SSY097 - Image Analysis

Lecture 4 - Statistical Learning

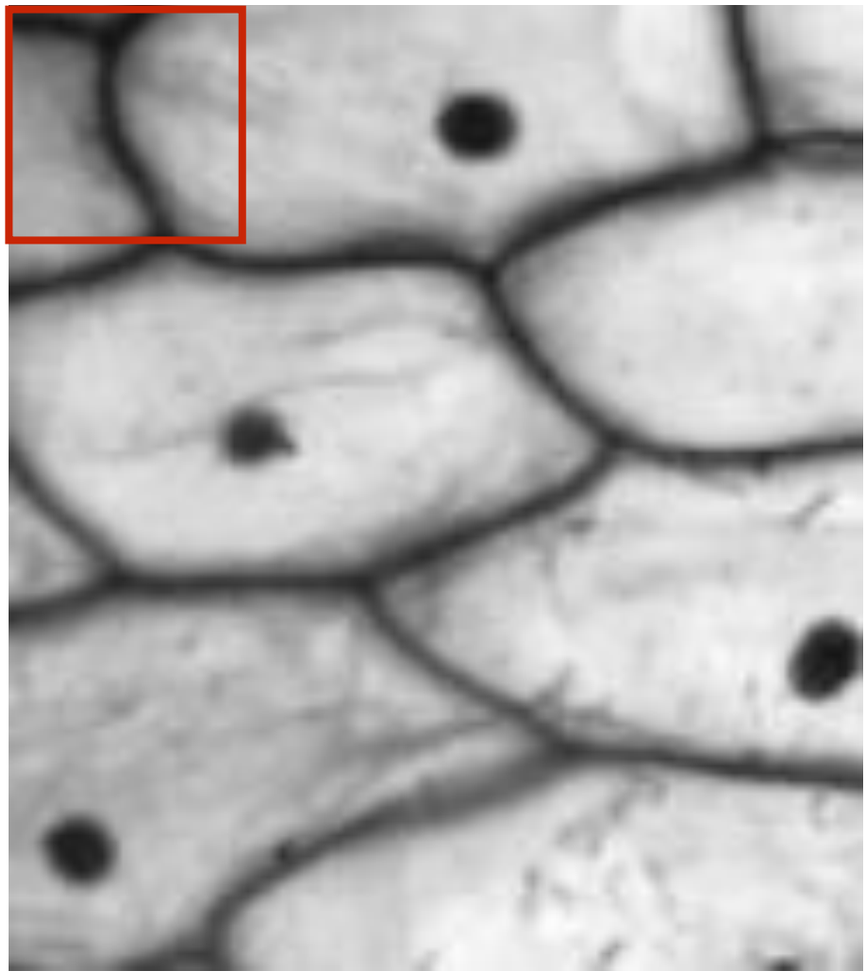
Torsten Sattler
(slides adapted from Olof Enqvist)

Two Lectures Ago

Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	**Filtering, gradients, scale**	Lab 1
Jan. 27	Local features	
Jan. 30	Learning a classifier	Lab 2
Feb. 3	Convolutional neural networks	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
Feb. 17	Camera Geometry	Lab 4
Feb. 20	More camera geometry	
Feb. 24	Generative neural networks	
Feb. 27	Generative neural networks	
Mar. 2	TBA	
Mar. 9	TBA	

