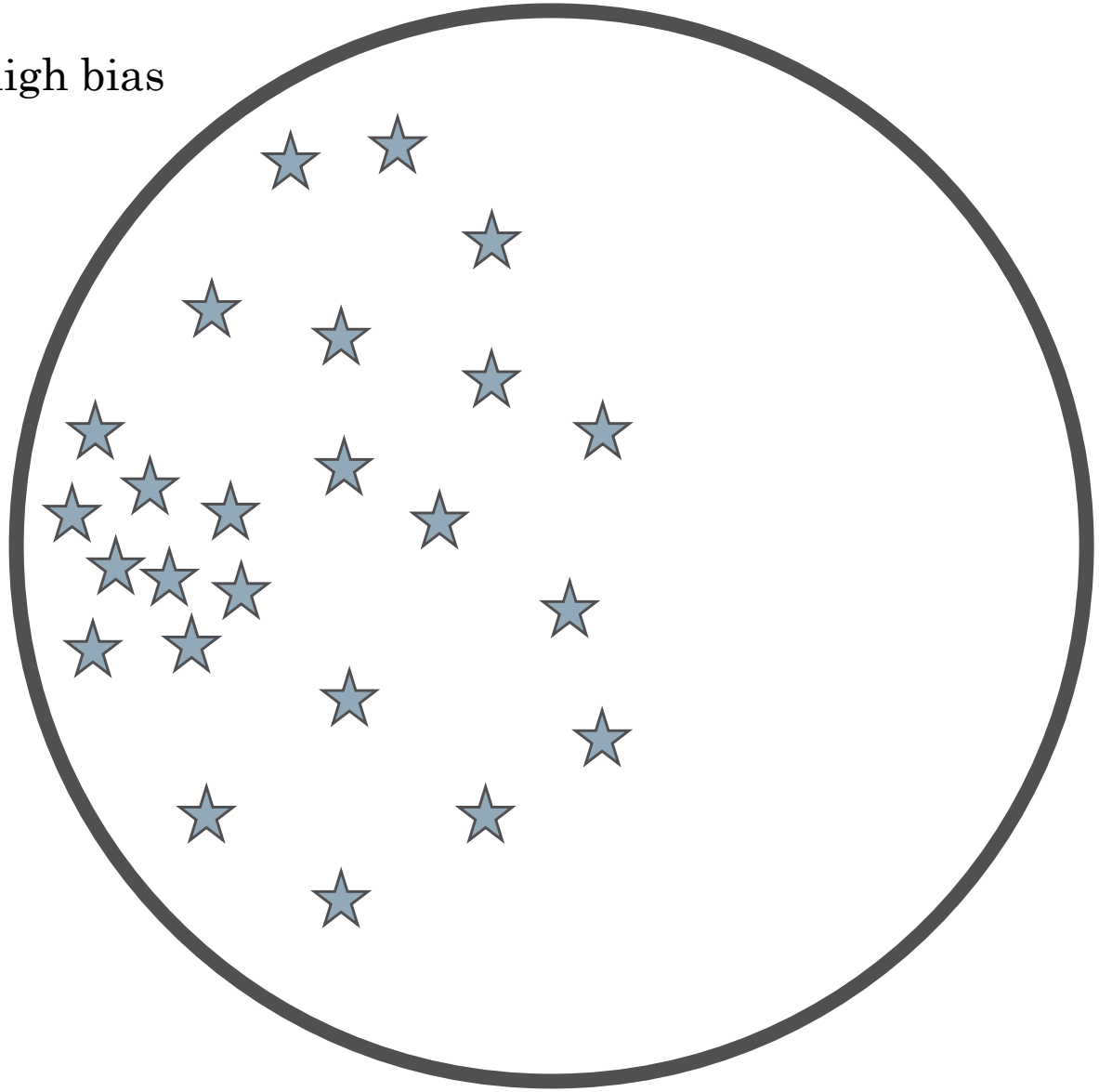
Bias vs variance

Low variance, high bias



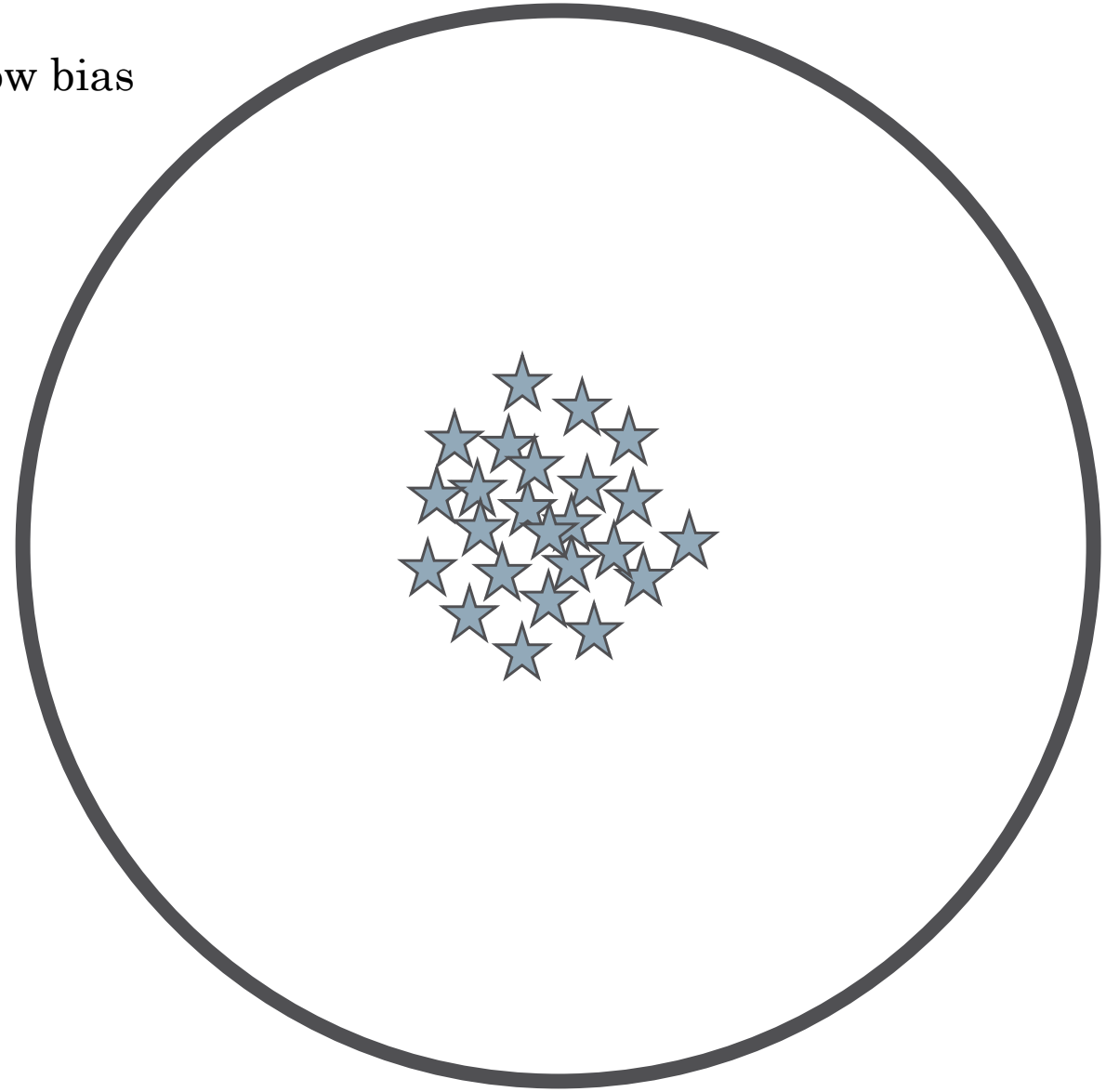
Bias vs variance

High variance, high bias



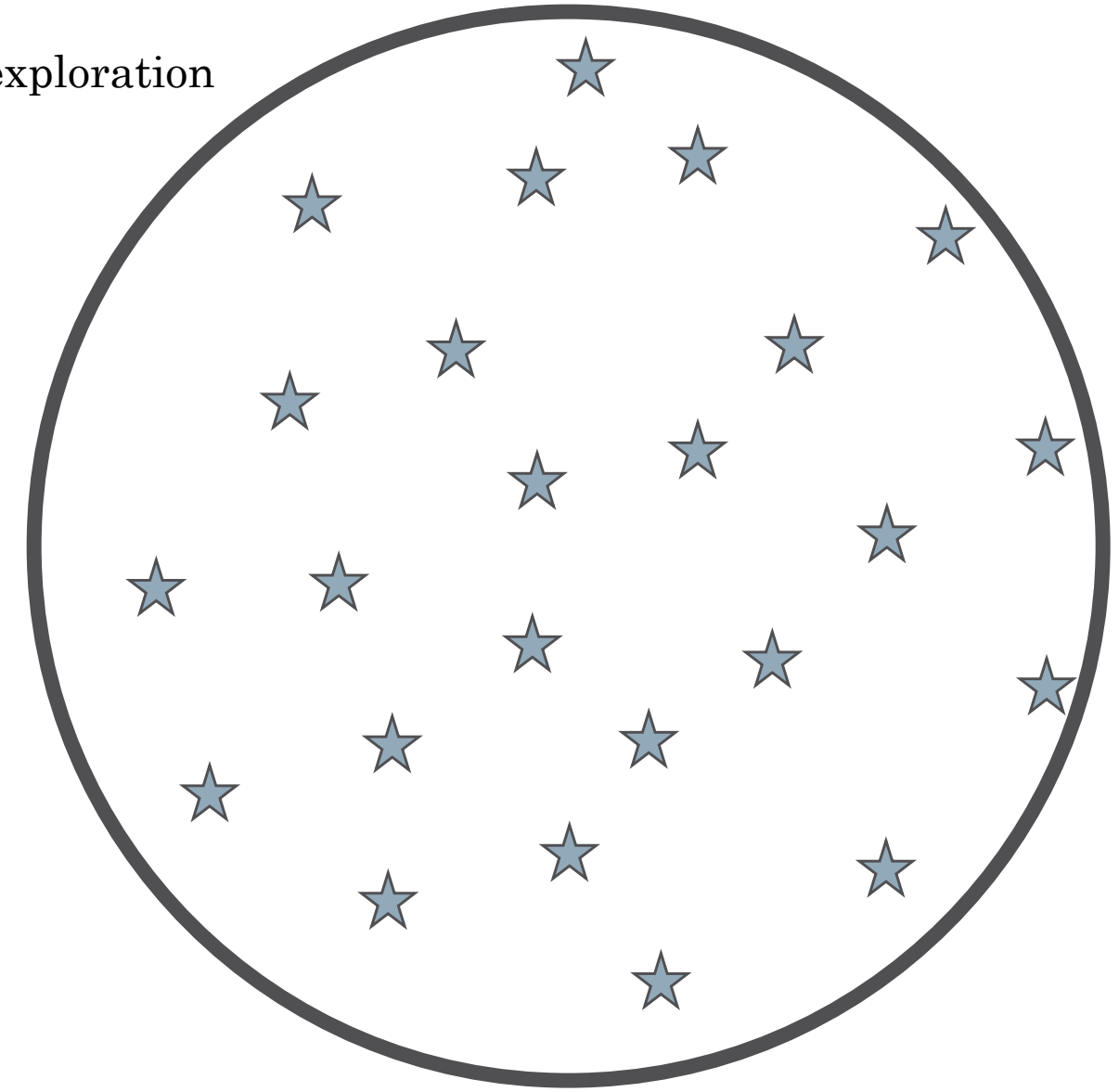
Bias vs variance

Low variance, low bias