First Lecture

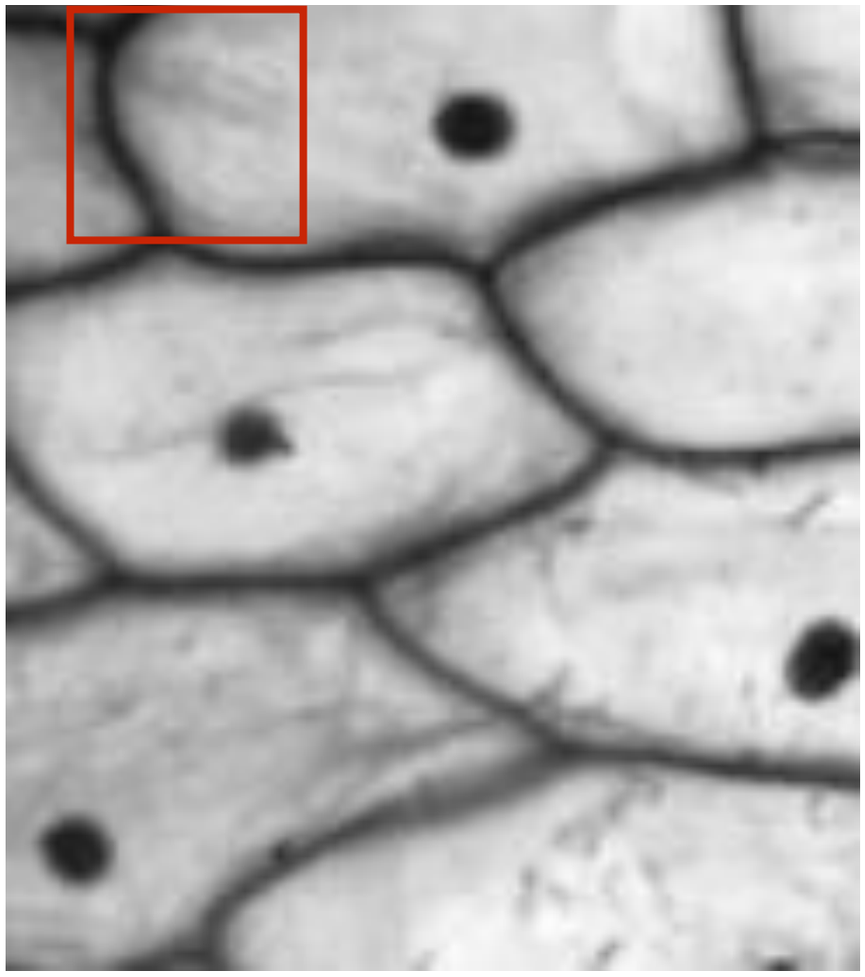
Sliding window classification



$$\cdot w < \tau$$

First Lecture

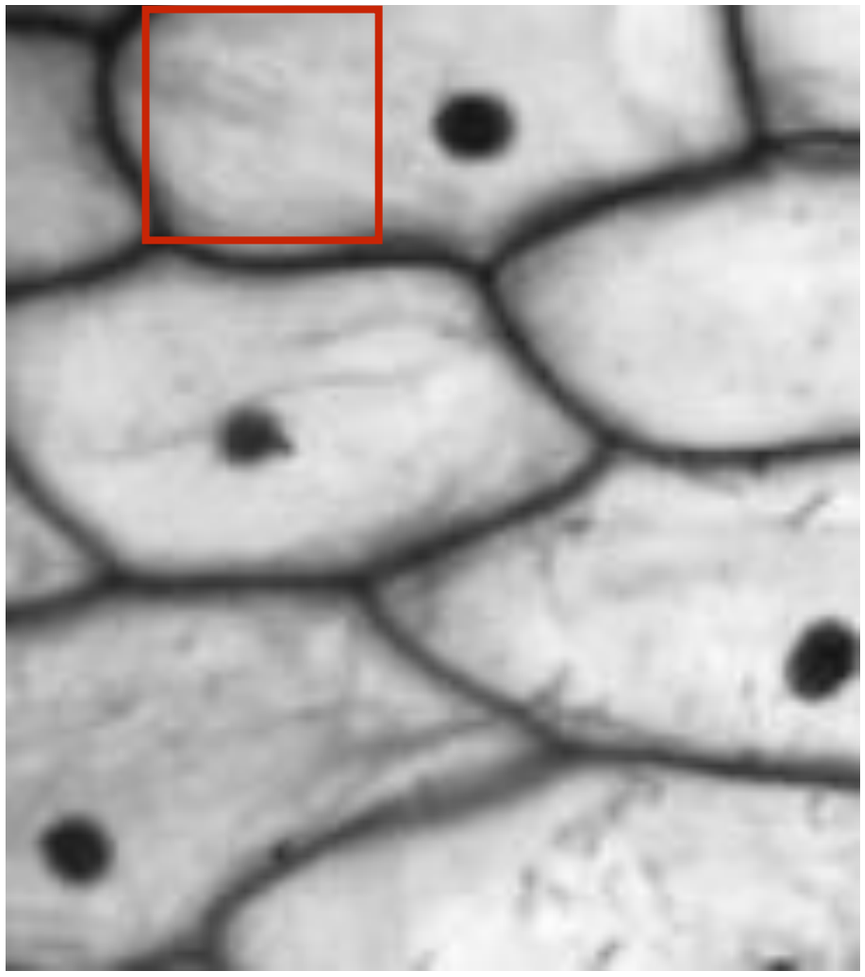
Sliding window classification



$$\cdot w < \tau$$

First Lecture

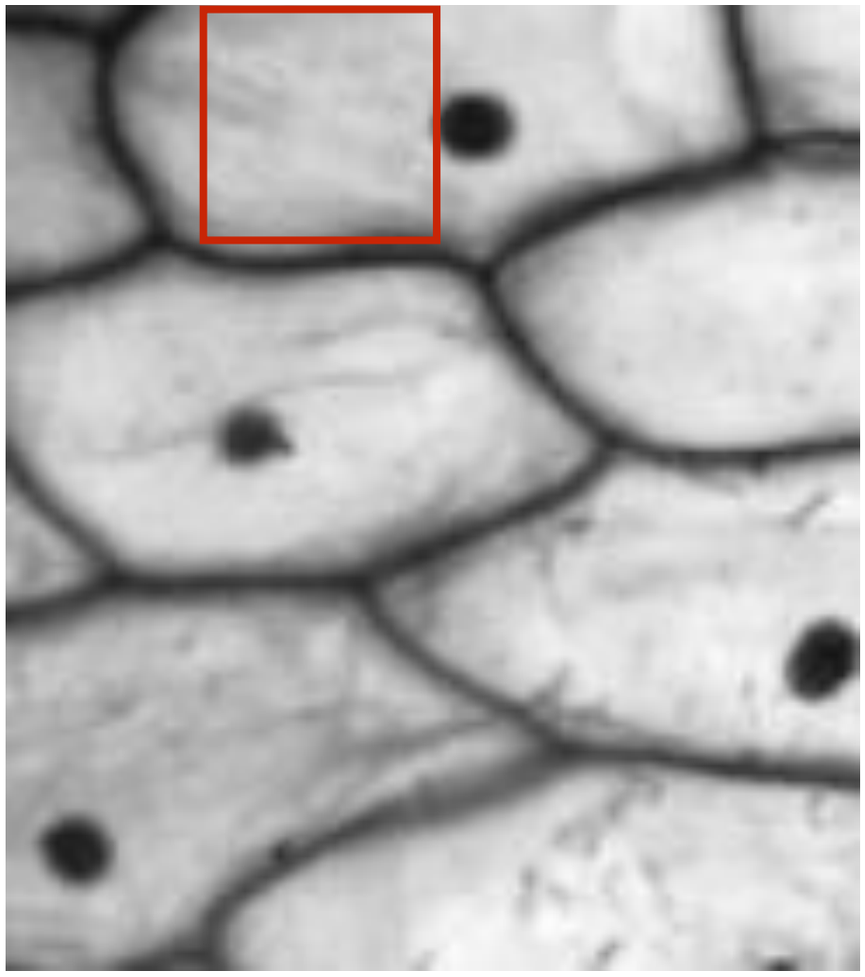
Sliding window classification



$$\cdot w < \tau$$

First Lecture

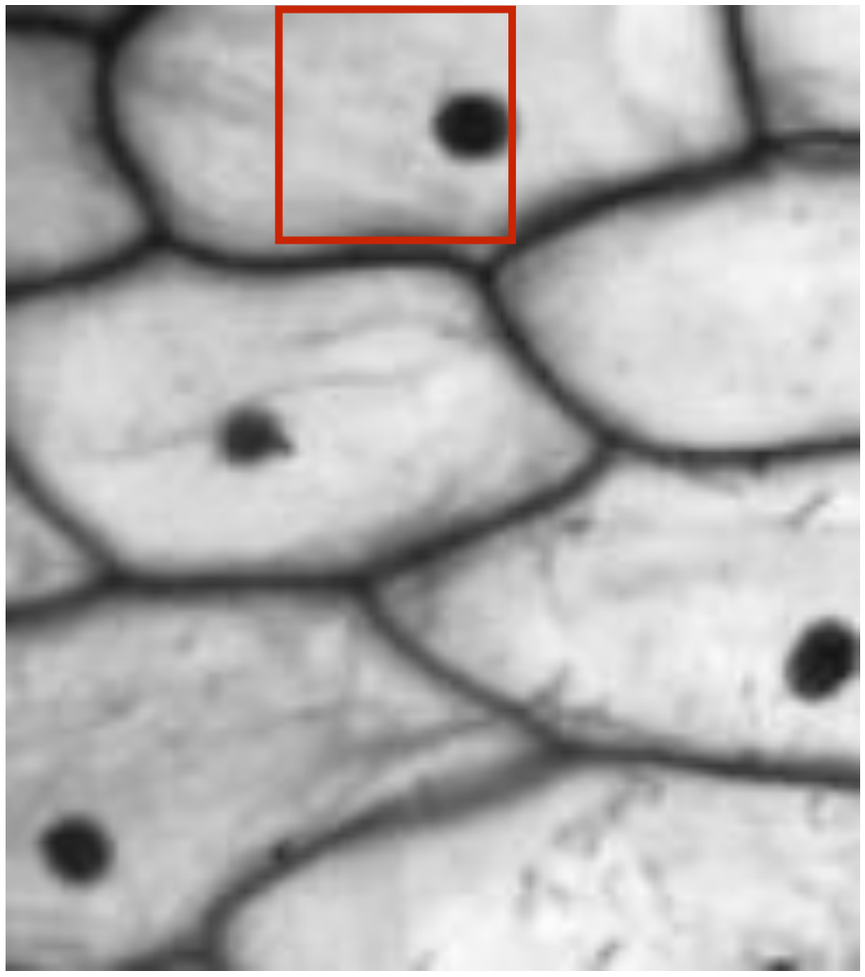
Sliding window classification



$$\cdot w < \tau$$

First Lecture

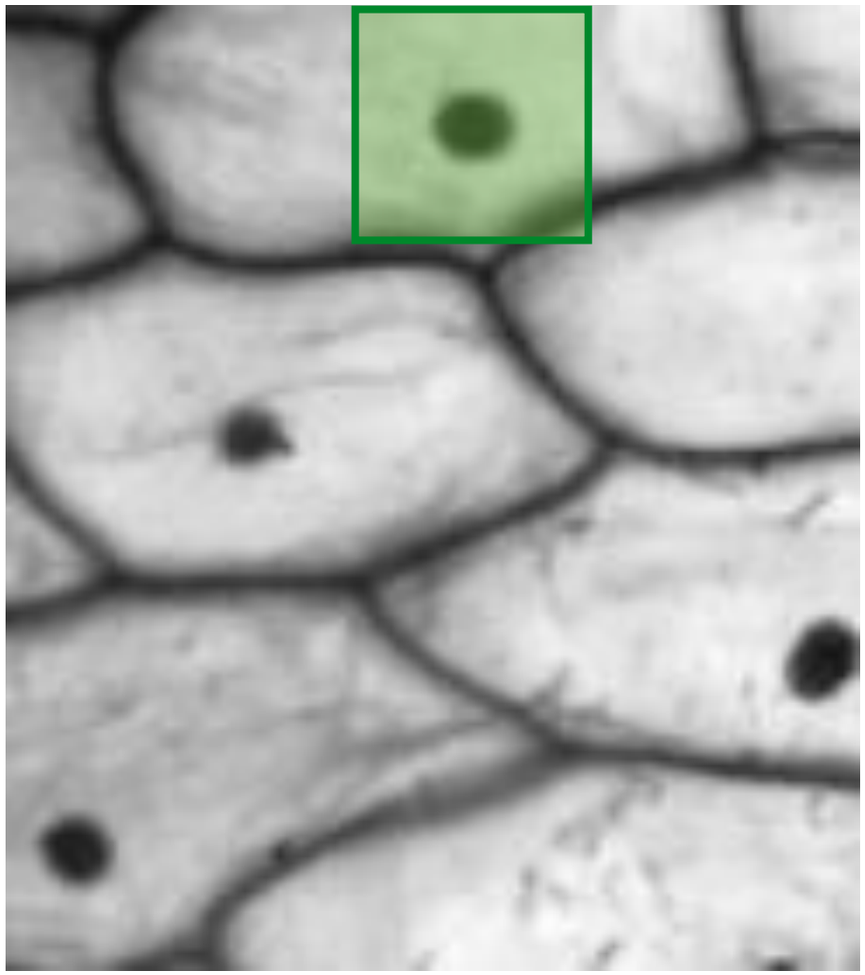
Sliding window classification



$$\cdot w < \tau$$

First Lecture

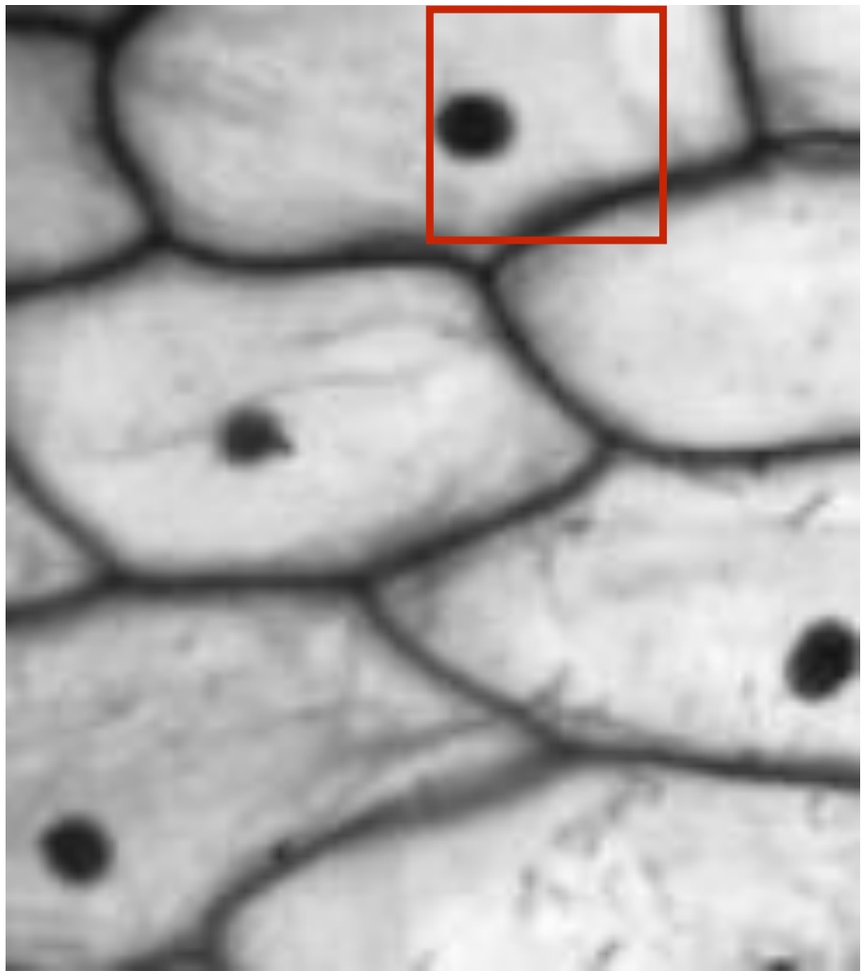
Sliding window classification



$$\cdot w > \tau$$

First Lecture

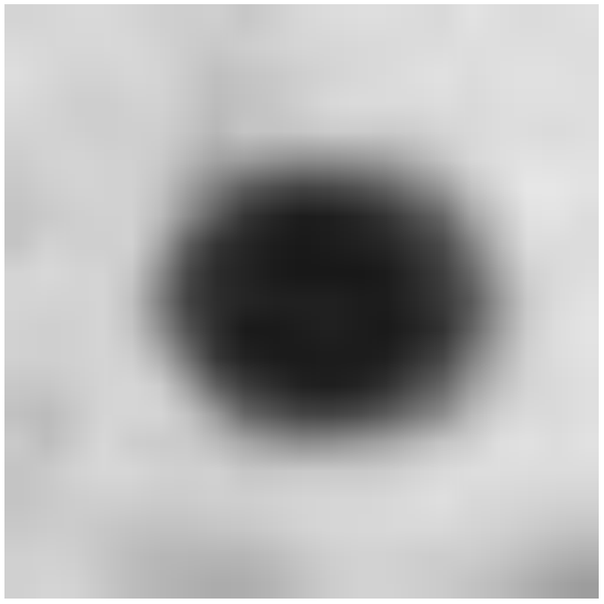
Sliding window classification



$$\cdot w < \tau$$

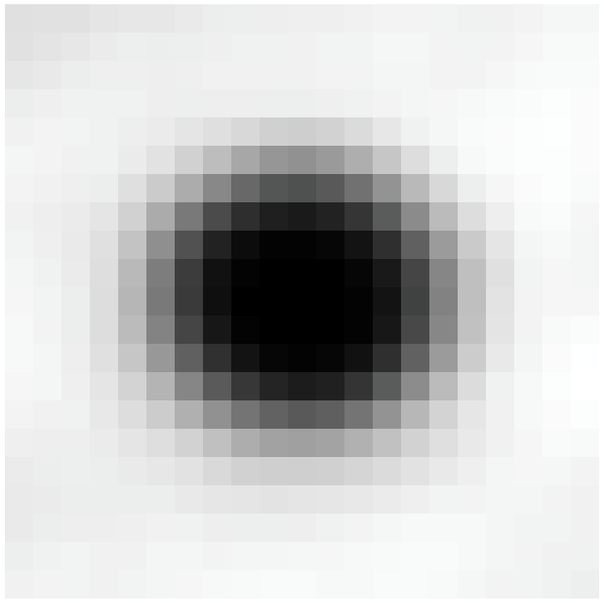
First Lecture

Linear filters

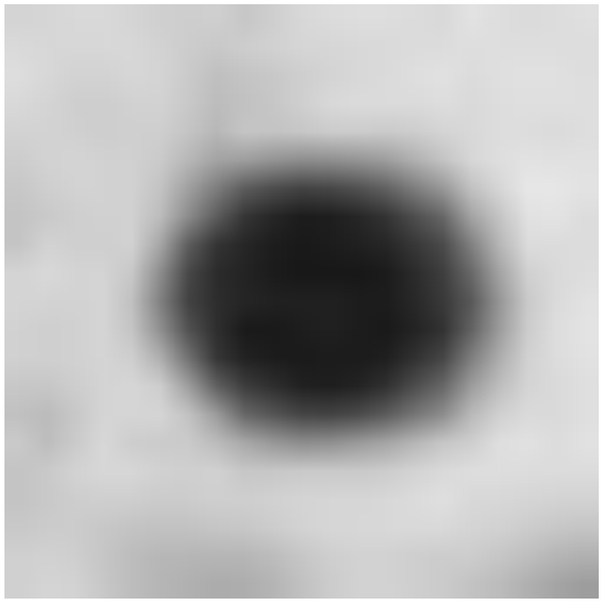
$$w \cdot$$  $$> \tau$$

First Lecture

Linear filters



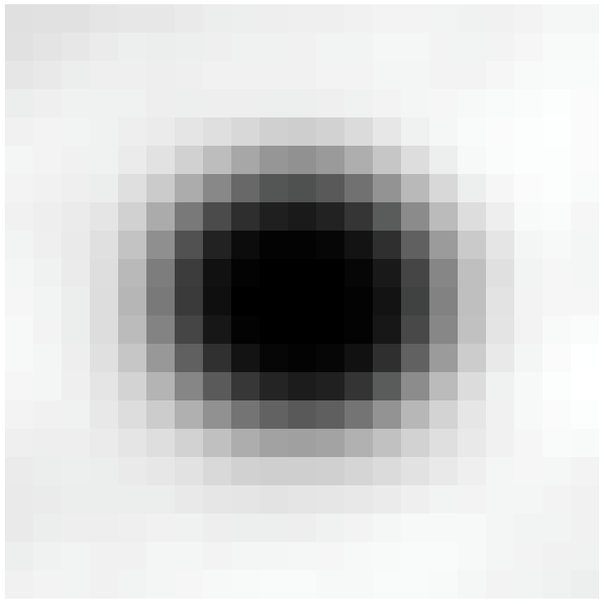
average template



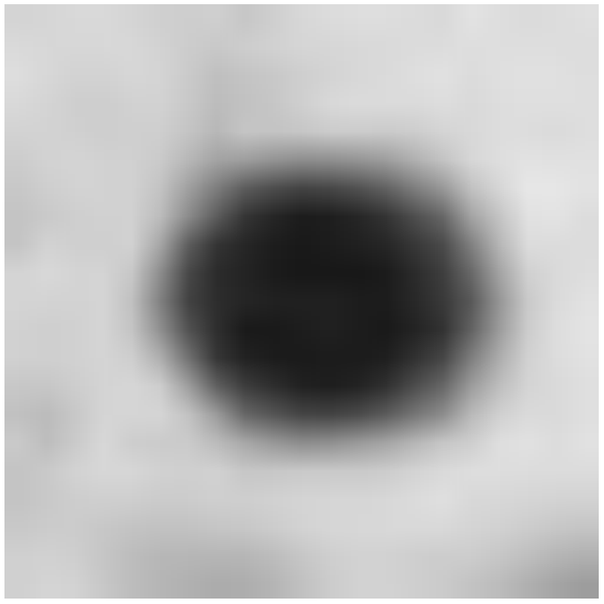
$$\cdot \qquad > \tau$$

First Lecture

Linear filters



average template

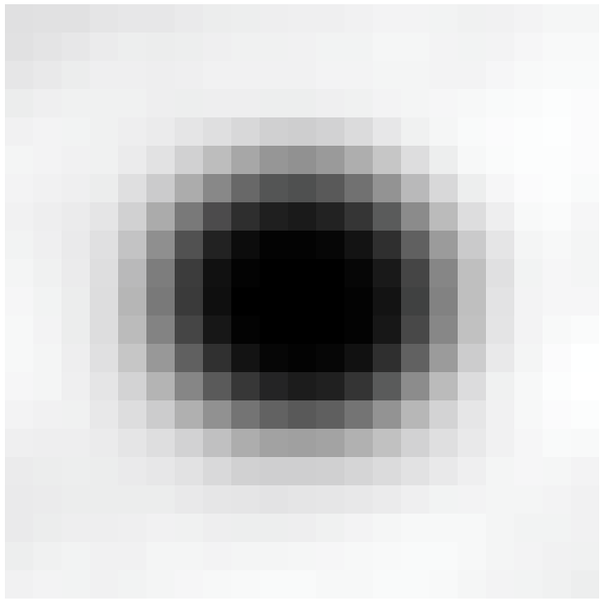


$$> \tau$$

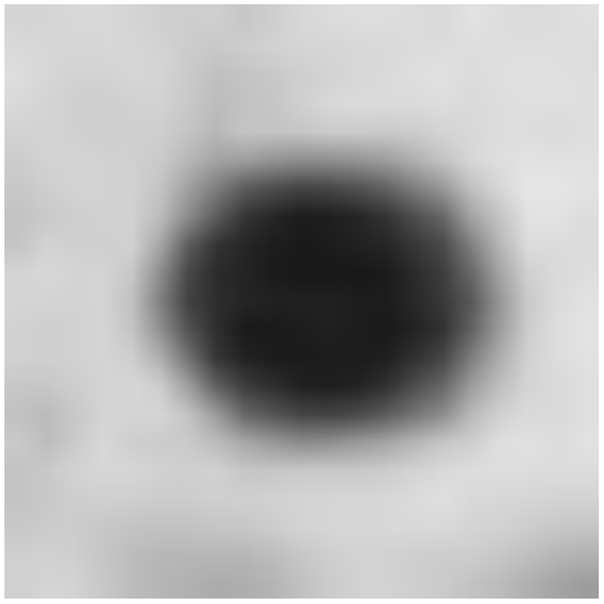
$$\frac{1}{N_{\text{elements}}}(T - \mu_T \mathbf{1})$$

First Lecture

Linear filters



average template



$$> \tau$$

$$\frac{1}{N_{\text{elements}}}(T - \mu_T \mathbf{1})$$

Can we do better than the average template?

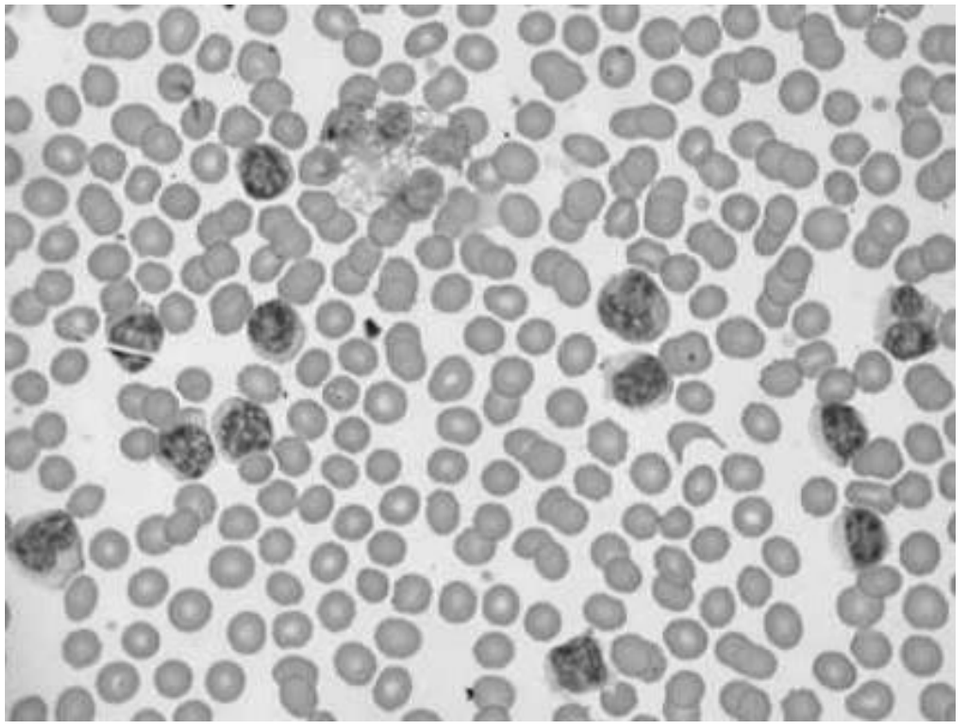
Today

- Learning a linear classifier
  - Loss functions
  - Optimization
  - Overfitting
  - Train - Validation - Test Splits

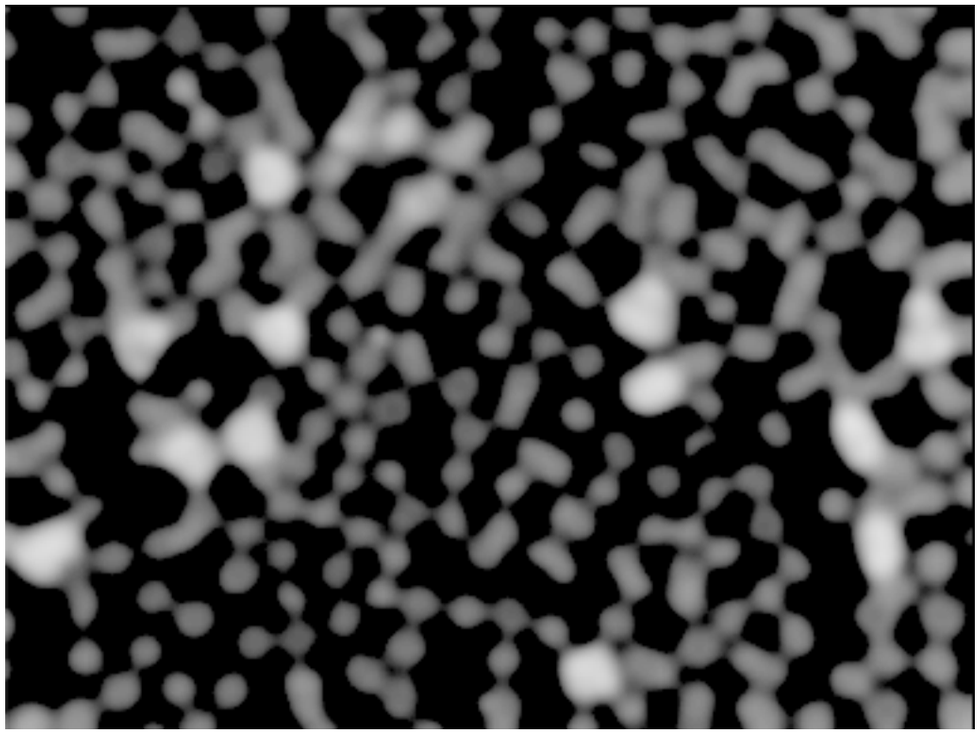
Statistical Learning

How to model learning a classifier?

Which is the best linear classifier?

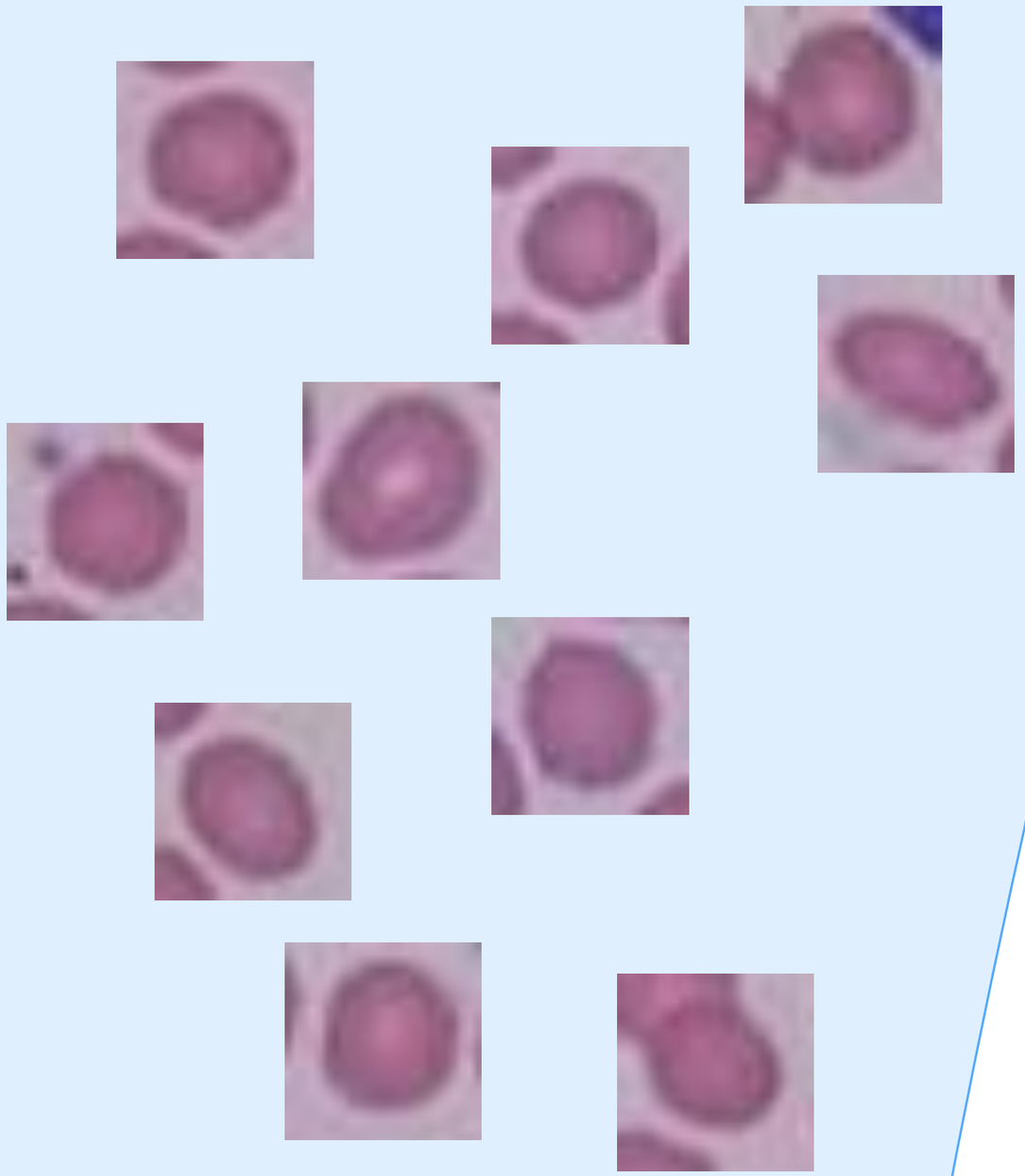


$\star w =$



…to use for sliding-window detection.

Manually Labelled Data



positive examples

negative examples

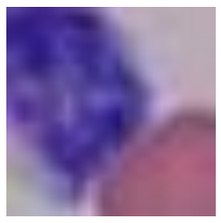
Linear Classifier

Find a w such that…

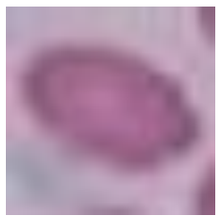
$\cdot\, w + w_0 > 0$

$\cdot\, w + w_0 < 0$

$\cdot\, w + w_0 > 0$



$\cdot\, w + w_0 < 0$



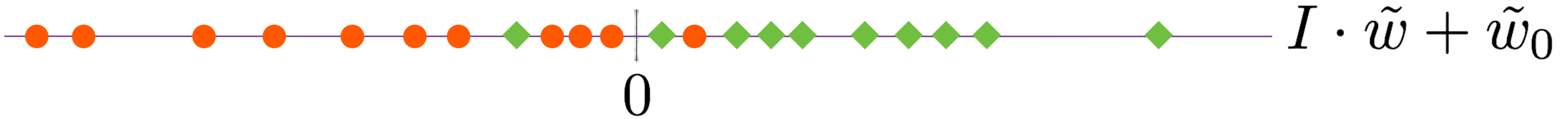
$\cdot\, w + w_0 > 0$

$\cdot\, w + w_0 < 0$

positive examples

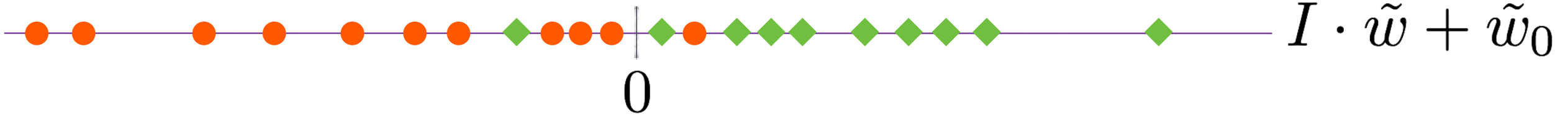
negative examples

Which Classifier is Better?

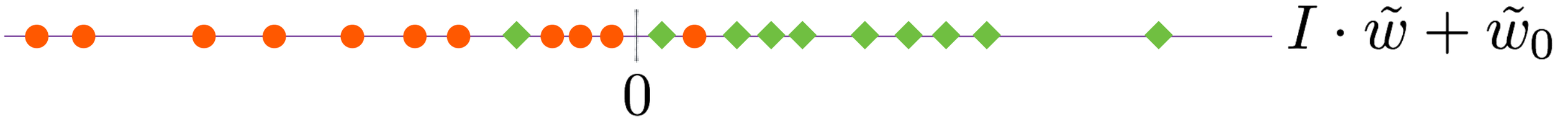


Which Classifier is Better?