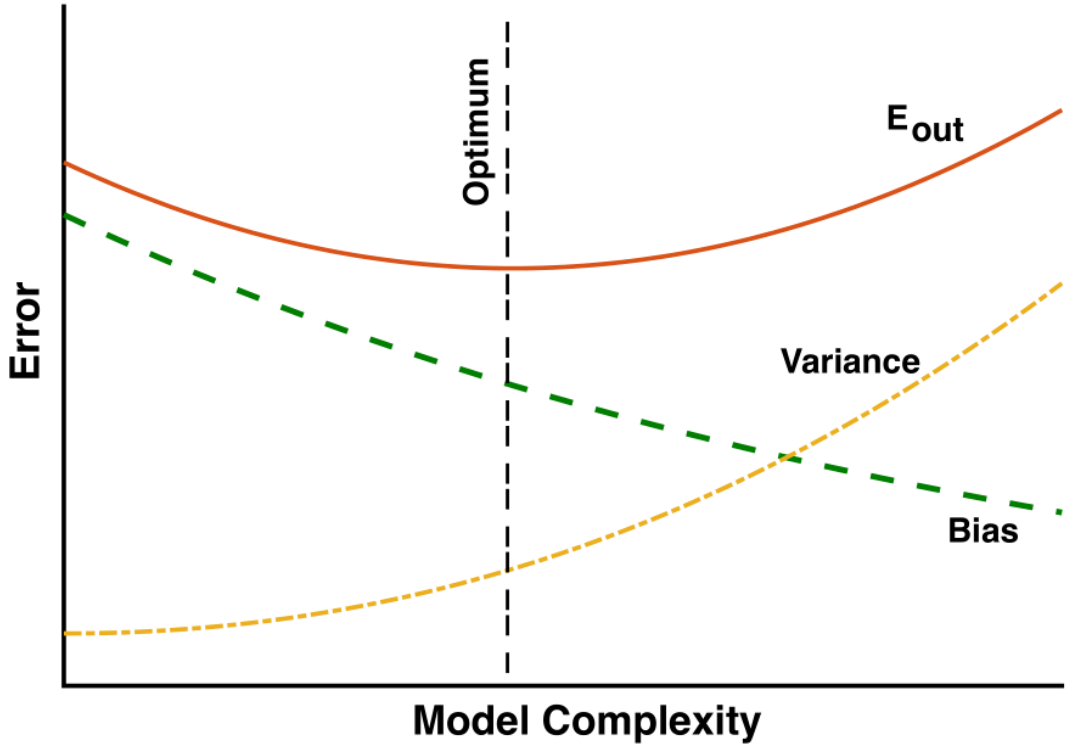
Bias vs variance

Exploitation vs exploration



Reducing bias or variance

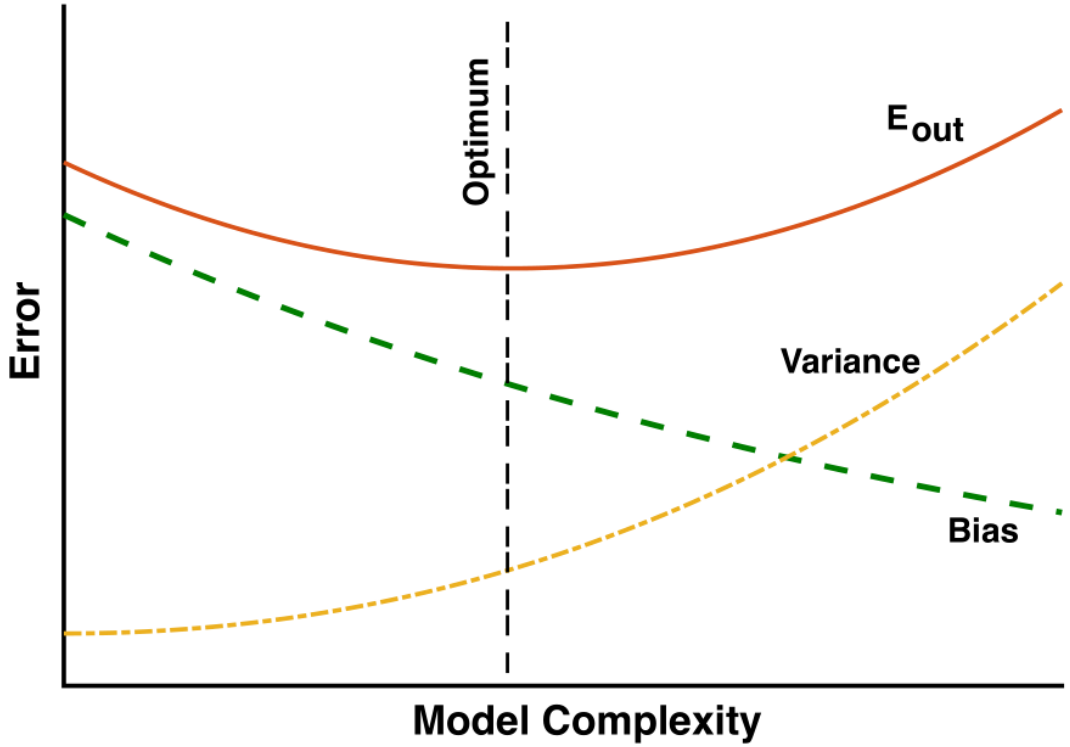
Model complexity



Mehta *et al.*, Physics reports **810**, 1-124 (2019)