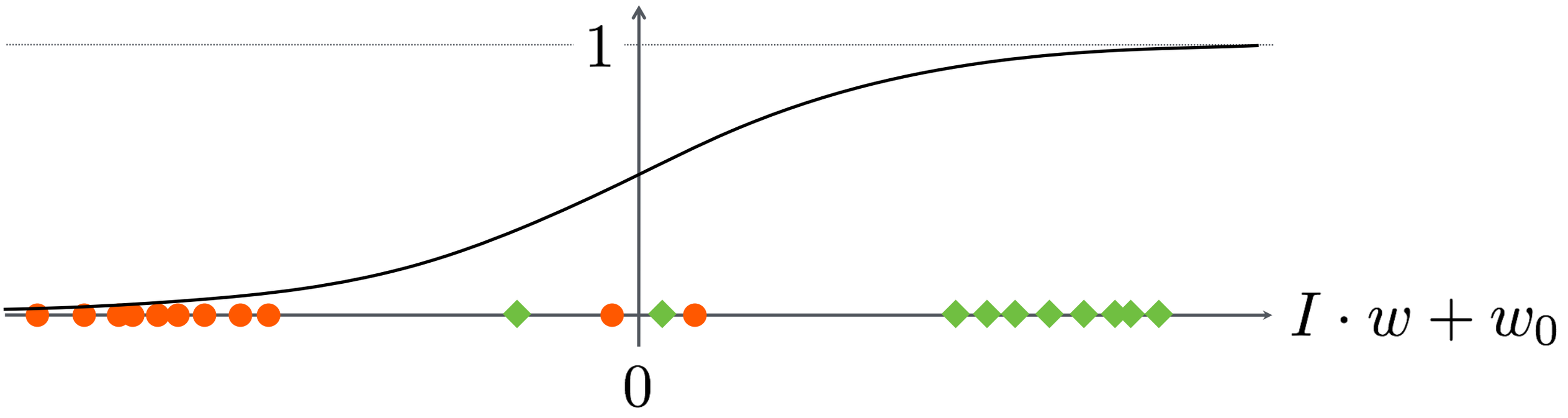
Both make the same number of mistakes!
Are both equally good?

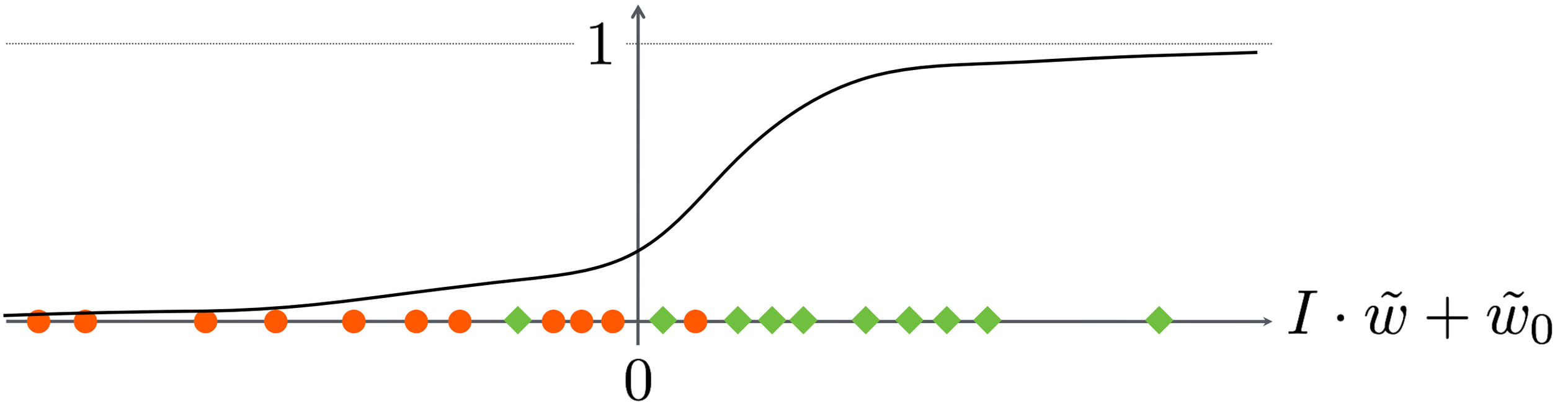


Produce a Probability

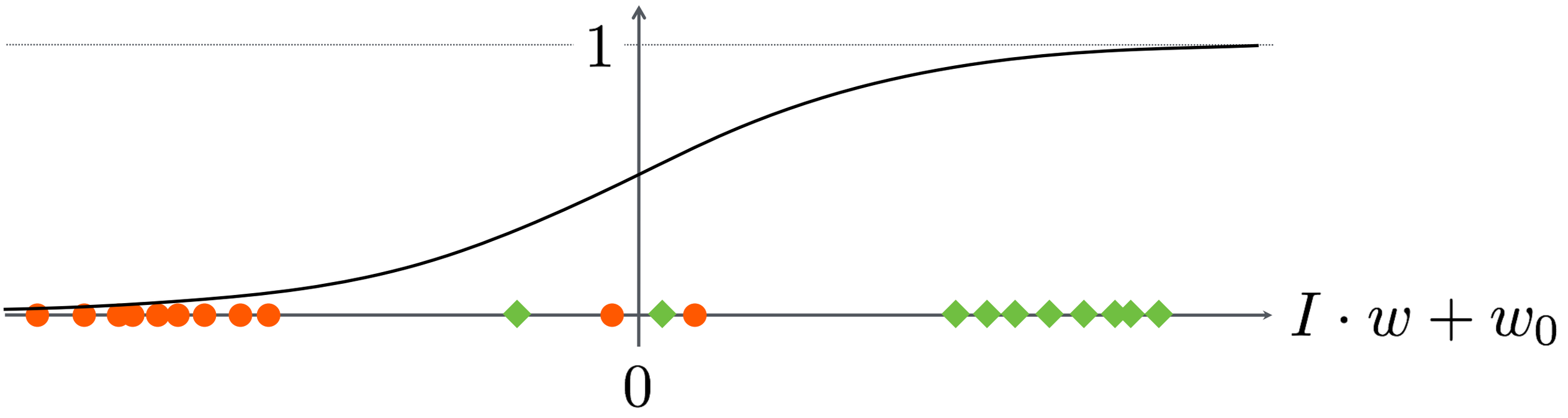


Produce a Probability

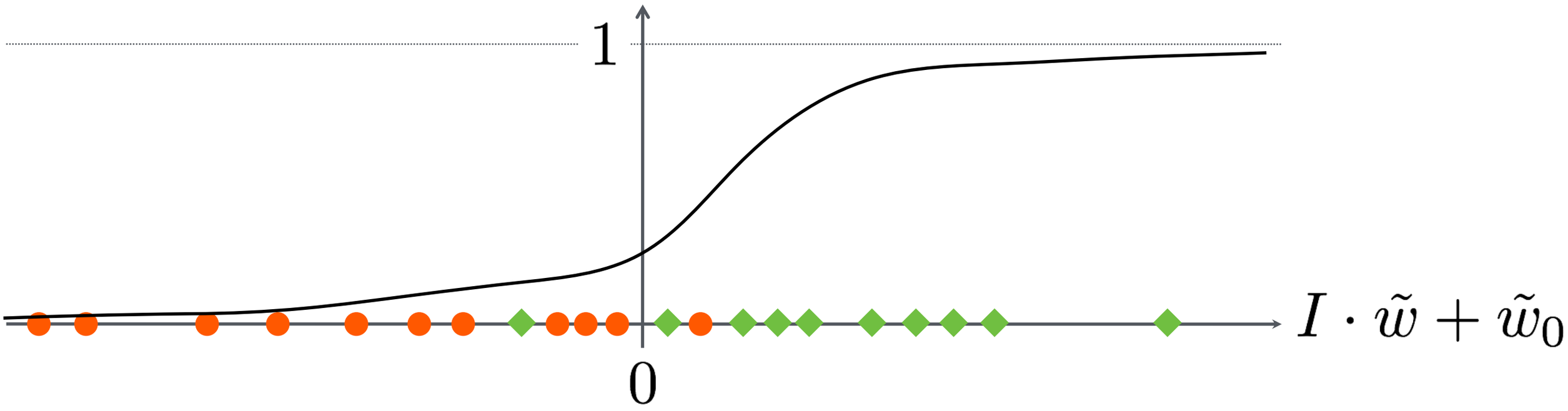




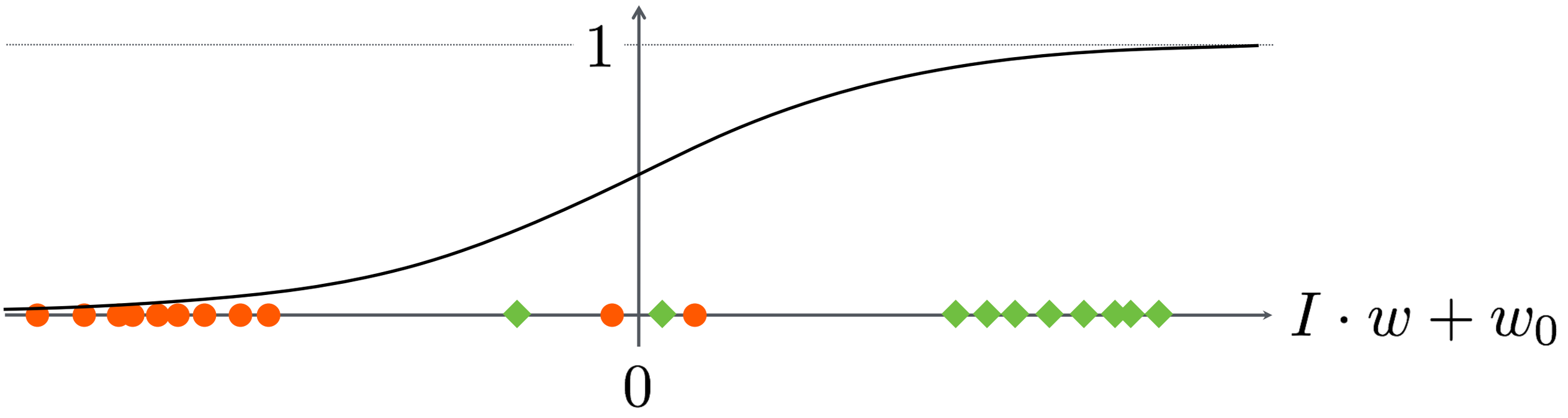
Produce a Probability



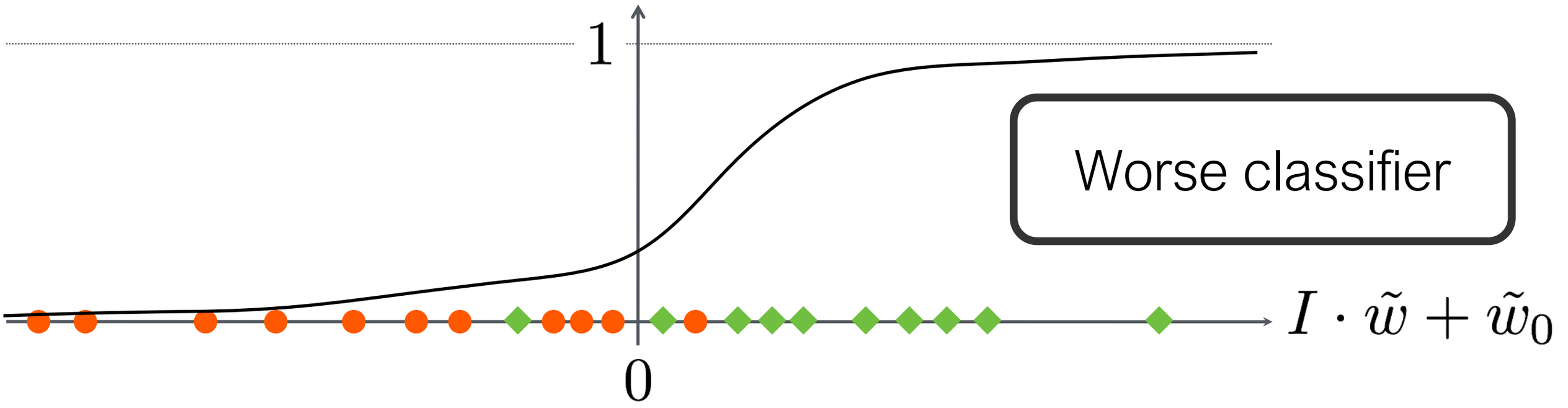
cumulative distribution function (cdf) for positive examples



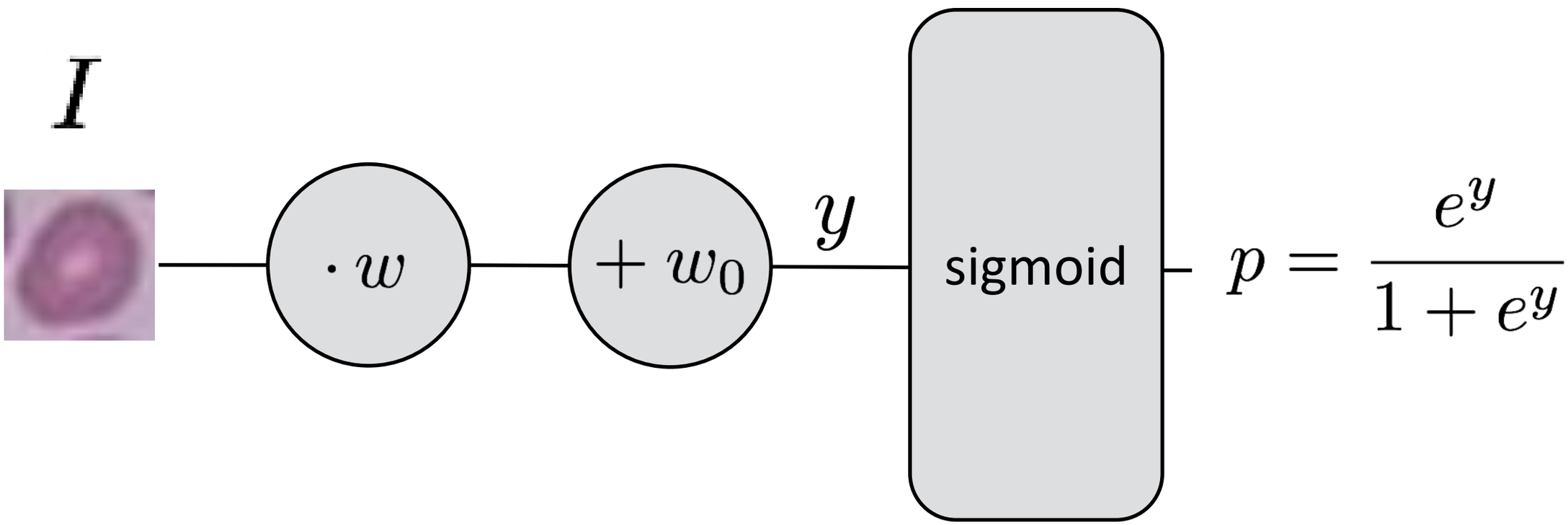
Produce a Probability



cumulative distribution function (cdf) for positive examples

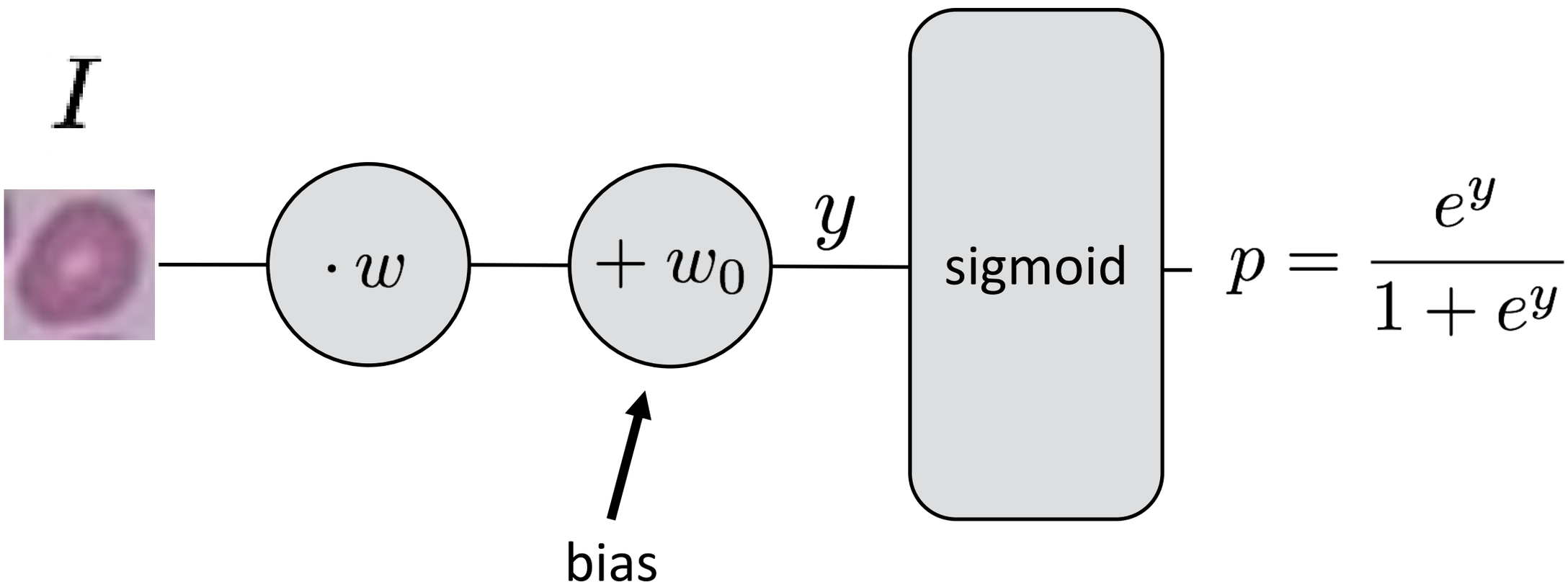


Sigmoid Function



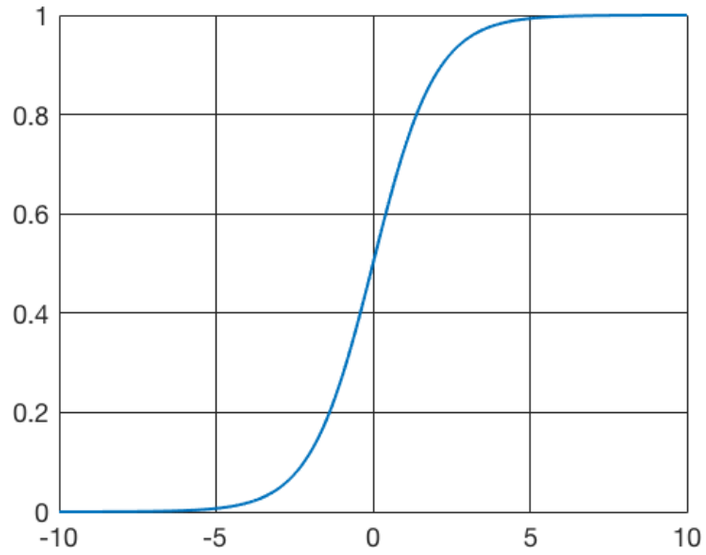
Linear classifier (again)

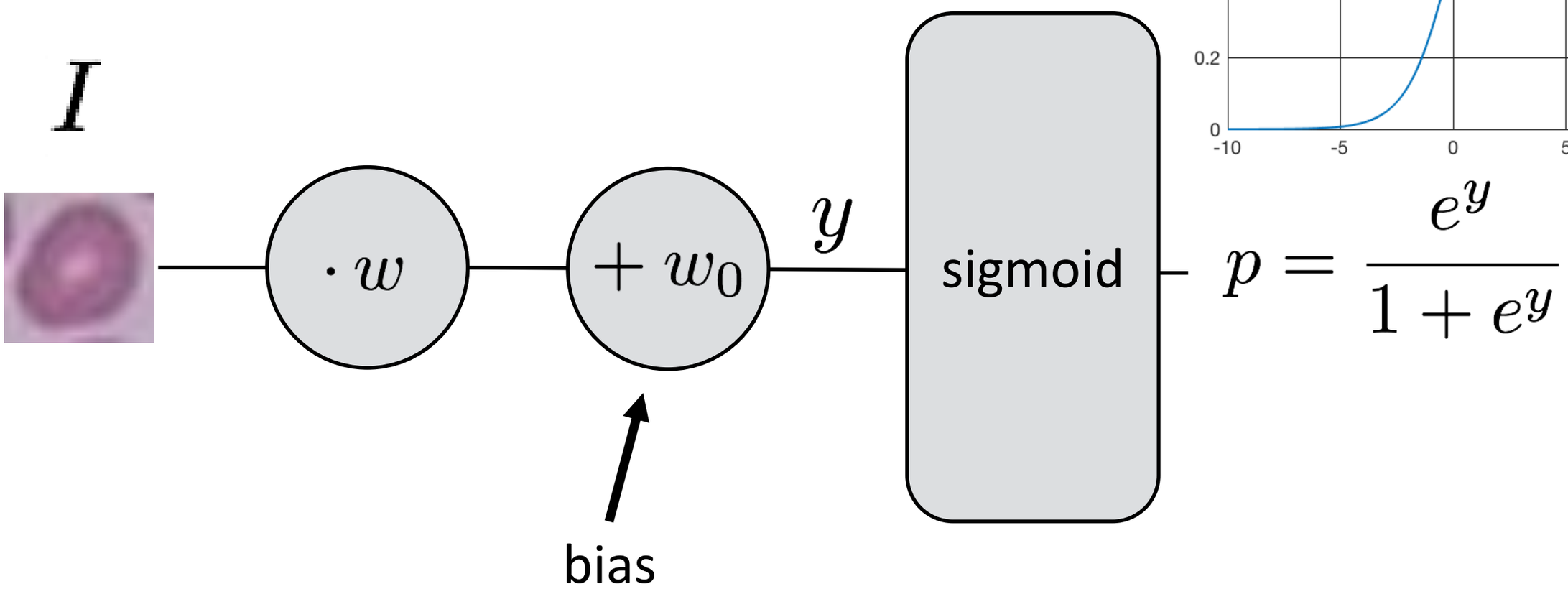
Sigmoid Function



Linear classifier (again)

Sigmoid Function





$$e^y$$

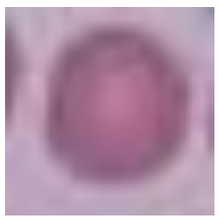
$$p = \frac{e^y}{1 + e^y}$$

Linear classifier (again)

Performance Function

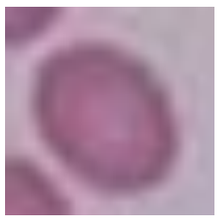
$\longrightarrow p_1$

$\longrightarrow p_4$

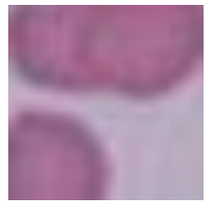


$\longrightarrow p_5$

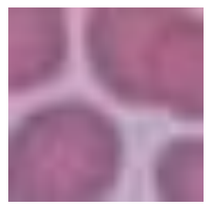
$\longrightarrow p_6$



$\longrightarrow p_8$



$\longrightarrow p_2$



$\longrightarrow p_3$

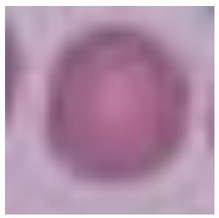
$\longrightarrow p_7$

$\longrightarrow p_9$

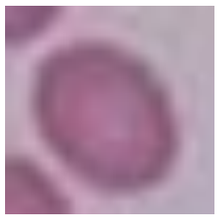
Performance Function

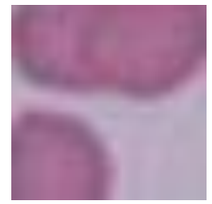

$\longrightarrow p_1$


$\longrightarrow p_4$


$\longrightarrow p_5$

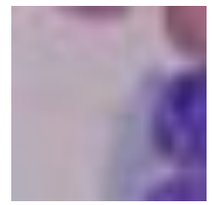

$\longrightarrow p_6$


$\longrightarrow p_8$


$\longrightarrow p_2$


$\longrightarrow p_3$


$\longrightarrow p_7$

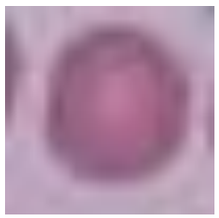

$\longrightarrow p_9$

How to measure quality of a classifier?

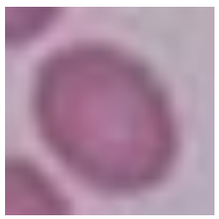
Performance Function: Likelihood

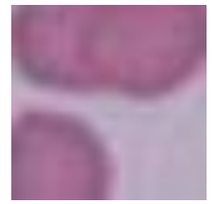

$\longrightarrow p_1$


$\longrightarrow p_4$


$\longrightarrow p_5$

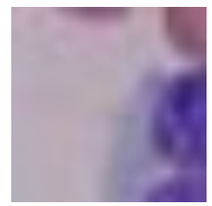

$\longrightarrow p_6$


$\longrightarrow p_8$


$\longrightarrow p_2$


$\longrightarrow p_3$


$\longrightarrow p_7$

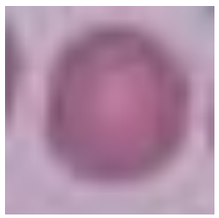

$\longrightarrow p_9$

$$\prod_{\text{positive examples}} p_i \cdot \prod_{\text{negative examples}} (1 - p_i)$$

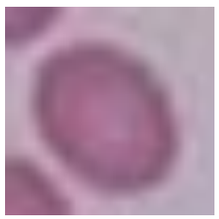
Performance Function: Log-Likelihood

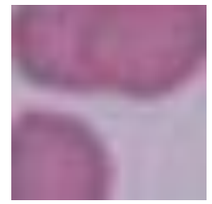

$\longrightarrow p_1$


$\longrightarrow p_4$


$\longrightarrow p_5$

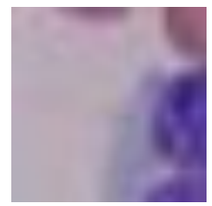

$\longrightarrow p_6$


$\longrightarrow p_8$


$\longrightarrow p_2$


$\longrightarrow p_3$


$\longrightarrow p_7$


$\longrightarrow p_9$

$$\sum \ln p_i \quad + \quad \sum \ln (1 - p_i)$$

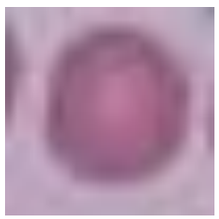
positive examples

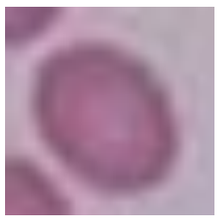
negative examples

Performance Function: Negative Log-Likelihood

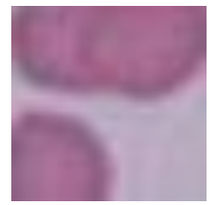
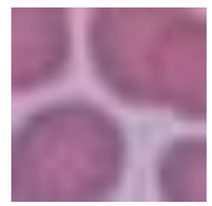
Loss

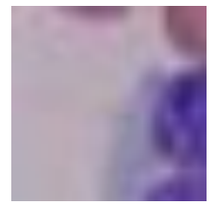






$$p_1 \quad p_4 \quad p_5 \quad p_6 \quad p_8$$



$$p_2 \quad p_3 \quad p_7 \quad p_9$$

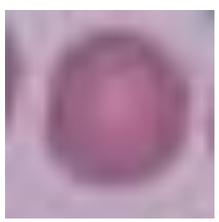
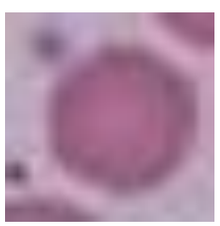
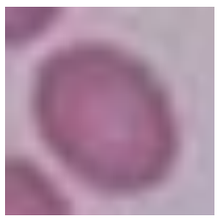
$$-\left(\sum \ln p_i \right) + \sum \ln (1 - p_i)$$

positive examples

negative examples

Performance Function: Negative Log-Likelihood

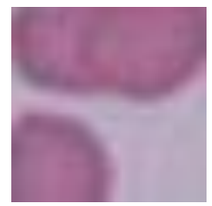
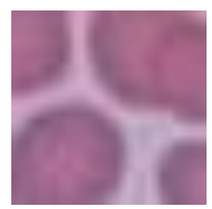


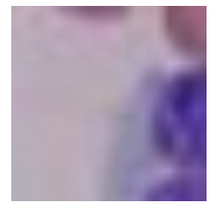




p_1
p_4
p_5
p_6
p_8

Loss

minimize



p_2
p_3
p_7
p_9

$$-\left(\sum \ln p_i \right) + \sum \ln (1 - p_i)$$

positive examples

negative examples

Multiple Outputs

$$p_k = \frac{e^{y_k}}{\sum_{m=0}^{n} e^{y_m}}$$

softmax function

Optimization

How to train a classifier?

Objective

Maximize likelihood :

$$\max_{\theta} \prod_{i \in S_1} p_i(\theta) \cdot \prod_{i \in S_0} (1 - p_i(\theta))$$

Objective

Maximize likelihood :

$$\max_{\theta} \prod_{i \in S_1} p_i(\theta) \cdot \prod_{i \in S_0} (1 - p_i(\theta))$$

parameters

e.g., θ = $\{w, w_0\}$

Objective

Maximize likelihood :

$$\max_{\theta} \prod_{i \in S_1} p_i(\theta) \cdot \prod_{i \in S_0} (1 - p_i(\theta))$$

parameters
e.g., θ = $\{w, w_0\}$

positive
examples

Objective

Maximize likelihood :

$$\max_{\theta} \prod_{i \in S_1} p_i(\theta) \cdot \prod_{i \in S_0} (1 - p_i(\theta))$$

parameters
e.g., θ = $\{w, w_0\}$

positive examples

negative examples

Objective

Maximize likelihood :

$$\max_{\theta} \prod_{i \in S_1} p_i(\theta) \cdot \prod_{i \in S_0} (1 - p_i(\theta))$$

parameters
e.g., θ = $\{w, w_0\}$

positive examples

negative examples

= minimize negative log-likelihood

$$\min_{\theta} L(\theta) = -\sum_{i \in S_1} \ln(p_i(\theta)) - \sum_{i \in S_0} \ln((1 - p_i(\theta)))$$

Objective

Maximize likelihood :

$$\max_{\theta} \prod_{i \in S_1} p_i(\theta) \cdot \prod_{i \in S_0} (1 - p_i(\theta))$$

parameters
e.g., θ = $\{w, w_0\}$

positive examples

negative examples

= minimize negative log-likelihood

$$\min_{\theta} L(\theta) = -\sum_{i \in S_1} \ln(p_i(\theta)) - \sum_{i \in S_0} \ln((1 - p_i(\theta)))$$

$$= \sum_{i} L_i(\theta)$$

Objective

Maximize likelihood :

$$\max_{\theta} \prod_{i \in S_1} p_i(\theta) \cdot \prod_{i \in S_0} (1 - p_i(\theta))$$

parameters
e.g., θ = $\{w, w_0\}$

positive examples

negative examples

= minimize negative log-likelihood

$$\min_{\theta} L(\theta) = -\sum_{i \in S_1} \ln(p_i(\theta)) - \sum_{i \in S_0} \ln((1 - p_i(\theta)))$$

$$= \sum_{i} L_i(\theta)$$

partial loss of i^{th} data point

Gradient Descent

Local instead of global optimization:

$$\nabla L(\theta) = \sum_i \nabla L_i(\theta)$$

Gradient Descent

Local instead of global optimization:

$$\nabla L(\theta) = \sum_i \nabla L_i(\theta)$$

Update rule:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \nabla L(\theta) = \theta^{(k)} - \mu \sum_i \nabla L_i(\theta)$$

Gradient Descent

Local instead of global optimization:

$$\nabla L(\theta) = \sum_i \nabla L_i(\theta)$$

Update rule:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \nabla L(\theta) = \theta^{(k)} - \mu \sum_i \nabla L_i(\theta)$$

learning rate

Stochastic Gradient Descent

Larger number of training examples = complex optimization

Stochastic Gradient Descent

Larger number of training examples = complex optimization
Use only random subset of data for update, e.g., single point:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \sum_i \nabla L_i(\theta) \approx \theta^{(k)} - \mu \nabla L_i(\theta)$$

Stochastic Gradient Descent

Larger number of training examples = complex optimization
Use only random subset of data for update, e.g., single point:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \sum_i \nabla L_i(\theta) \approx \theta^{(k)} - \mu \nabla L_i(\theta)$$

Better: Use a few data points (**mini batch**)

$$\theta^{(k+1)} \approx \theta^{(k)} - \mu \frac{1}{m} \sum_j \nabla L_j(\theta)$$

Stochastic Gradient Descent

Larger number of training examples = complex optimization
Use only random subset of data for update, e.g., single point:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \sum_i \nabla L_i(\theta) \approx \theta^{(k)} - \mu \nabla L_i(\theta)$$

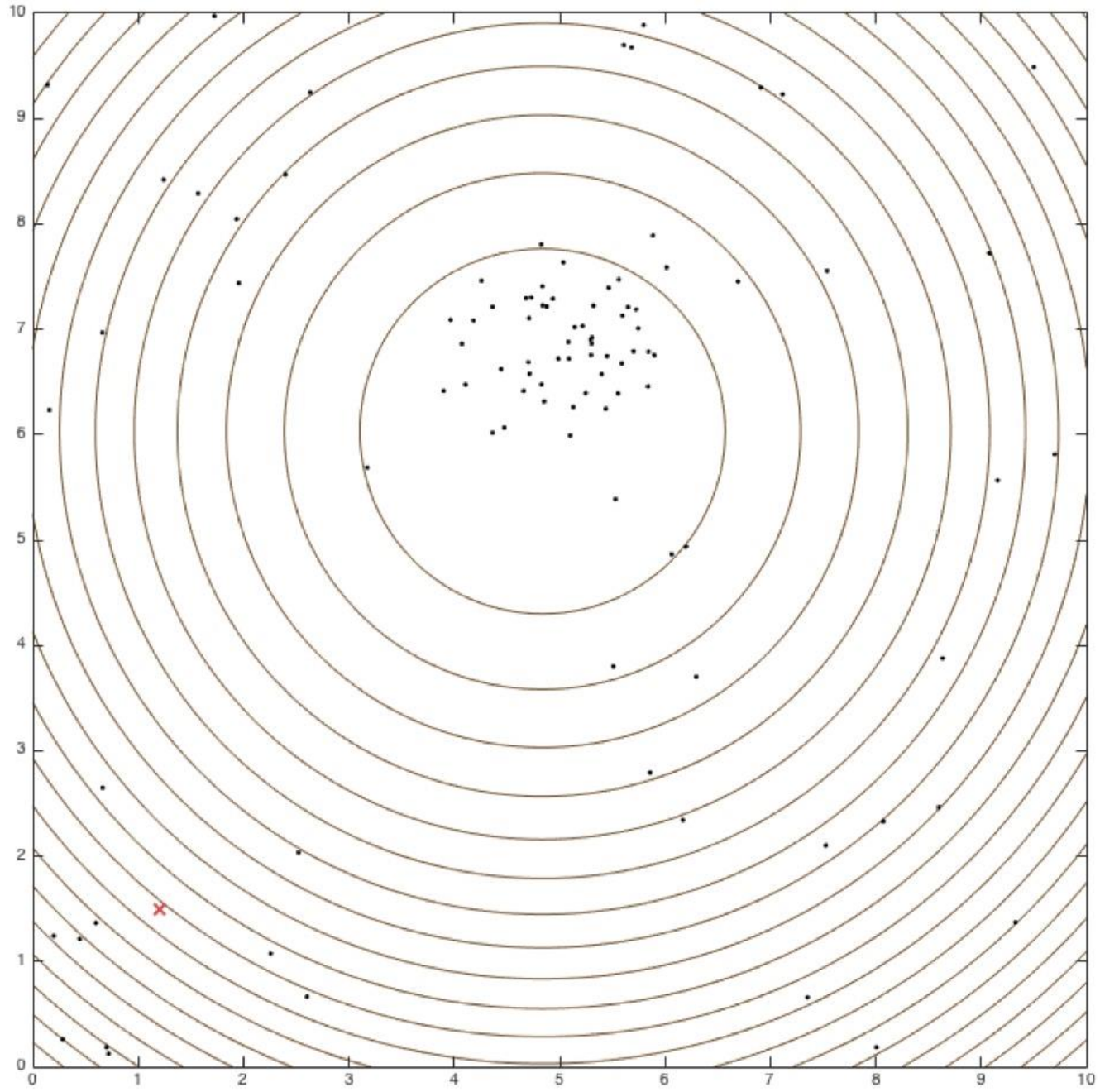
Better: Use a few data points (**mini batch**)

$$\theta^{(k+1)} \approx \theta^{(k)} - \mu \frac{1}{m} \sum_j \nabla L_j(\theta)$$

average over #points
in mini batch

Simple 2D Example

Compute 2D location θ from data points (Gaussian noise)



Simple 2D Example

Compute 2D location θ from data points (Gaussian noise)

Residual per measurement: $r_i(\theta) = x_i - \theta$

Simple 2D Example

Compute 2D location θ from data points (Gaussian noise)

Residual per measurement: $r_i(\theta) = x_i - \theta$

Likelihood: $\prod_i e^{\frac{r_i(\theta)^2}{2\sigma^2}} = \prod_i e^{\frac{(x_i-\theta)^2}{2\sigma^2}}$

Simple 2D Example

Compute 2D location θ from data points (Gaussian noise)

Residual per measurement: $r_i(\theta) = x_i - \theta$

Likelihood: $\prod_i e^{\frac{r_i(\theta)^2}{2\sigma^2}} = \prod_i e^{\frac{(x_i-\theta)^2}{2\sigma^2}}$

$$\ln\left(e^{\frac{(x_i-\theta)^2}{2\sigma^2}}\right) = \frac{(x_i-\theta)^2}{2\sigma^2}$$

Simple 2D Example

Compute 2D location θ from data points (Gaussian noise)

Residual per measurement: $r_i(\theta) = x_i - \theta$

Likelihood: $\prod_i e^{\frac{r_i(\theta)^2}{2\sigma^2}} = \prod_i e^{\frac{(x_i-\theta)^2}{2\sigma^2}}$

$$\ln\left(e^{\frac{(x_i-\theta)^2}{2\sigma^2}}\right) = \frac{(x_i-\theta)^2}{2\sigma^2}$$

Negative Log-Likelihood: $L(\theta) = \sum_i (x_i-\theta)^2 = \sum_i L_i(\theta)$

Simple 2D Example

Objective: $$\min_{\theta} L(\theta) = \sum_i (x_i - \theta)^2 = \sum_i L_i(\theta)$$