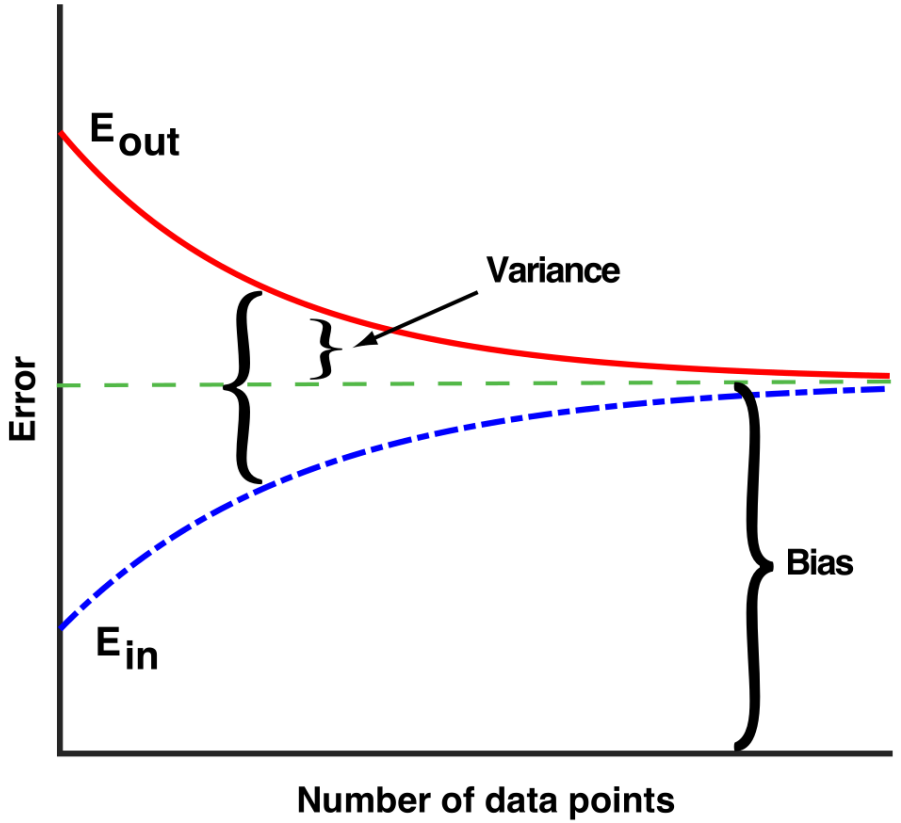
Reducing bias or variance

Model complexity



Mehta *et al.*, Physics reports **810**, 1-124 (2019)

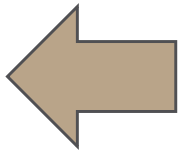
Data set size



Mehta *et al.*, Physics reports **810**, 1-124 (2019)

Gradient descent

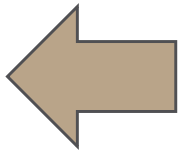
$$E = \sum_i |y_i - \hat{y}_i|$$



How to reduce the error function?

Gradient descent

$$E(w) = \sum_i |y_i - \hat{y}(w)_i|$$

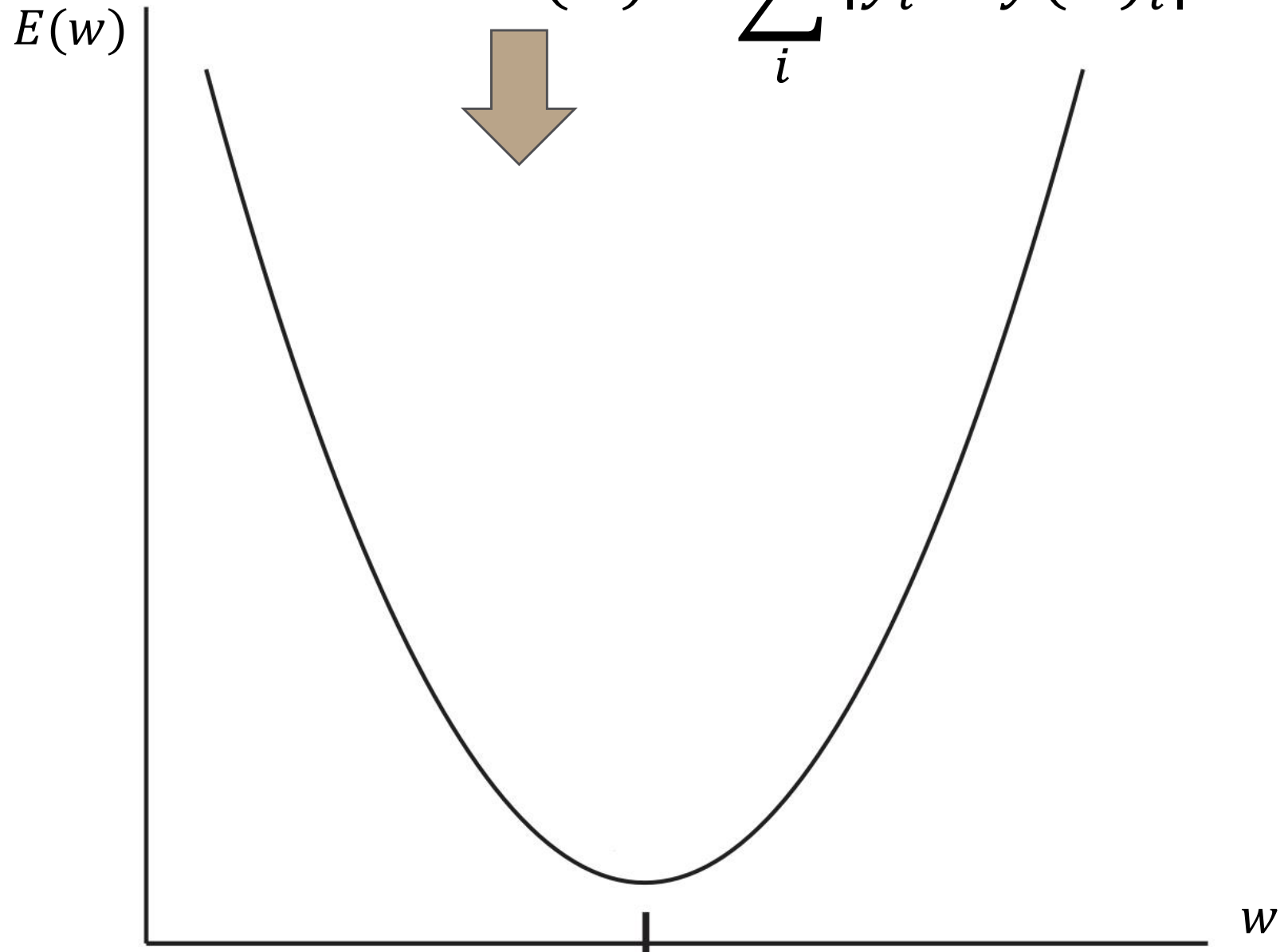


How to reduce the error function?

Gradient descent

$$E(w) = \sum_i |y_i - \hat{y}(w)_i|$$

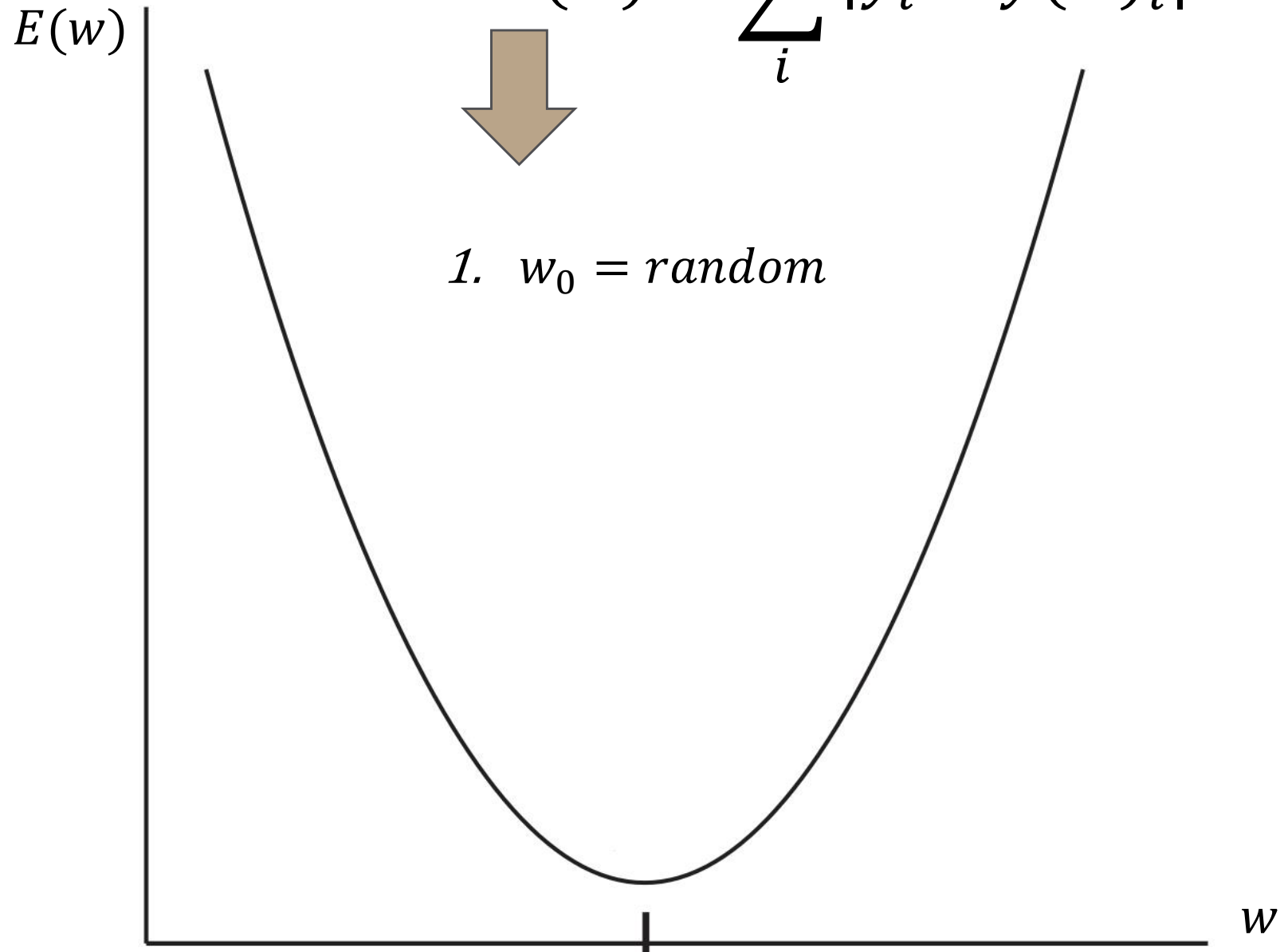
How to reduce the error function?