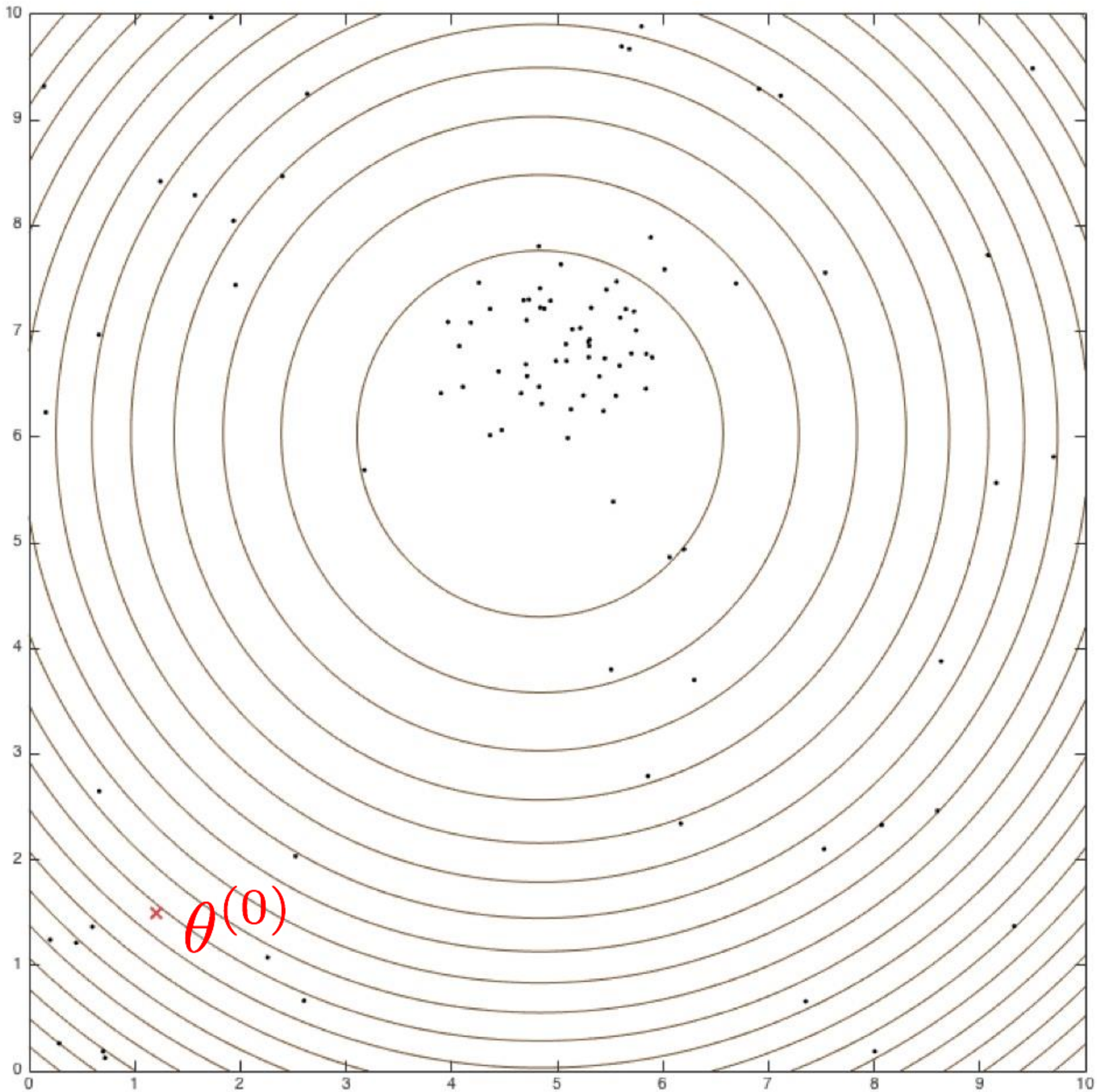
Simple 2D Example

Objective: $$\min_{\theta} L(\theta) = \sum_i (x_i - \theta)^2 = \sum_i L_i(\theta)$$

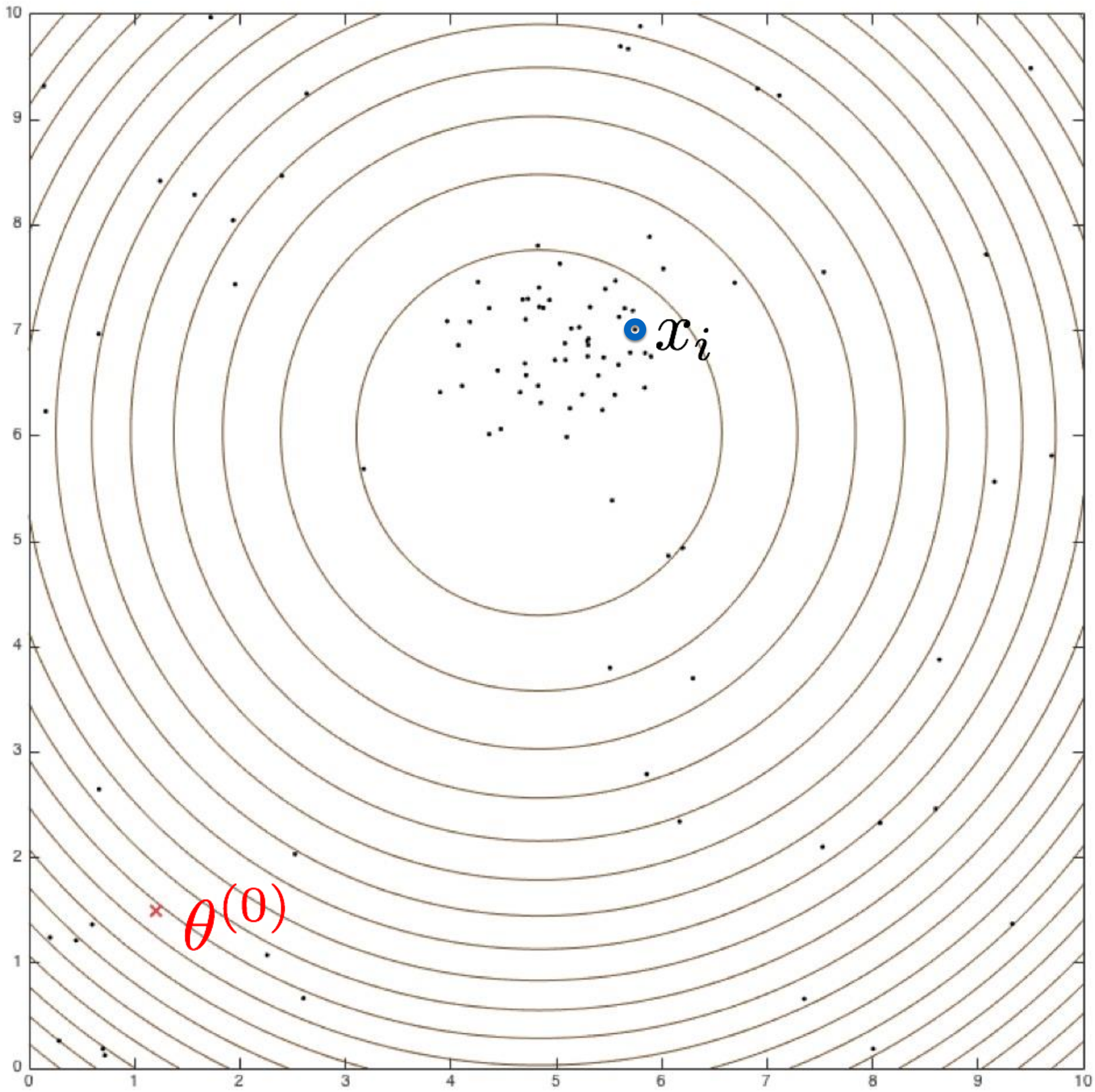
Gradient:

$$\nabla L(\theta) = -\sum_i 2 \cdot (x_i - \theta)$$

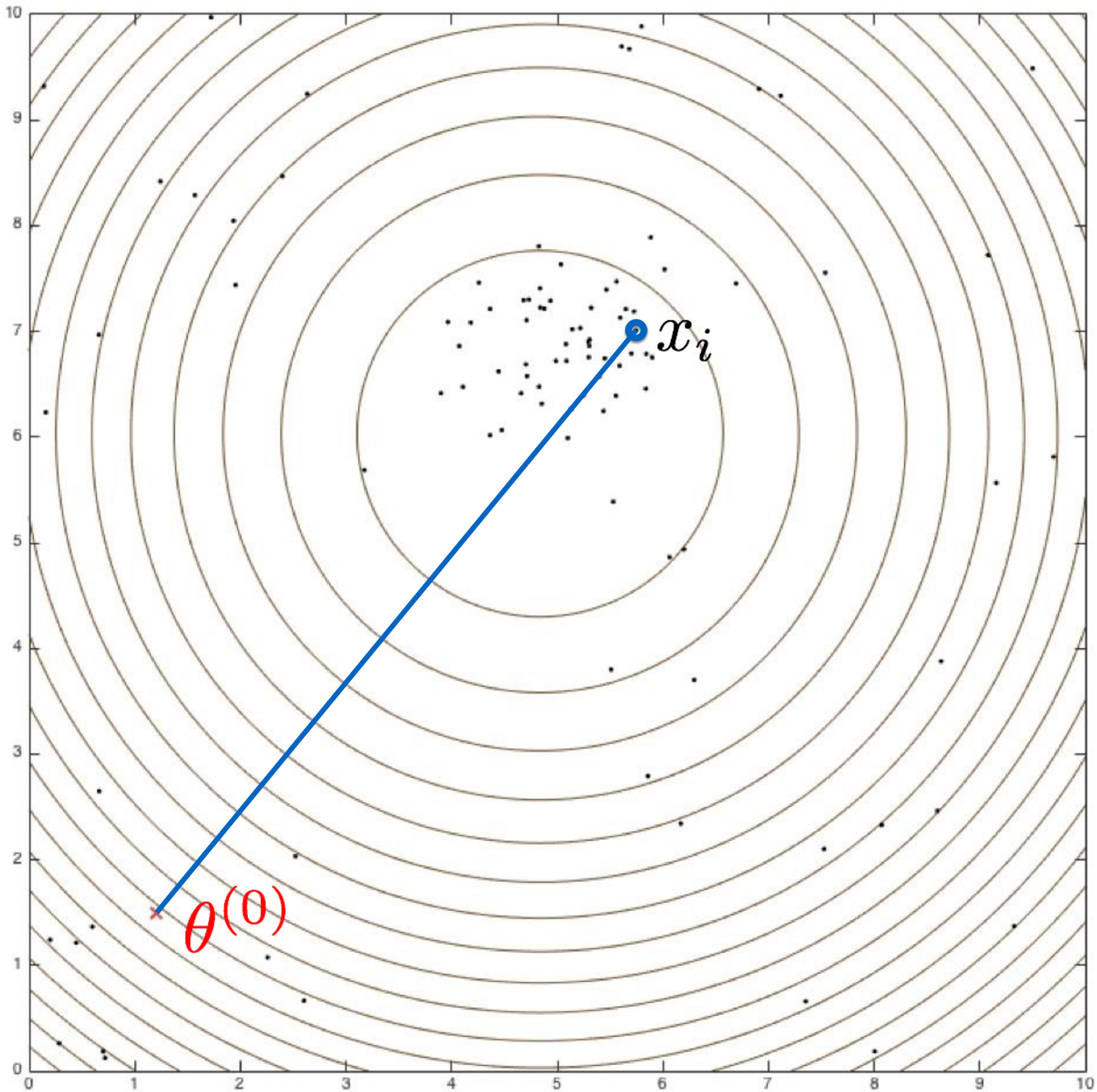
Stochastic Gradient Descent



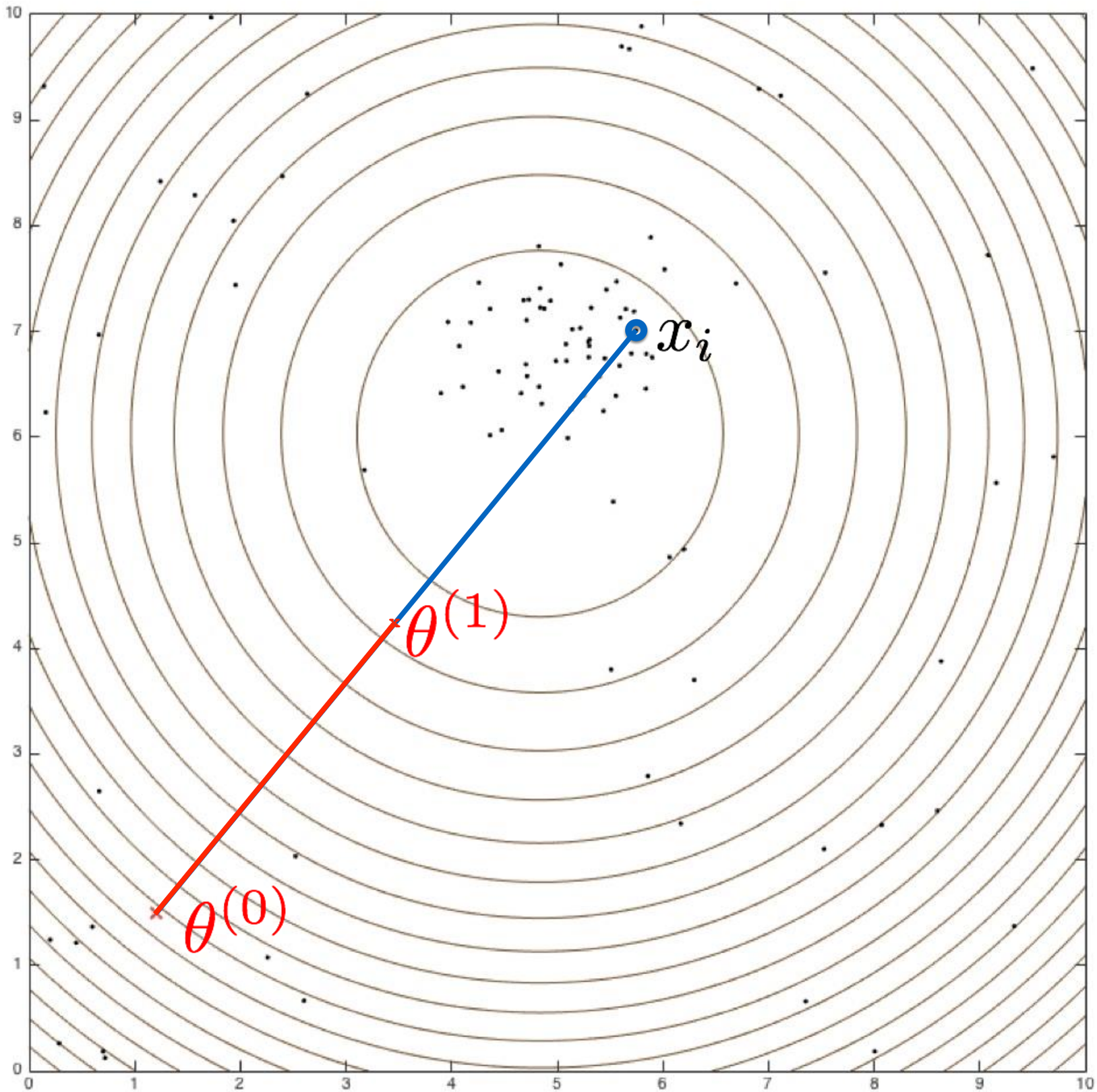
Stochastic Gradient Descent



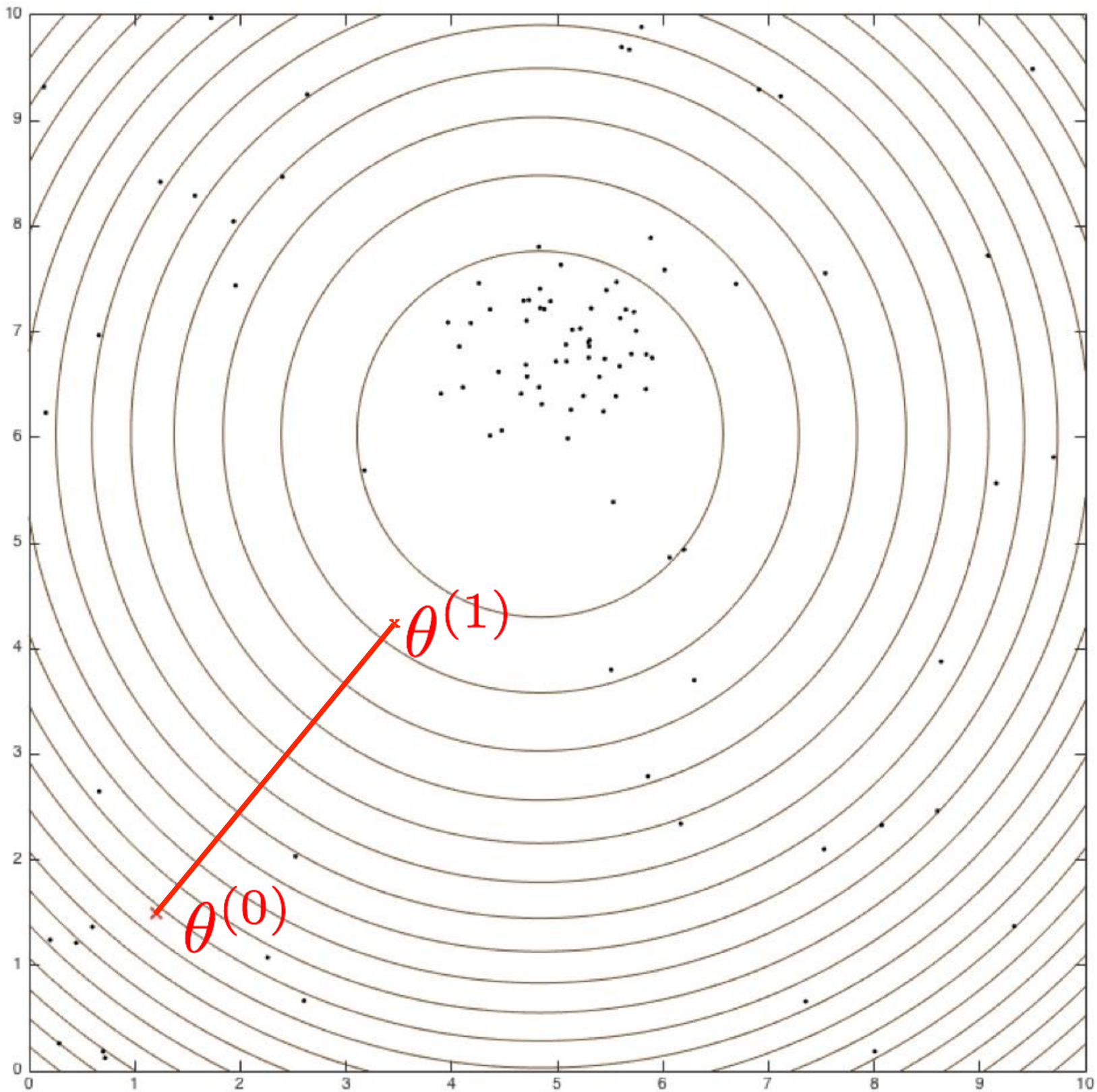
Stochastic Gradient Descent



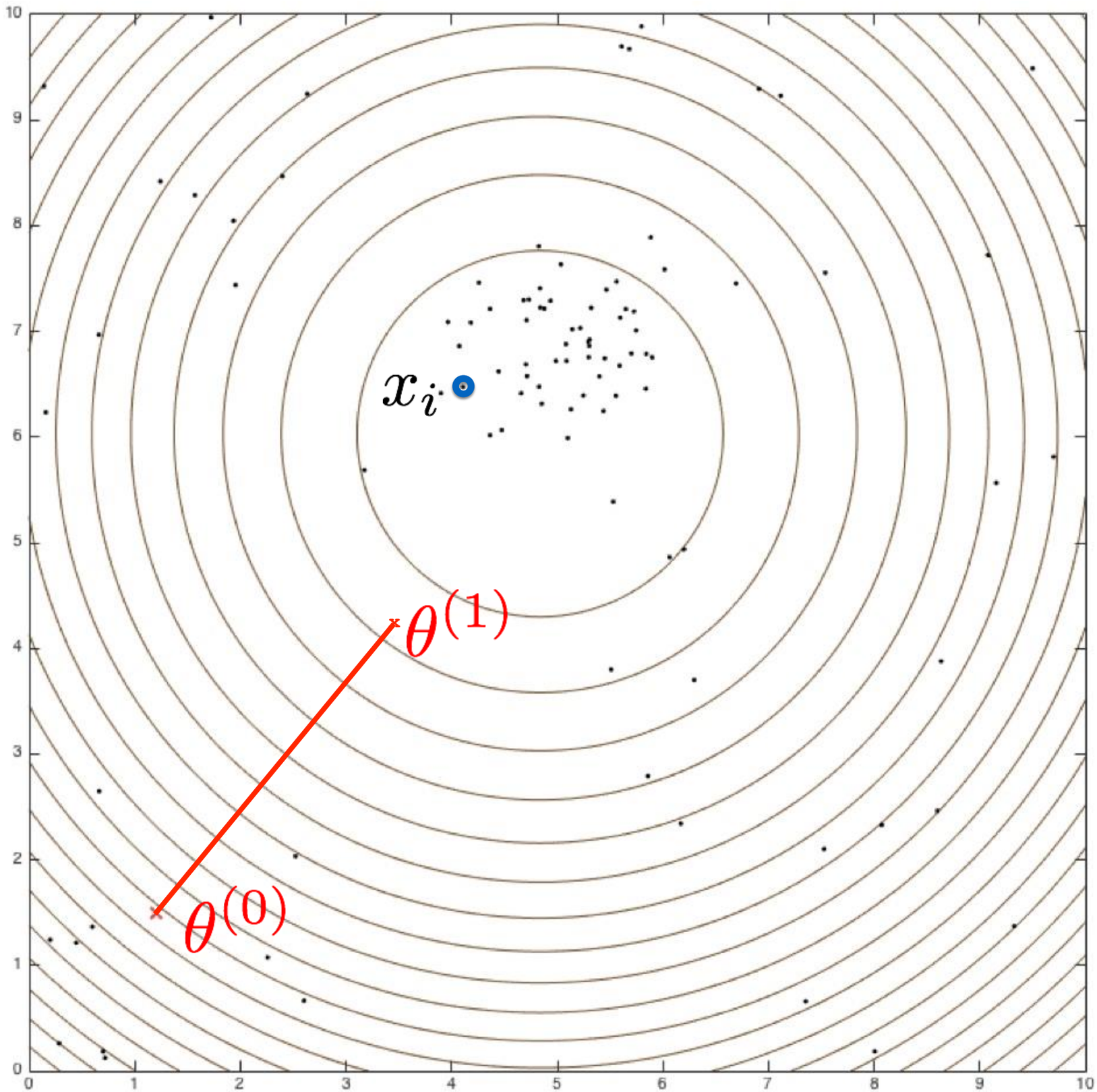
Stochastic Gradient Descent



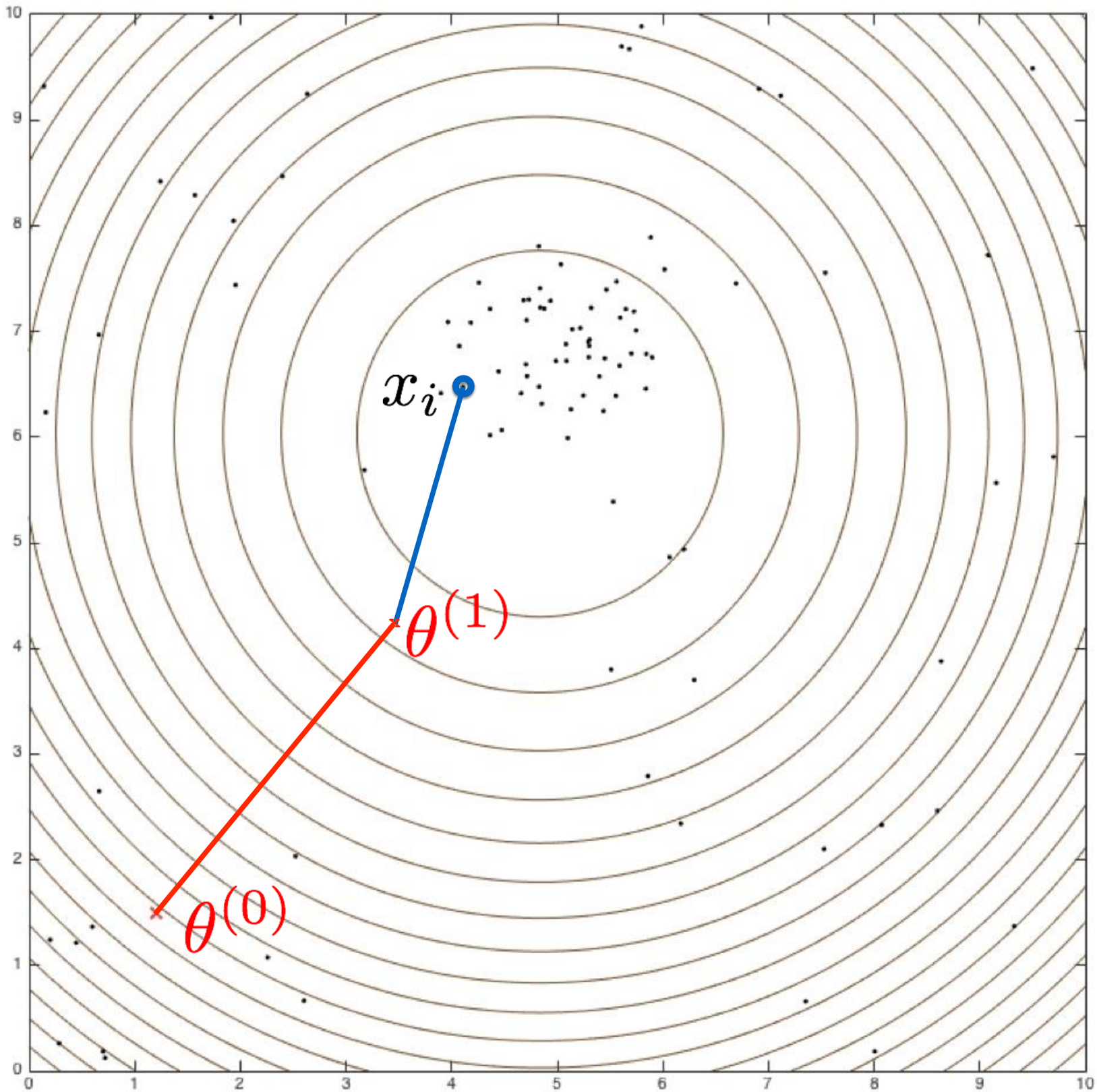
Stochastic Gradient Descent



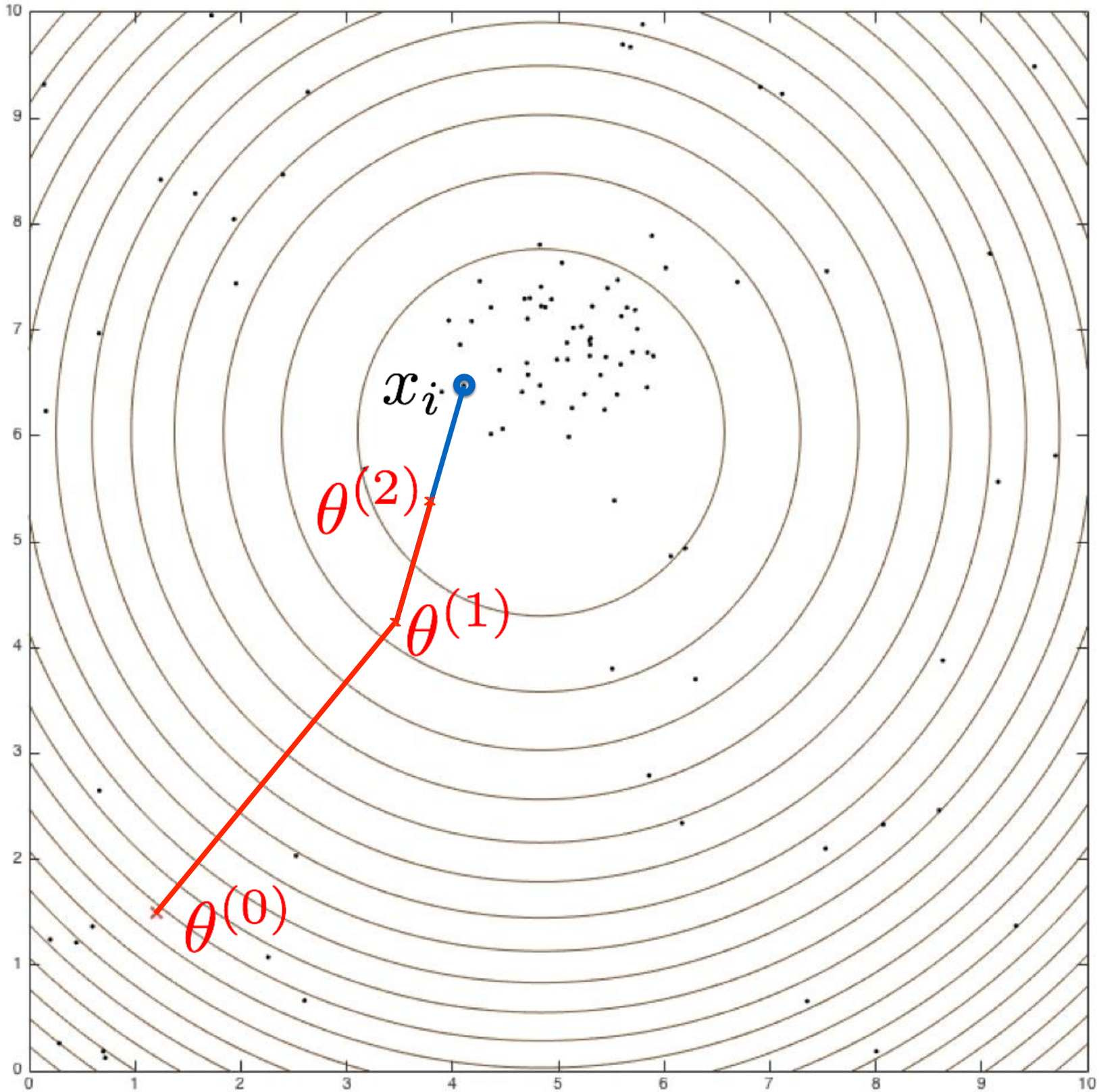
Stochastic Gradient Descent



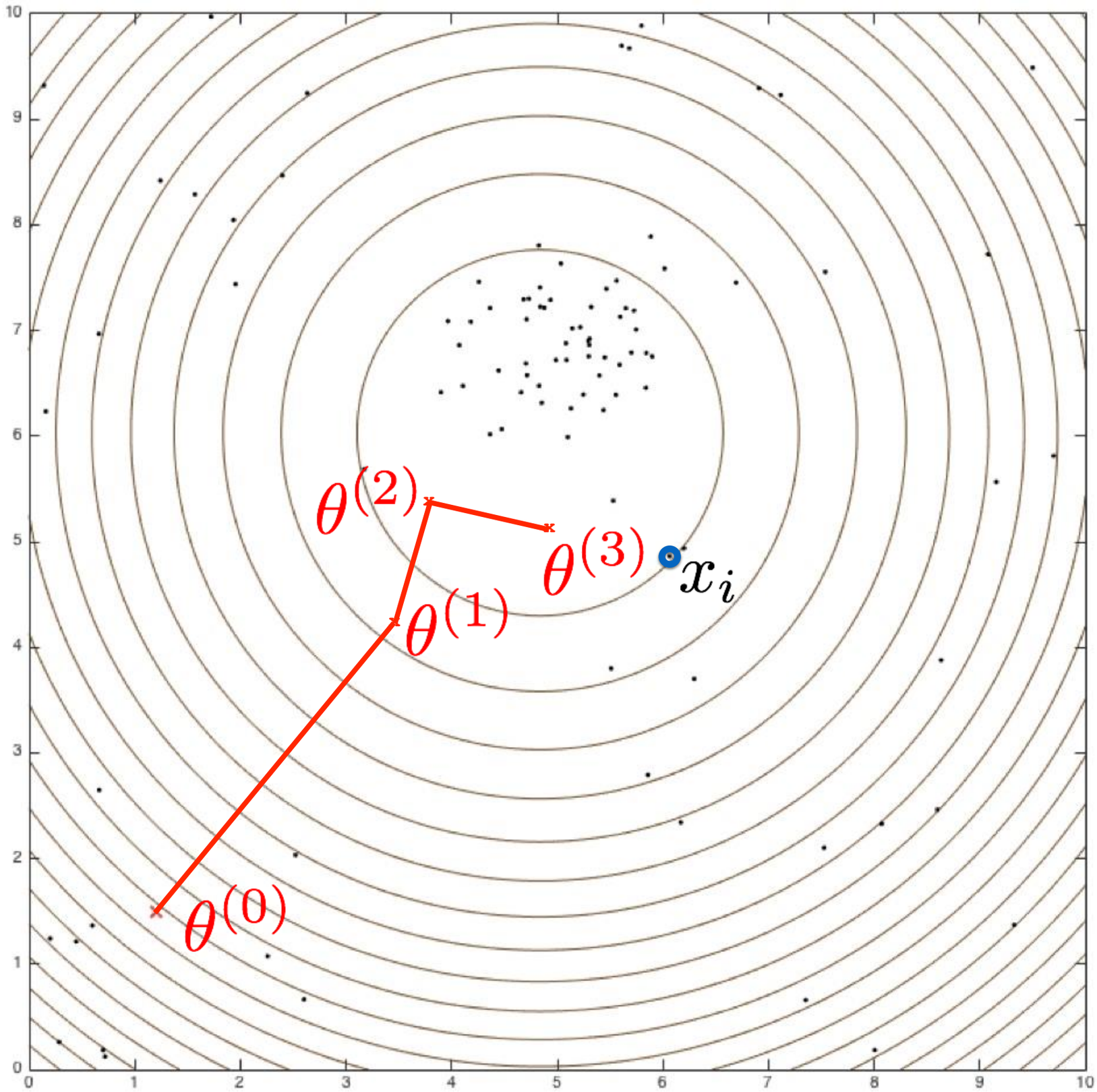
Stochastic Gradient Descent



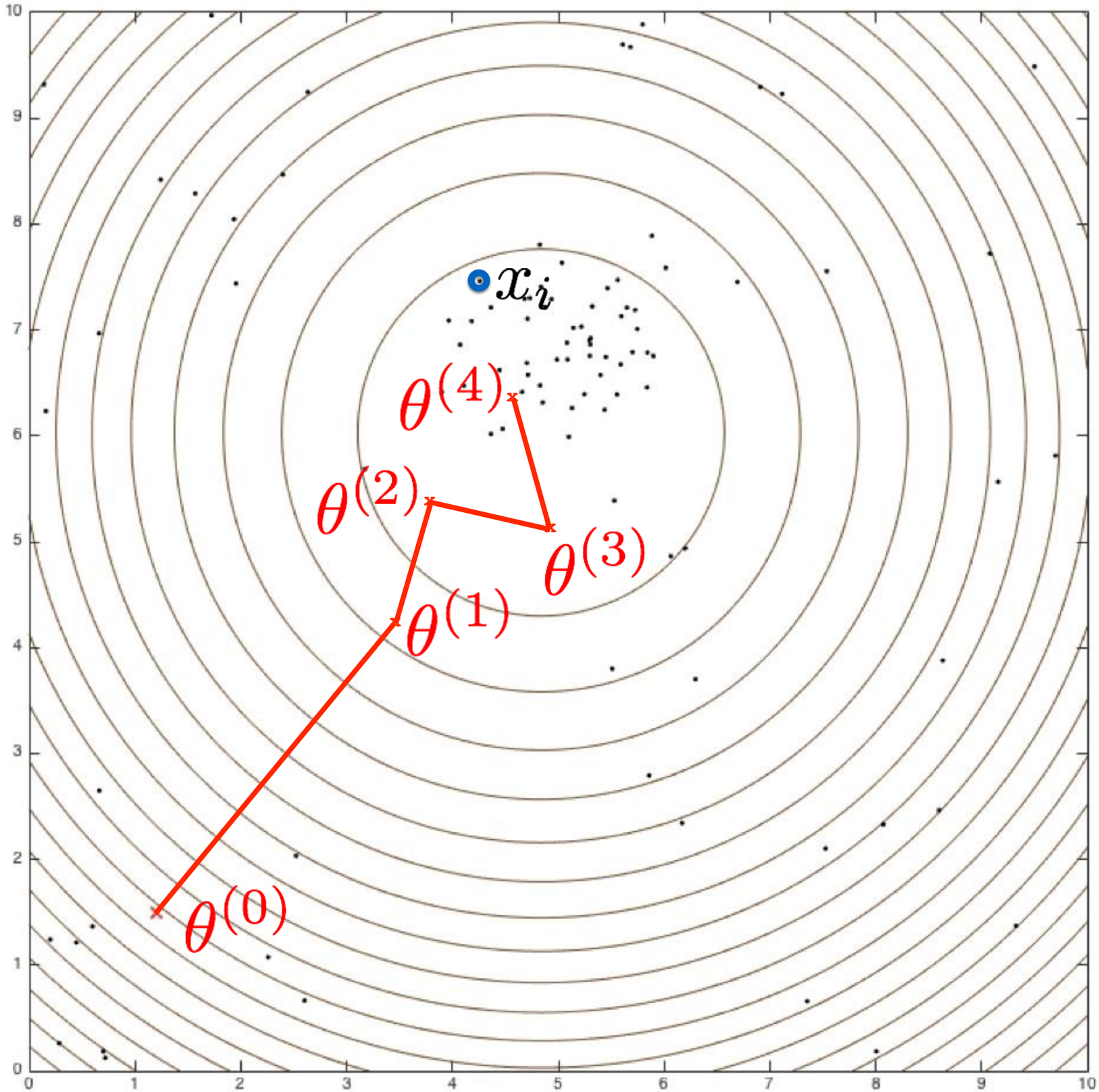
Stochastic Gradient Descent



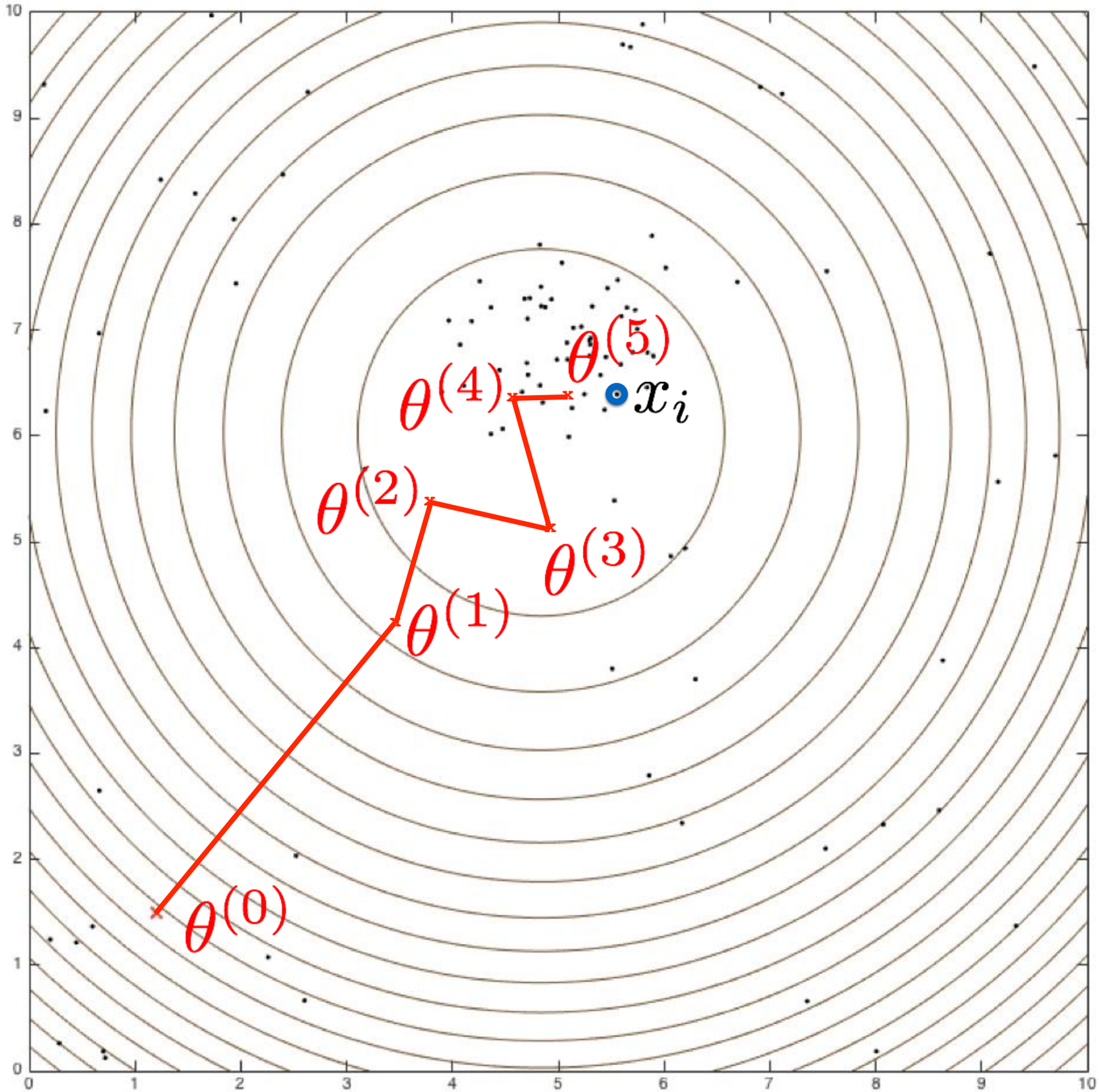
Stochastic Gradient Descent



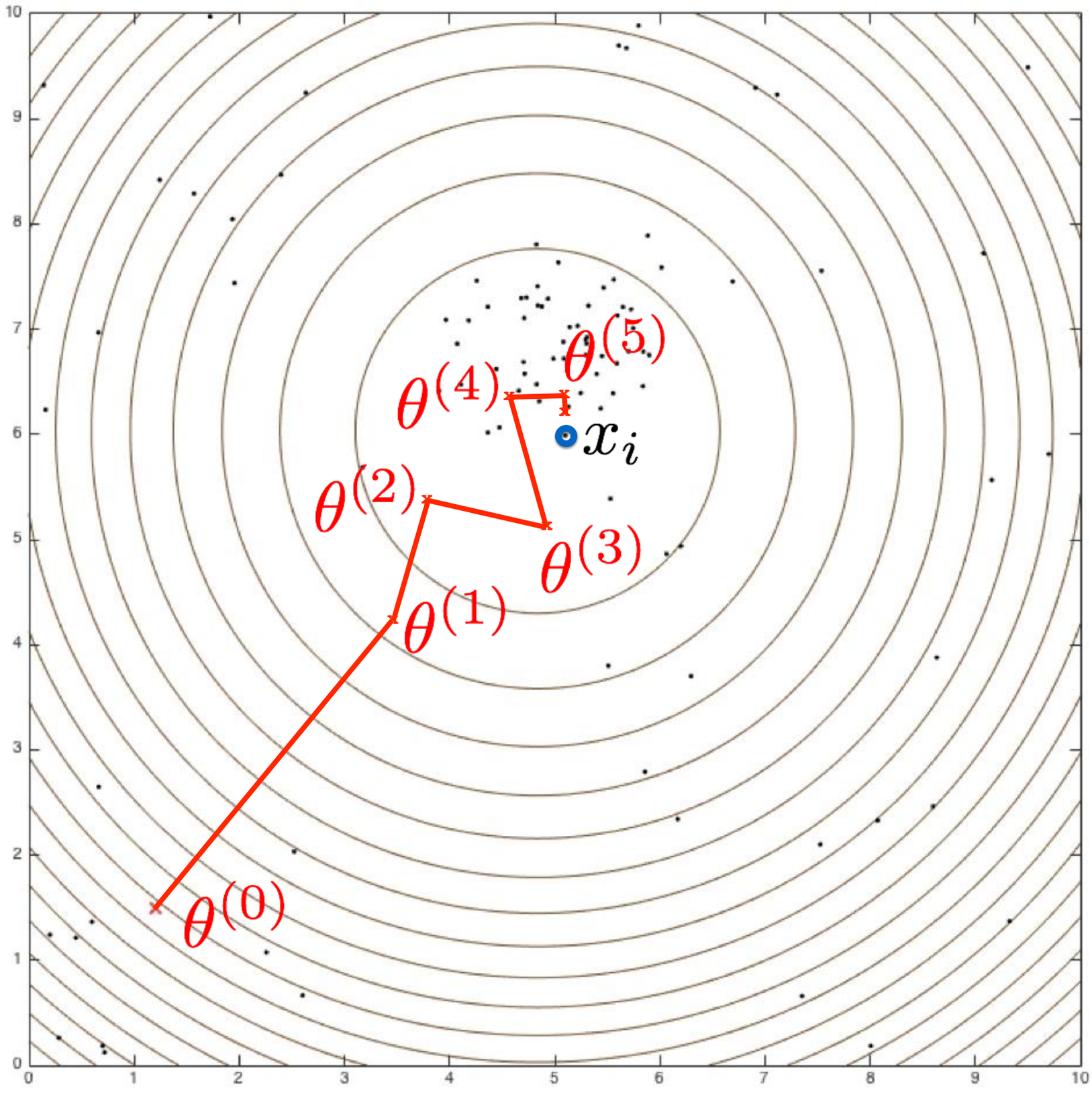
Stochastic Gradient Descent



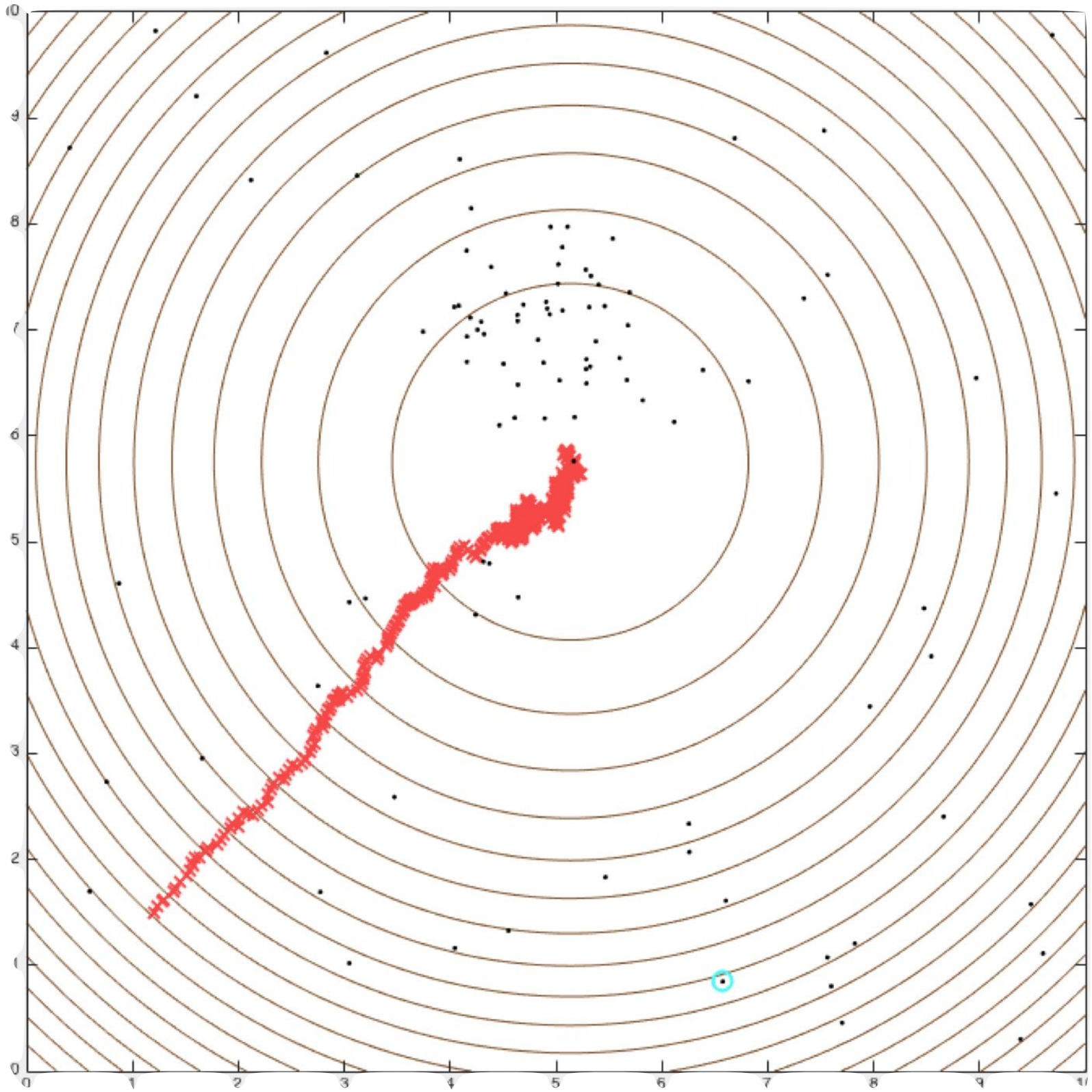
Stochastic Gradient Descent



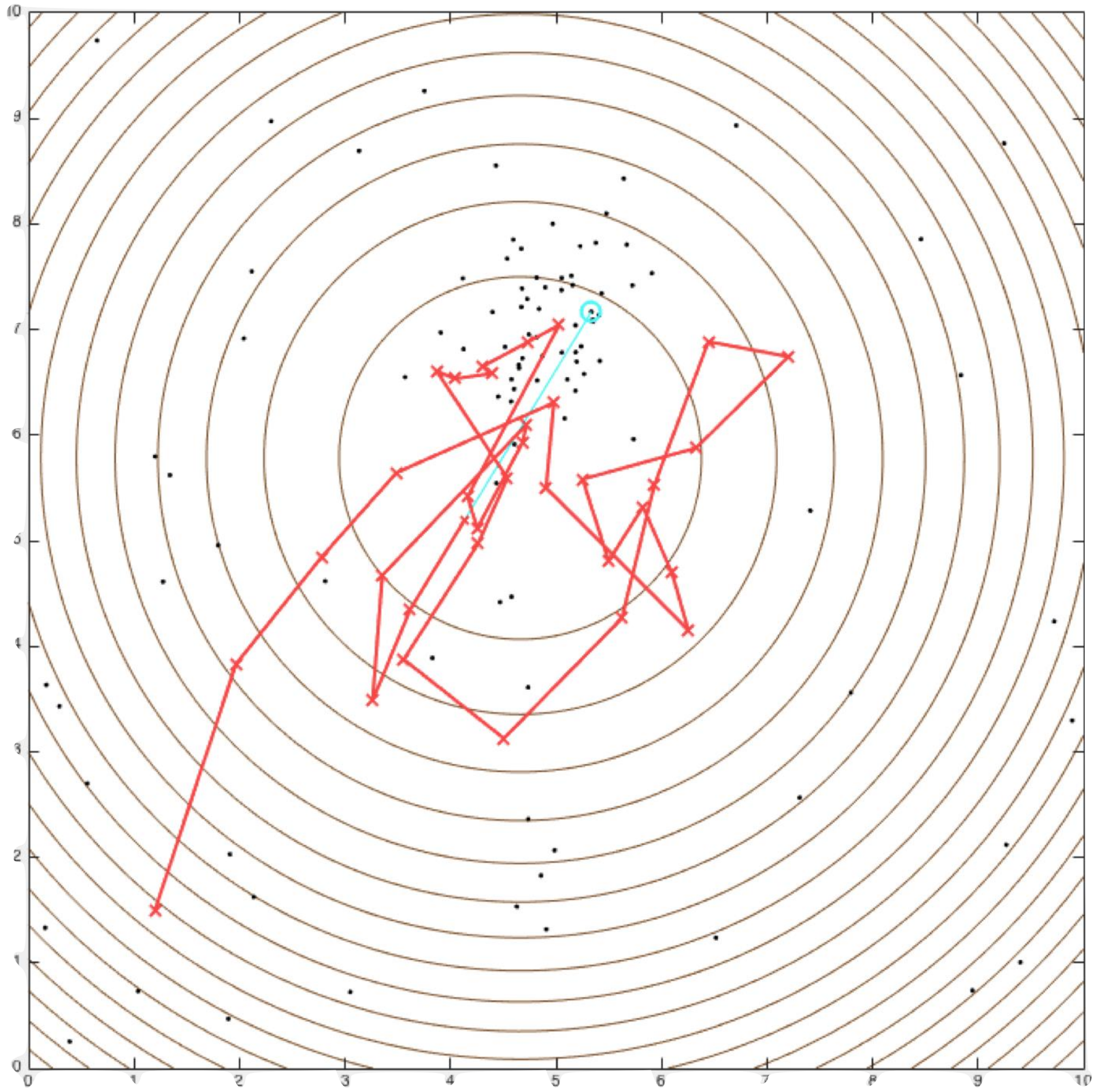
Stochastic Gradient Descent



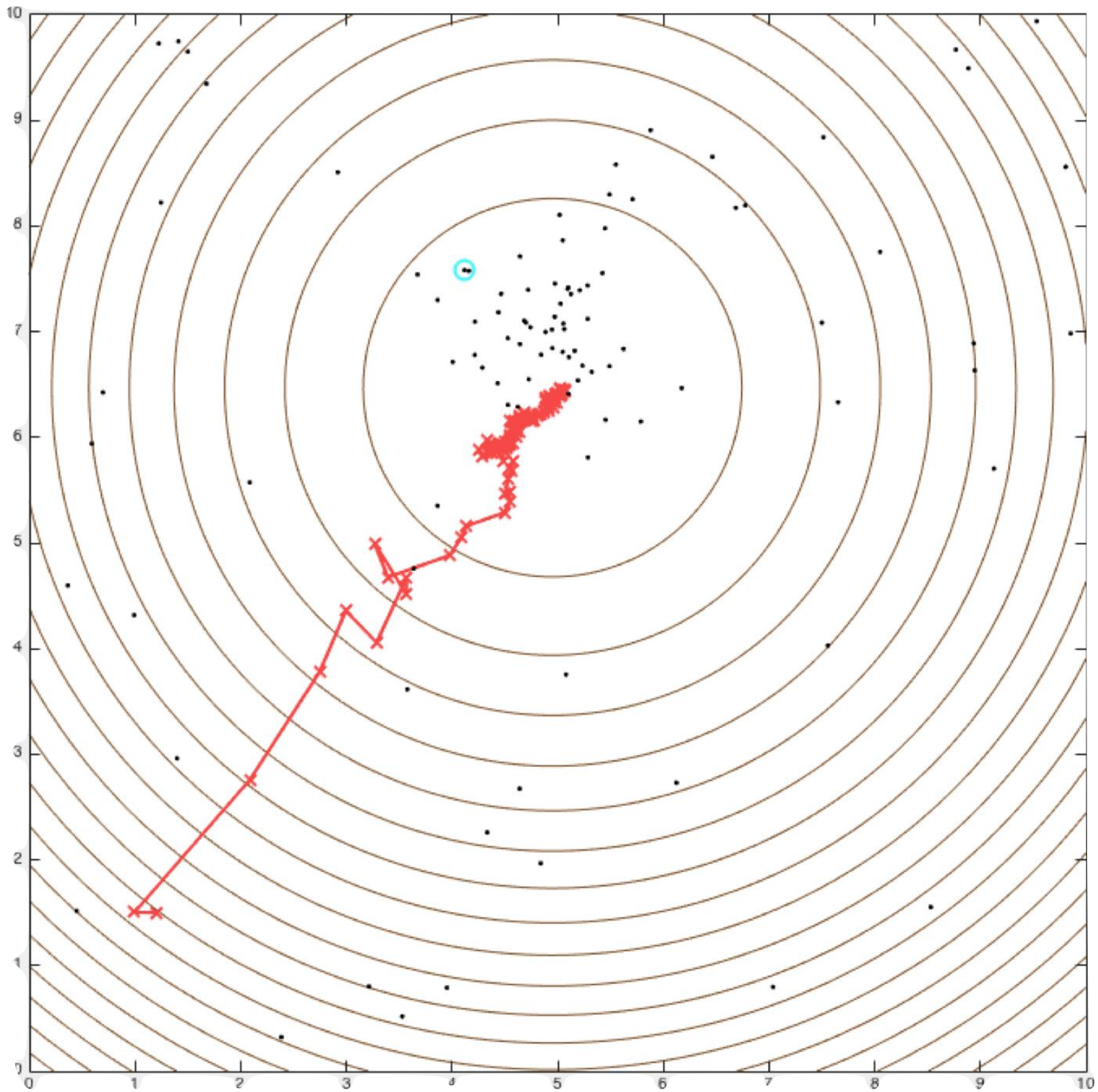
Small Learning Rate



Large Learning Rate



Decreasing Learning Rate



Decrease learning rate over time.

Stochastic Gradient Descent with Momentum

Stochastic gradient descent:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \nabla L_i(\theta)$$