Gradient descent

$$E(w) = \sum_i |y_i - \hat{y}(w)_i|$$

How to reduce the error function?

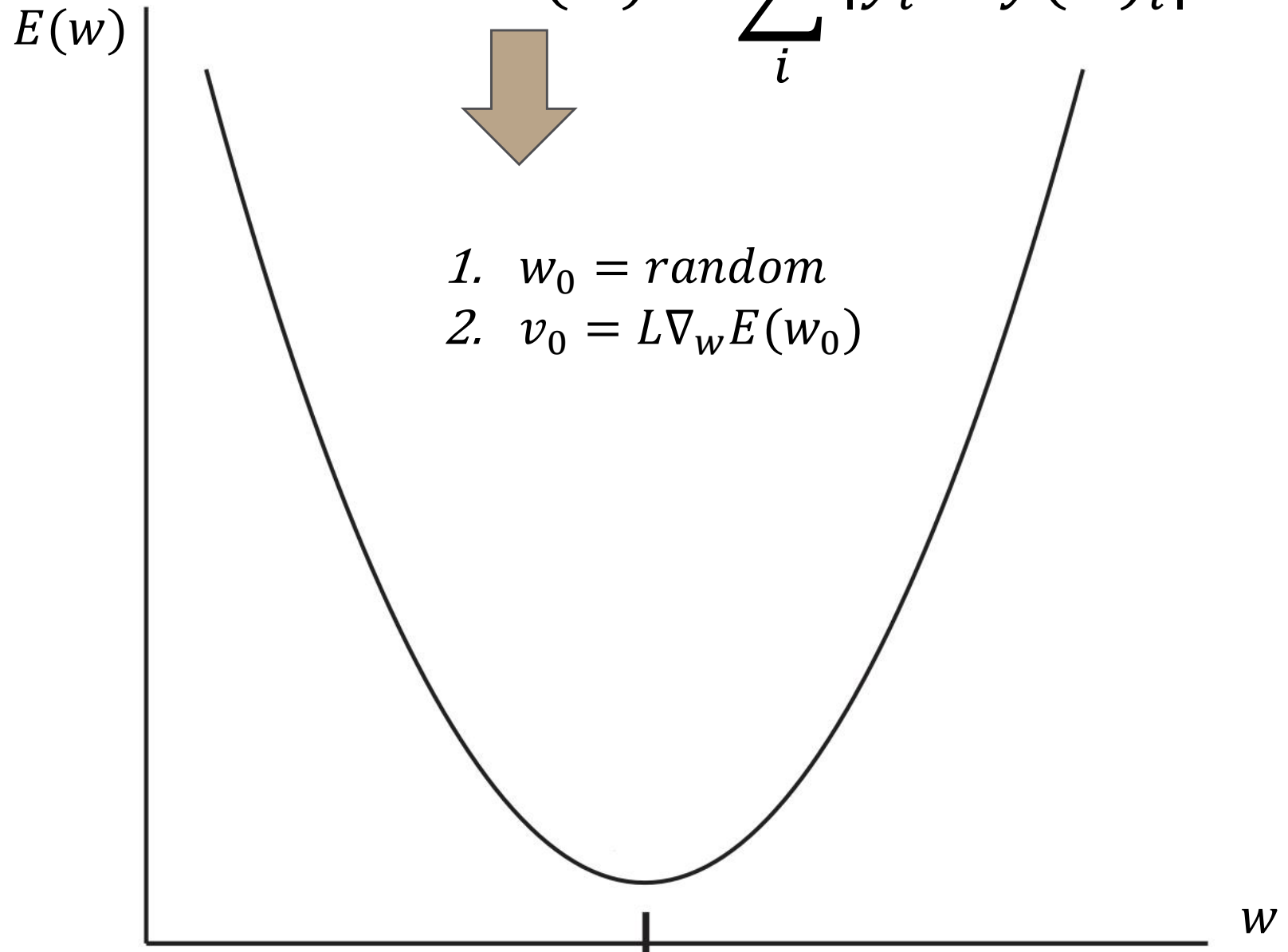


1. $w_0 = random$

Gradient descent

$$E(w) = \sum_i |y_i - \hat{y}(w)_i|$$

How to reduce the error function?

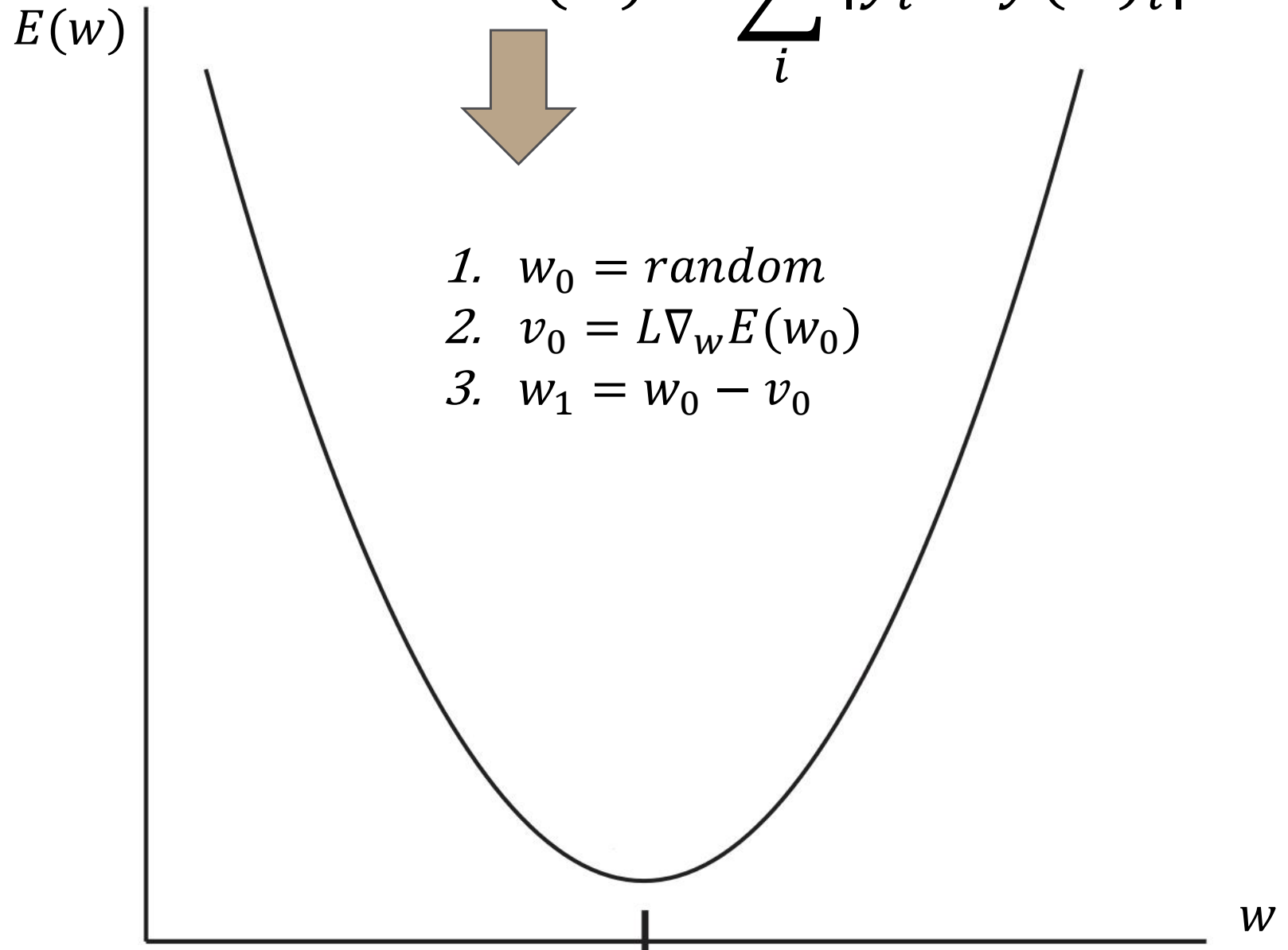


1. $w_0 = random$
2. $v_0 = L\nabla_w E(w_0)$

Gradient descent

$$E(w) = \sum_i |y_i - \hat{y}(w)_i|$$

How to reduce the error function?