Stochastic Gradient Descent with Momentum

Stochastic gradient descent:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \nabla L_i(\theta)$$

Stochastic gradient descent with momentum:

$$\theta^{(k+1)} = \theta^{(k)} - \mu g^{(k)}$$

Stochastic Gradient Descent with Momentum

Stochastic gradient descent:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \nabla L_i(\theta)$$

Stochastic gradient descent with momentum:

$$\theta^{(k+1)} = \theta^{(k)} - \mu g^{(k)}$$

$$g^{(k)} = \gamma g^{(k-1)} + (1-\gamma) \nabla L_i(\theta^{(k)})$$

Stochastic Gradient Descent with Momentum

Stochastic gradient descent:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \nabla L_i(\theta)$$

Stochastic gradient descent with momentum:

$$\theta^{(k+1)} = \theta^{(k)} - \mu g^{(k)}$$

$$g^{(k)} = \gamma g^{(k-1)} + (1-\gamma) \nabla L_i(\theta^{(k)})$$

momentum

Stochastic Gradient Descent with Momentum

Stochastic gradient descent:

$$\theta^{(k+1)} = \theta^{(k)} - \mu \nabla L_i(\theta)$$

Stochastic gradient descent with momentum:

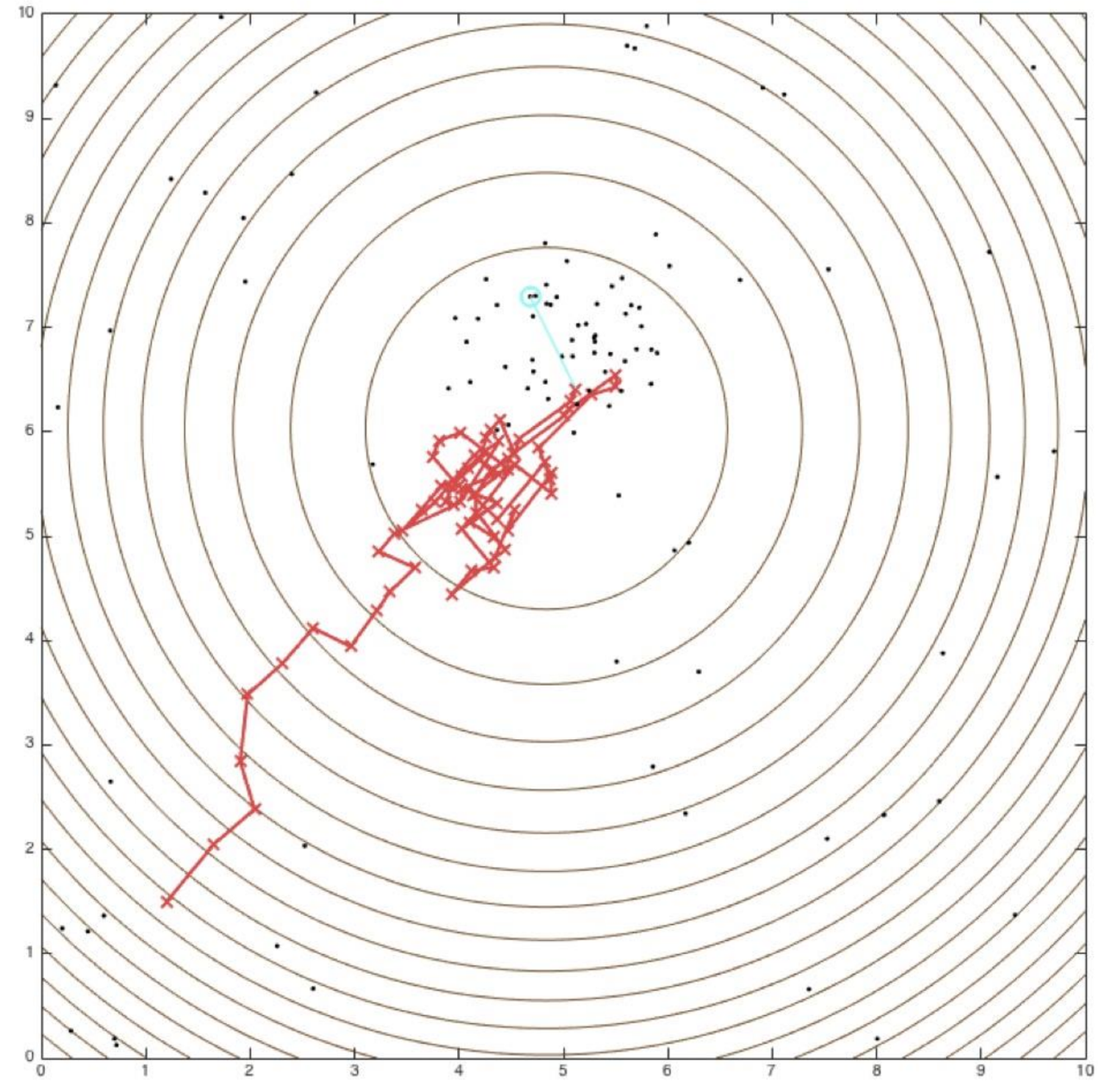
$$\theta^{(k+1)} = \theta^{(k)} - \mu g^{(k)}$$