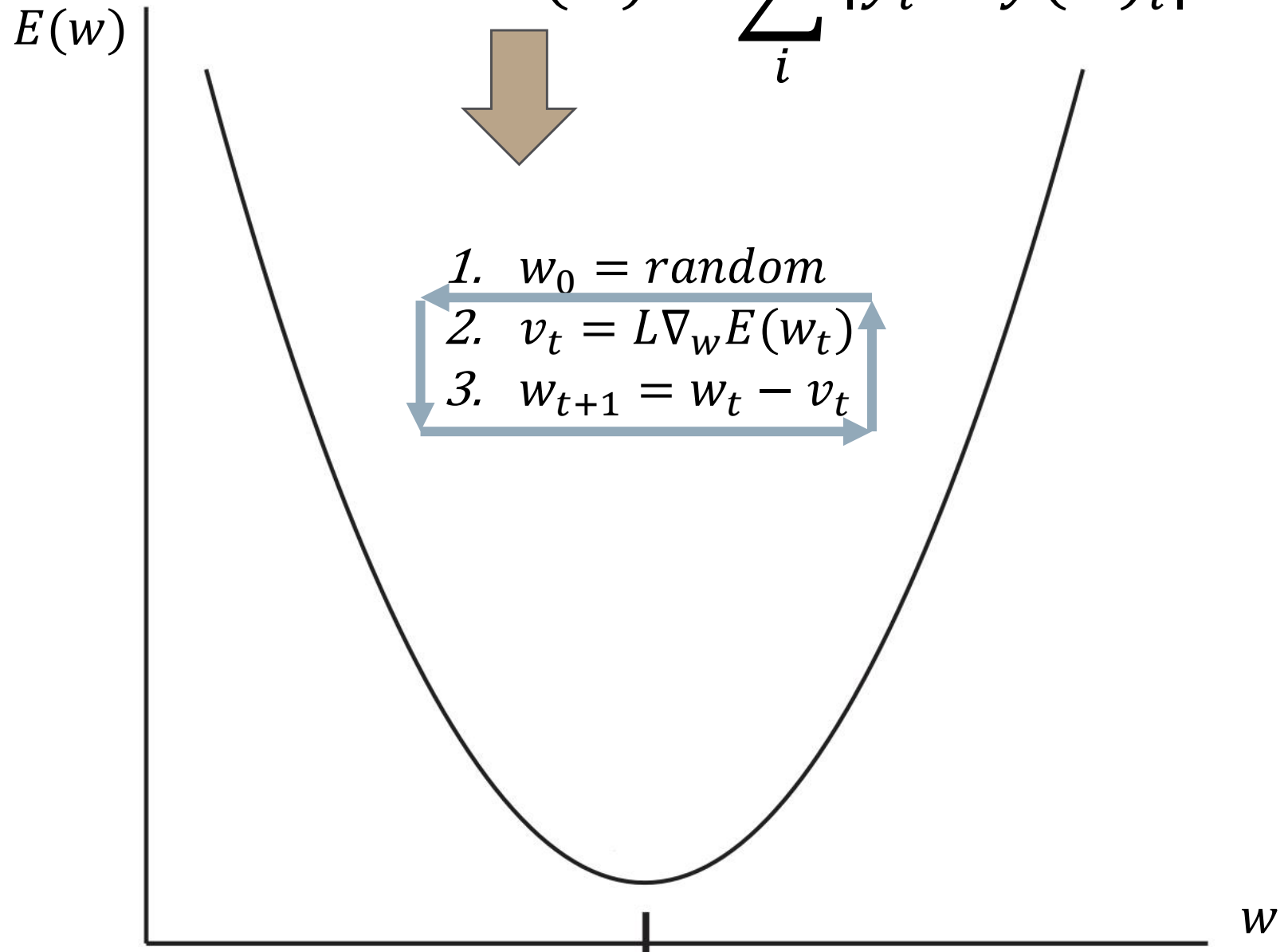


1. $w_0 = random$
2. $v_0 = L\nabla_w E(w_0)$
3. $w_1 = w_0 - v_0$

Gradient descent

$$E(w) = \sum_i |y_i - \hat{y}(w)_i|$$

How to reduce the error function?

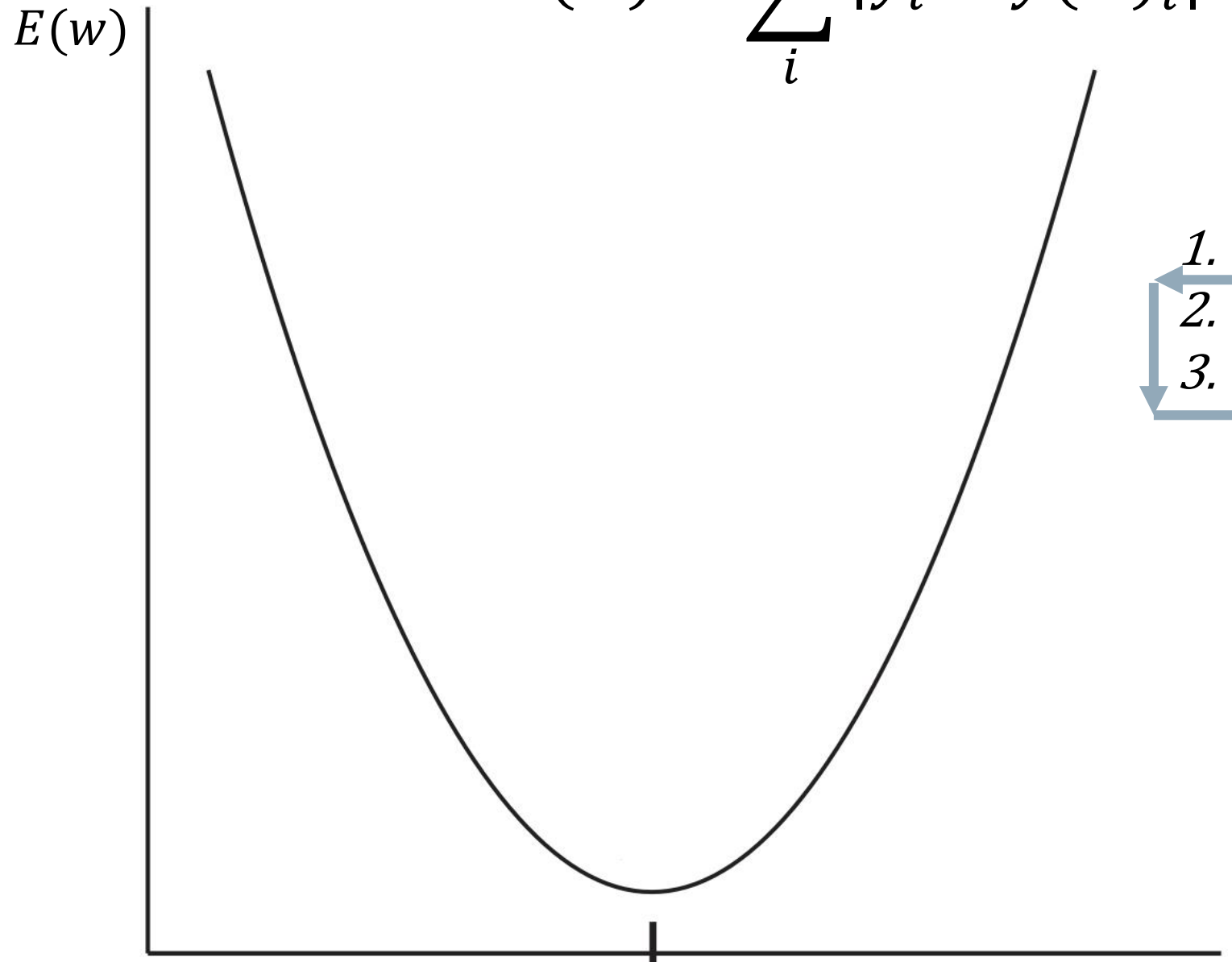


1. $w_0 = random$
2. $v_t = L\nabla_w E(w_t)$
3. $w_{t+1} = w_t - v_t$

Gradient descent

$$E(w) = \sum_i |y_i - \hat{y}(w)_i|$$

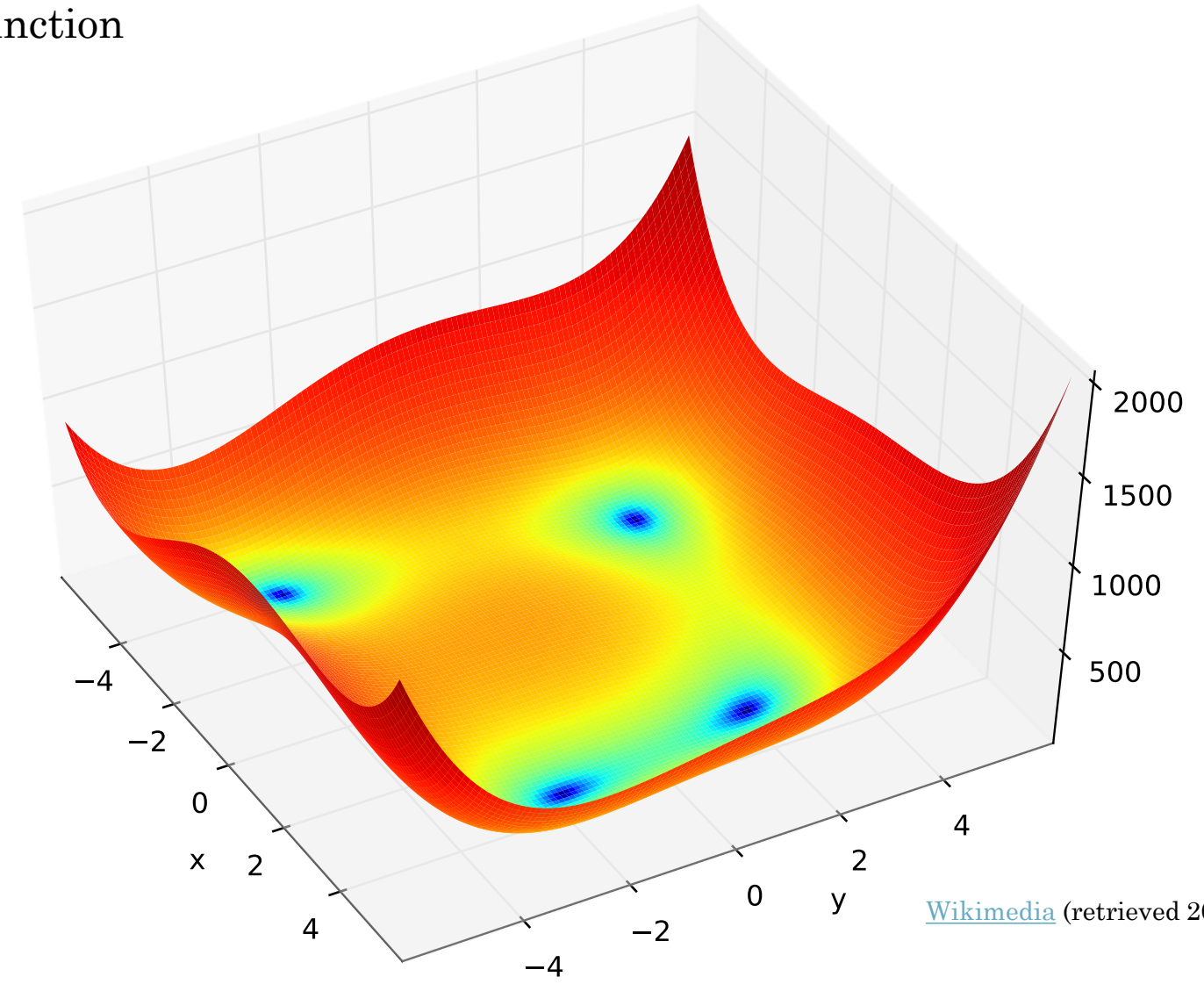
How to reduce the error function?



1. $w_0 = random$
2. $v_t = L\nabla_w E(w_t)$
3. $w_{t+1} = w_t - v_t$

Real-world error functions

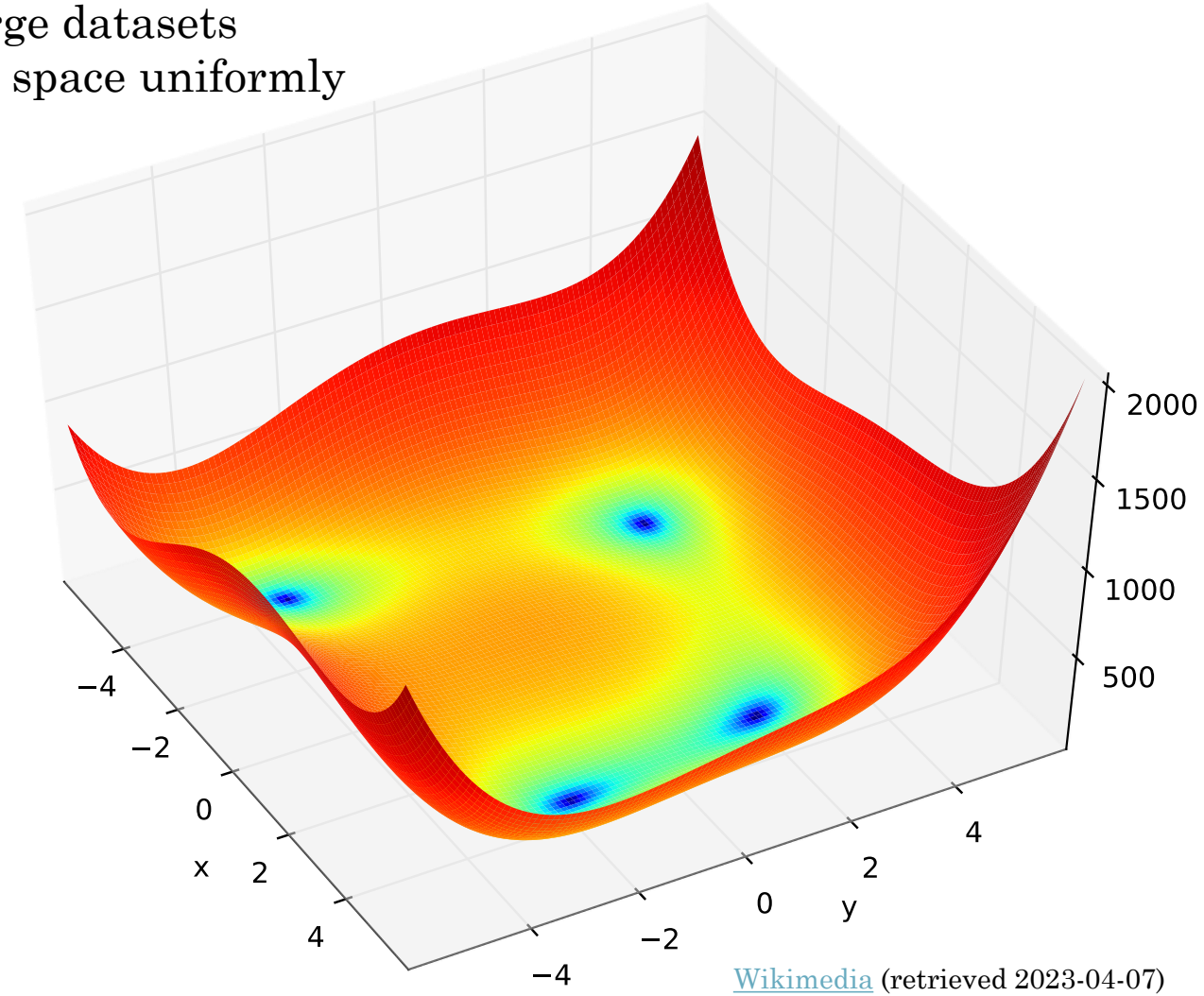
Himmelblau's function



Wikimedia (retrieved 2023-04-07)

Weeknesses of the gradient descent

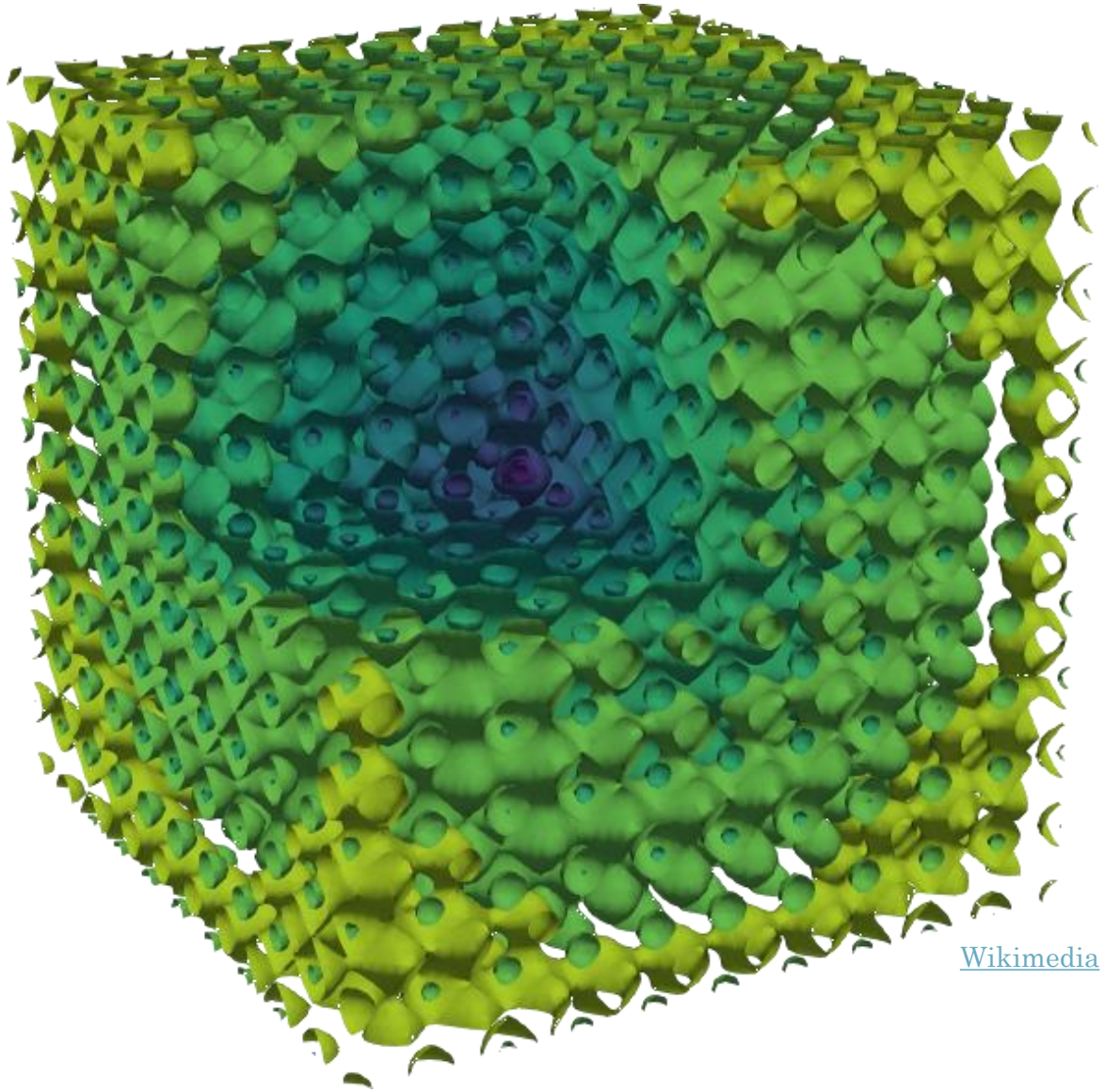
- GD finds local minima of the cost function
- Computationally expensive for large datasets
- Treats all directions in parameter space uniformly
- Sensitive to L
- Sensitive to w_0



Wikimedia (retrieved 2023-04-07)

Real-world error functions

Ackley function



Wikimedia (retrieved 2023-04-07)

Stochastic gradient descent

$$E(w) = \sum_{i \in B_k} |y_i - \hat{y}(w)_i|$$

Photo by Mark Stebnicki from Pexels retrieved 2023-04-02)

Stochastic gradient descent

$$E_{\boldsymbol{B}}(w) = \sum_{i \in B_k} |y_i - \hat{y}(w)_i|$$

- ✓ Less prone to getting stuck in local minima
- ✓ Less computationally expensive
- • Treats all directions in parameter space uniformly
- • Sensitive to L
- ✓ Less sensitive to w_0

1. $w_0 = random$
2. $v_t = L\nabla_w E_B(w_t)$
3. $w_{t+1} = w_t - v_t$

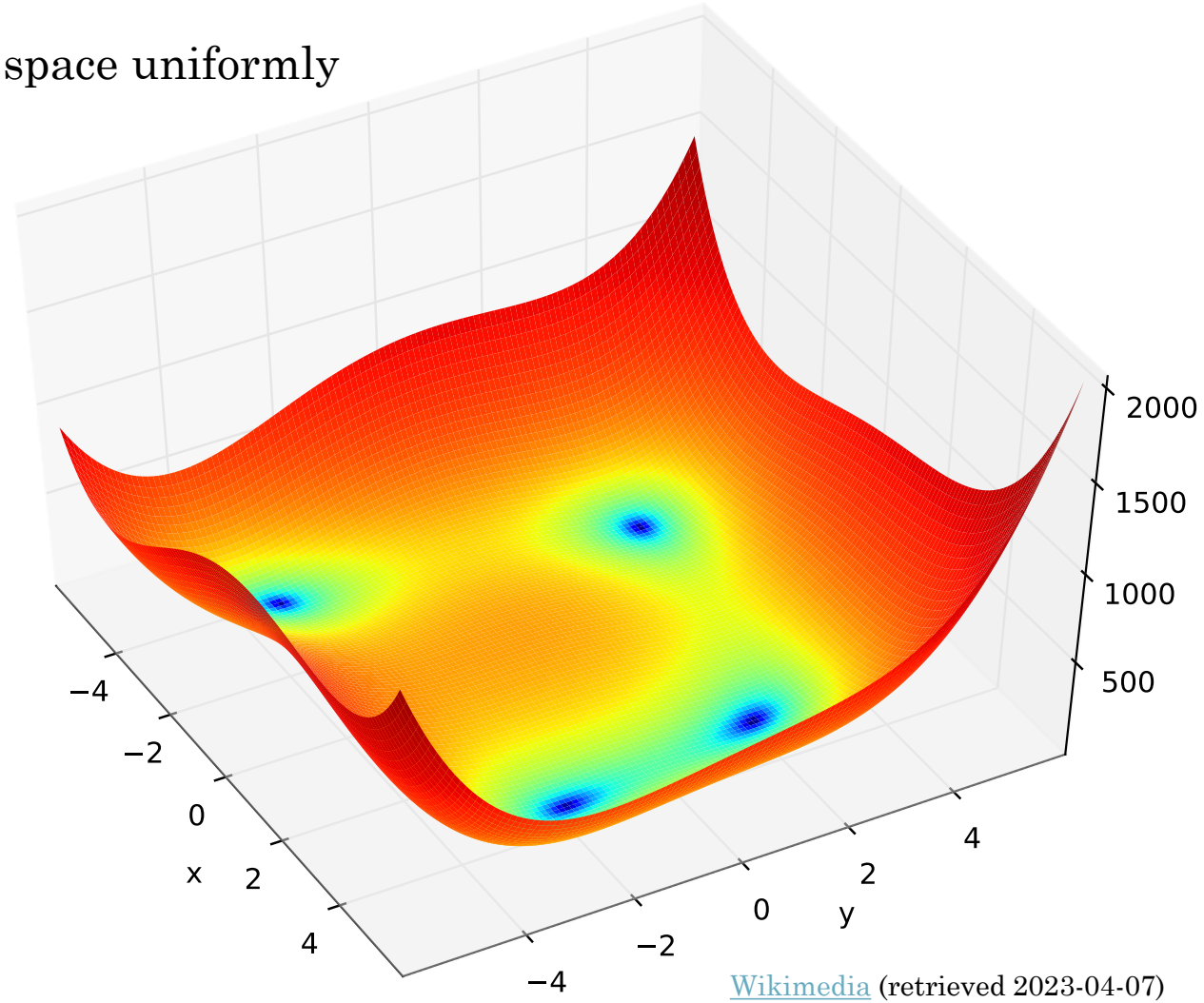
Photo by Mark Stebnicki from Pexels (retrieved 2023-04-02)

Stochastic gradient descent

$$E_{\boldsymbol{B}}(w) = \sum_{i \in B_k} |y_i - \hat{y}(w)_i|$$

- ✓ Less prone to getting stuck in local minima
- ✓ Less computationally expensive
- • Treats all directions in parameter space uniformly
- • Sensitive to L
- ✓ Less sensitive to w_0

1. $w_0 = random$
2. $v_t = L\nabla_w E_B(w_t)$
3. $w_{t+1} = w_t - v_t$



Wikimedia (retrieved 2023-04-07)

Adam

$$E_B(w) = \sum_{i \in B_k} |y_i - \hat{y}(w)_i|$$

- ✓ Less prone to getting stuck in local minima
- ✓ Less computationally expensive
- ✓ Doesn't treat all directions in parameter space uniformly
- ✓ Less sensitive to L
- ✓ Less sensitive to w_0

1. $w_0 = random$
2. $v_t = L\frac{m_t}{\sqrt{s_t}+\epsilon}$
3. $w_{t+1} = w_t - v_t$

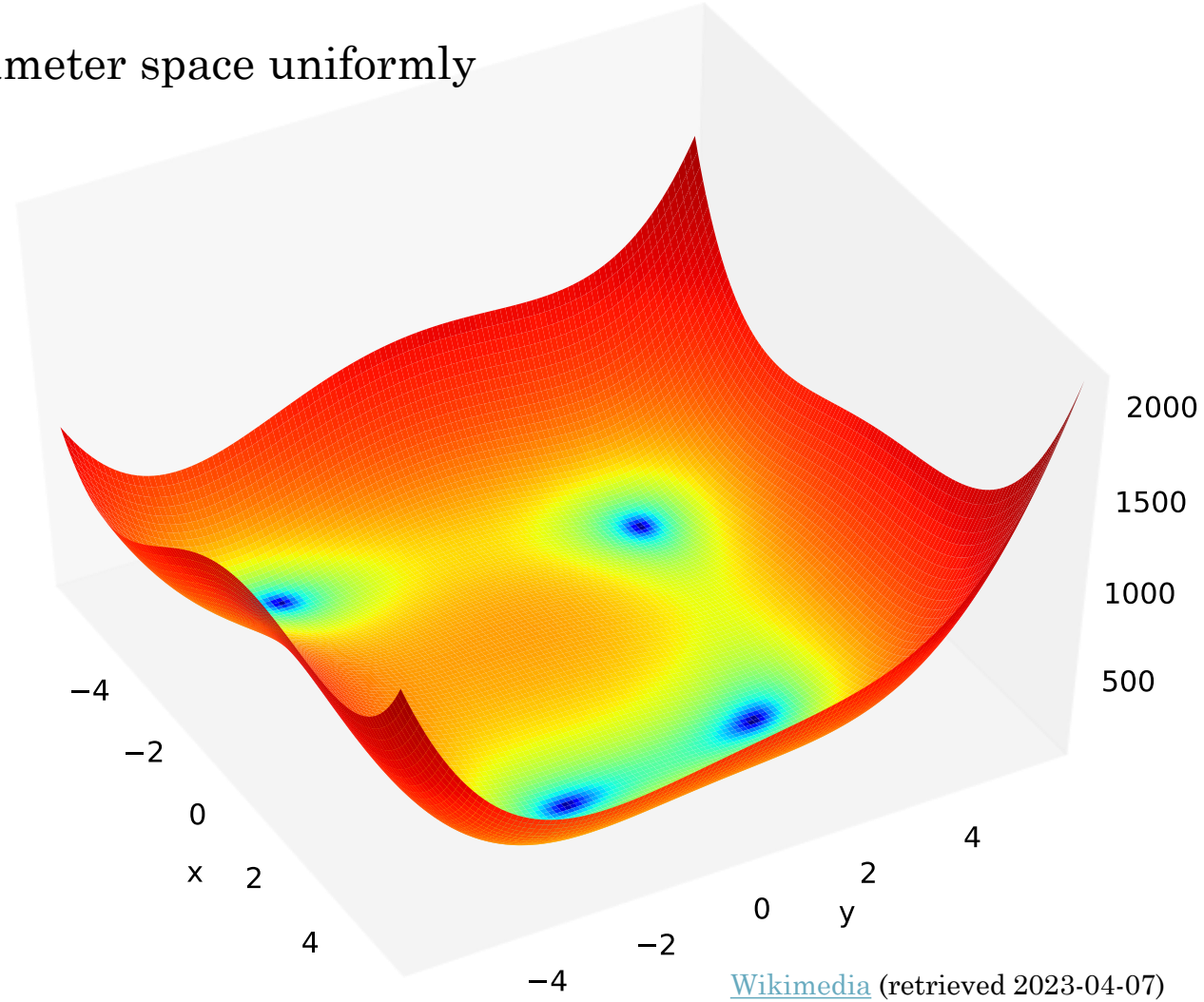
Photo by Ketut Subiyanto from Pexels (retrieved 2023-04-02)

Adam

$$E_B(w) = \sum_{i \in B_k} |y_i - \hat{y}(w)_i|$$

- ✓ Less prone to getting stuck in local minima
- ✓ Less computationally expensive
- ✓ Doesn't treat all directions in parameter space uniformly
- ✓ Less sensitive to L
- ✓ Less sensitive to w_0

1. $w_0 = random$
2. $v_t = L\frac{m_t}{\sqrt{s_t}+\epsilon}$
3. $w_{t+1} = w_t - v_t$



Wikimedia (retrieved 2023-04-07)

Regression vs classification

Regression	Classification

Classification: sigmoid and softmax

Sigmoid
- output is a vector of elements in (0,1)
- each element evaluated separately

$$f(x_i) = \frac{e^{x_i}}{e^{x_i} + 1}$$

$$\begin{pmatrix} 3 \\ 5 \\ 8 \\ 2 \\ 3 \\ 4 \end{pmatrix} \rightarrow ML \rightarrow sigmoid \rightarrow round \rightarrow \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Softmax
- output is a vector of elements in (0,1)
- "survival of the fittest"

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

$$\begin{pmatrix} 3 \\ 5 \\ 8 \\ 2 \\ 3 \\ 4 \end{pmatrix} \rightarrow ML \rightarrow softmax \rightarrow round \rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Classification: cross-entropy

The go-to error function for classification

$$P = \prod_i f(x_i, w)^{y_i} (1 - f(x_i, w))^{1-y_i}$$

Classification: cross-entropy

The go-to error function for classification

$$P = \prod_i f(x_i, w)^{y_i}(1 - f(x_i, w))^{1-y_i}$$

$$l = \sum_i y_i \log(f)(1 - y_i)\log(1 - f)$$

Classification: cross-entropy

The go-to error function for classification

$$P = \prod_i f(x_i, w)^{y_i}(1 - f(x_i, w))^{1-y_i}$$

$$l = \sum_i y_i \log(f)(1 - y_i)\log(1 - f)$$

$$\hat{w} = argmax_w\ (l)$$

Classification: cross-entropy

The go-to error function for classification

$$P = \prod_i f(x_i, w)^{y_i}(1 - f(x_i, w))^{1-y_i}$$

$$l = \sum_i y_i \log(f)(1 - y_i)\log(1 - f)$$

$$\hat{w} = argmax_w\ (l) = argmin_w\ (-l)$$

Classification: cross-entropy

The go-to error function for classification

$$P = \prod_i f(x_i, w)^{y_i}(1 - f(x_i, w))^{1-y_i}$$

$$l = \sum_i y_i \log(f)(1 - y_i)\log(1 - f)$$

$$\hat{w} = argmax_w\ (l) = argmin_w\ (-l)$$

cross-entropy $= -\sum_i y_i \log(f)(1 - y_i)\log(1 - f)$

Bagging and boosting

- BAGGing = Bootstrap AGGregation
- Bootstrapping: sampling data with replacement (not to confuse with *batching*)

Wikimedia (retrieved 2023-04-07)

Bagging and boosting

- BAGGing = Bootstrap AGGregation
- Bootstrapping: sampling data with replacement
- Approach:
  1. Bootstrap various data samples
  2. Create a weak classifier for each bootstrapped sample
  3. Aggregate the weak classifiers into one "collectively" strong classifier

Wikimedia (retrieved 2023-04-07)

Bagging and boosting

- BAGGing = Boostrap AGGregation
- Bootstrapping: sampling data with replacement
- Approach:
  1. Bootstrap various data samples
  2. Create a weak classifier for each bootstrapped sample
  3. Apply a weight to each weak classifier (=**boosting**)
  4. Aggregate the weak classifiers into one "collectively" strong classifier

Image by svstudioart on Freepik (retrieved 2023-04-07)

Bagging and boosting

- BAGGing = Boostrap AGGregation
- Bootstrapping: sampling data with replacement
- Approach:
  1. Bootstrap various data samples
  2. Create a weak classifier for each bootstrapped sample
  3. Apply a weight to each weak classifier (=**boosting**)
  4. Aggregate the weak classifiers into one "collectively" strong classifier

- Bagging reduces variance but increases bias
- Best practice: use with "unstable" ML models

Image by svstudioart on Freepik (retrieved 2023-04-07)

Types of learning

Supervised

Photo by Andrea Piacquadio from Pexels (retrieved 2023-04-07)

Types of learning

Supervised

Photo by Andrea Piacquadio from Pexels (retrieved 2023-04-07)

Data 1	Data 2	Data …	Data 100	Labels
x11	x21	…	x1001	0
x12	x22	…	x1002	1
x13	x23		x1003	0
x14	x24		x1004	0
x15	x25		x1005	1

Types of learning

Unsupervised

Photo by Steve Johnson from Pexels (retrieved 2023-04-07)

Types of learning

Unsupervised

Examples:
- Clustering
- Dimensionality reduction
- Generative modeling

Photo by Steve Johnson from Pexels (retrieved 2023-04-07)

Types of learning

Reinforcement

Photo by Francesco Ungaro from Pexels (retrieved 2023-04-07)

Types of learning

Reinforcement

- Agent interacting with the environment.
- Updating behavior based on reward
- Often used in games and robotics

Photo by Francesco Ungaro from Pexels (retrieved 2023-04-07)