$$g^{(k)} = \gamma g^{(k-1)} + (1 - \gamma) \nabla L_i(\theta^{(k)})$$

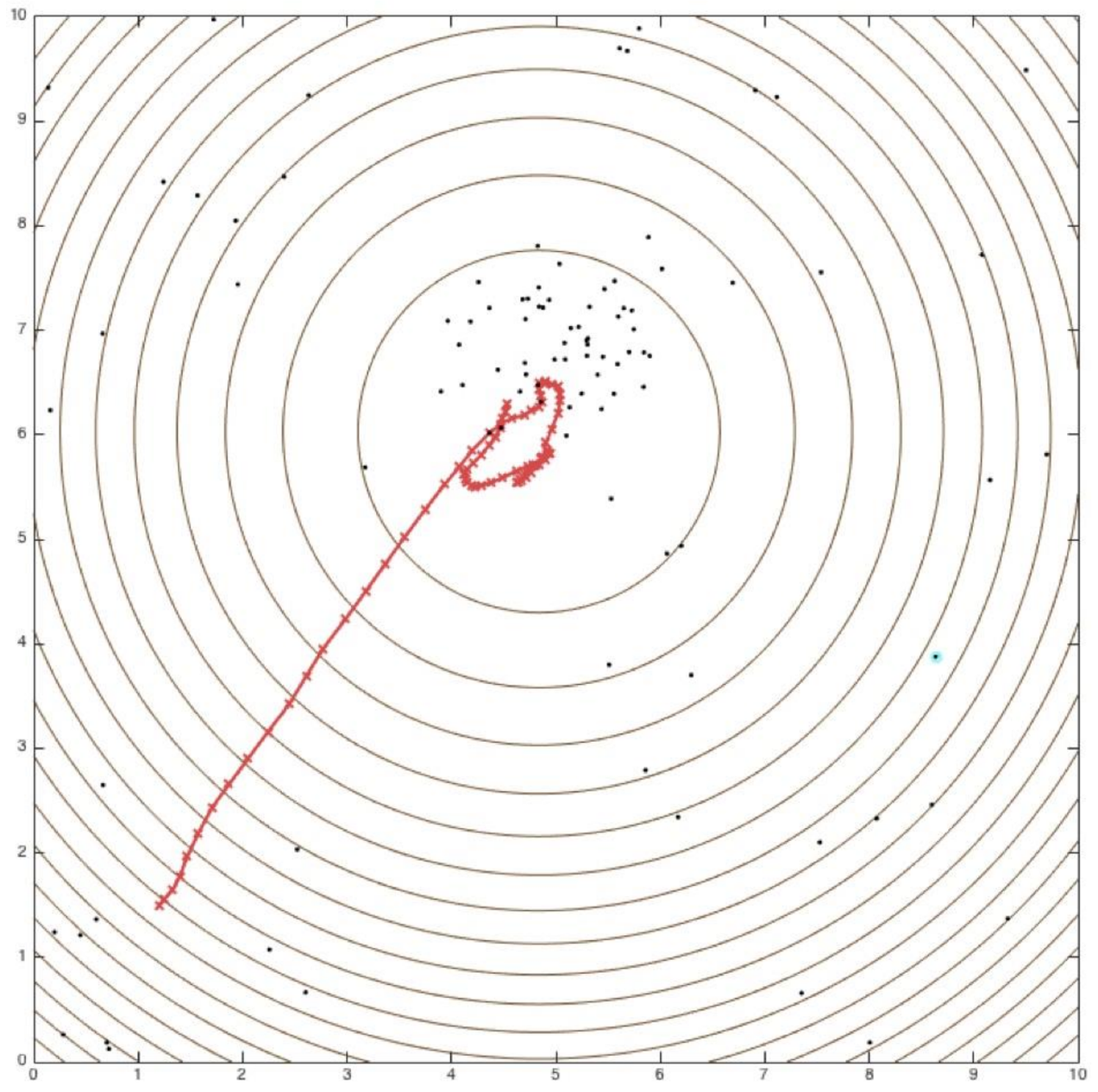
momentum

sometimes not used

Stochastic Gradient Descent with Momentum



without momentum



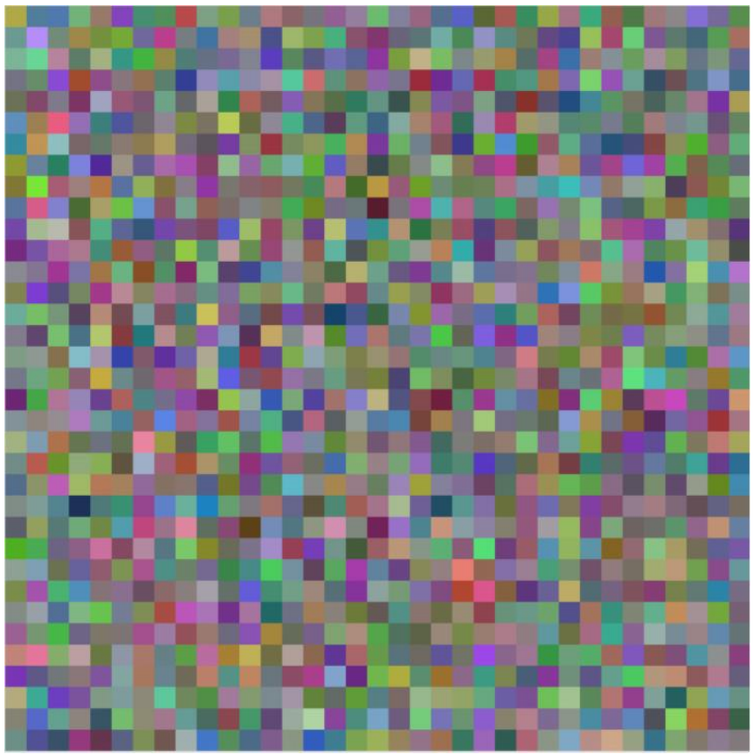
with momentum

More on Optimization

See Chapter 8 of [Goodfellow, Bengio, Courville, Deep Learning, MIT Press 2016]

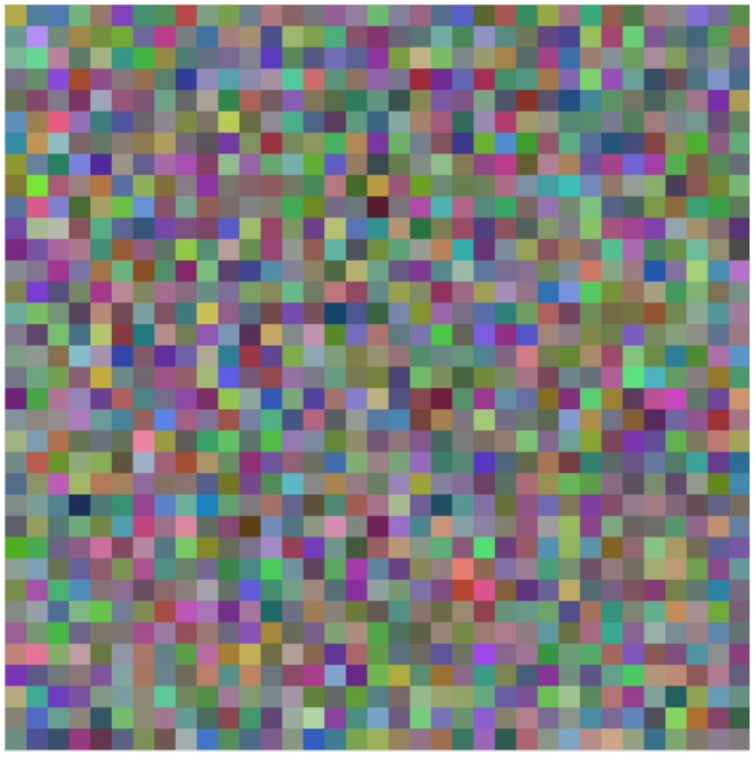
Back to learning a linear classifier …

Linear Cell Classifier

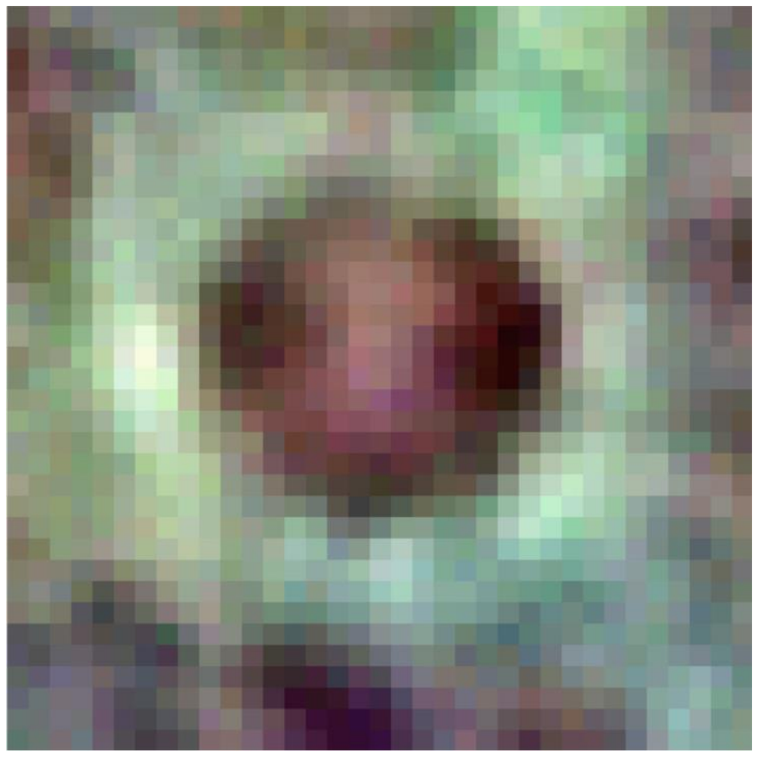


Initialization

Linear Cell Classifier

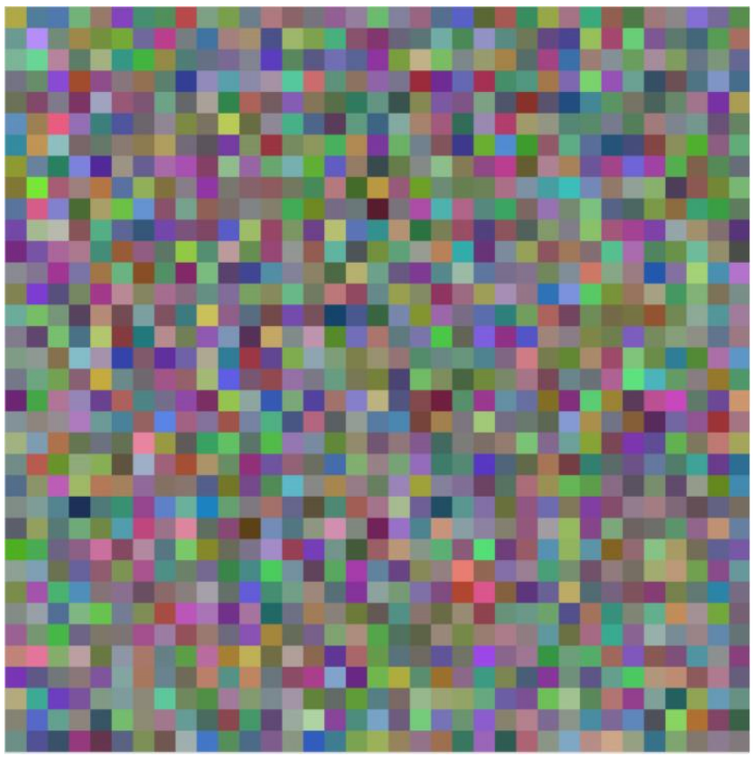


Initialization

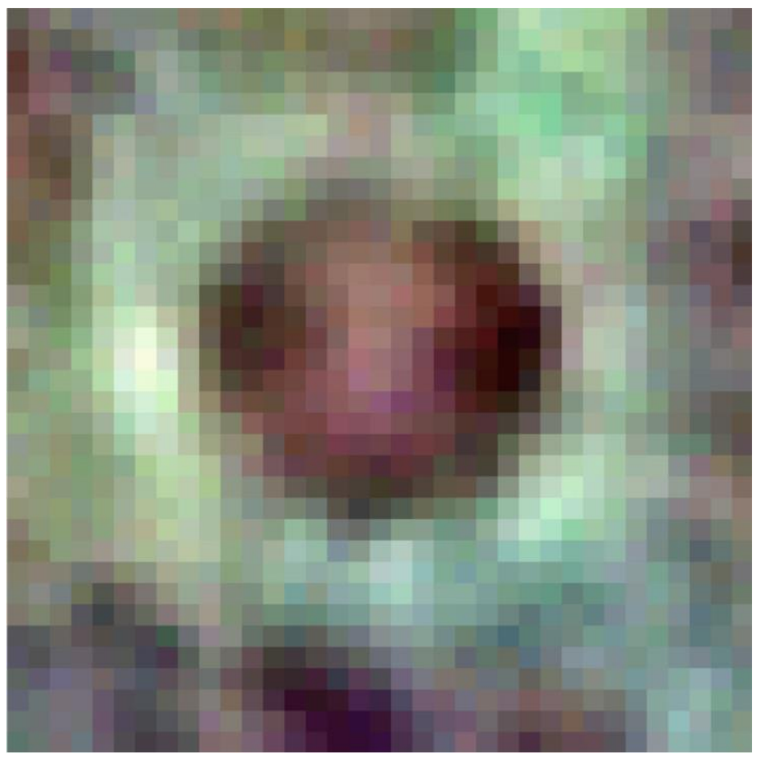


50 iterations

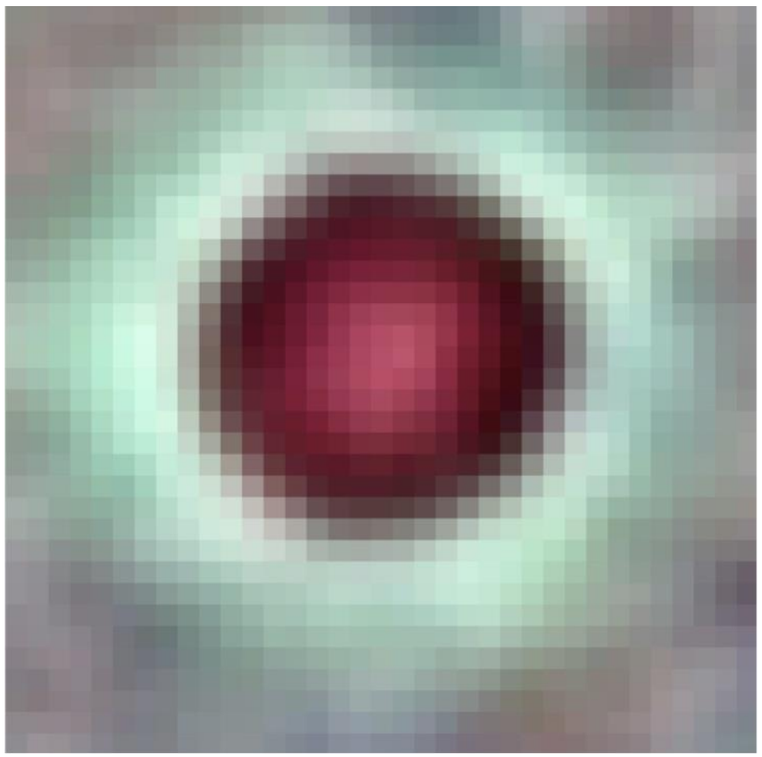
Linear Cell Classifier



Initialization

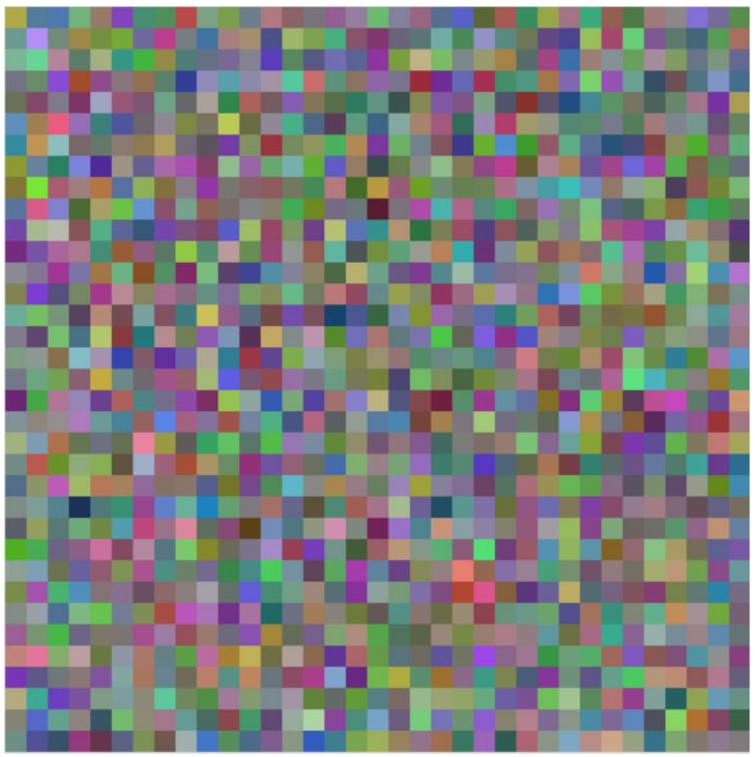


50 iterations

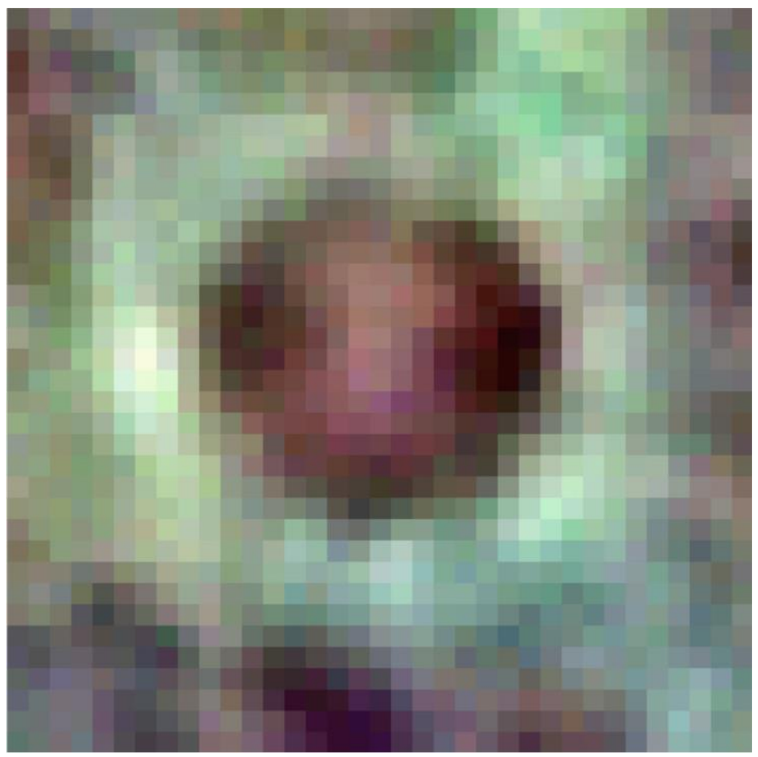


1000 iterations

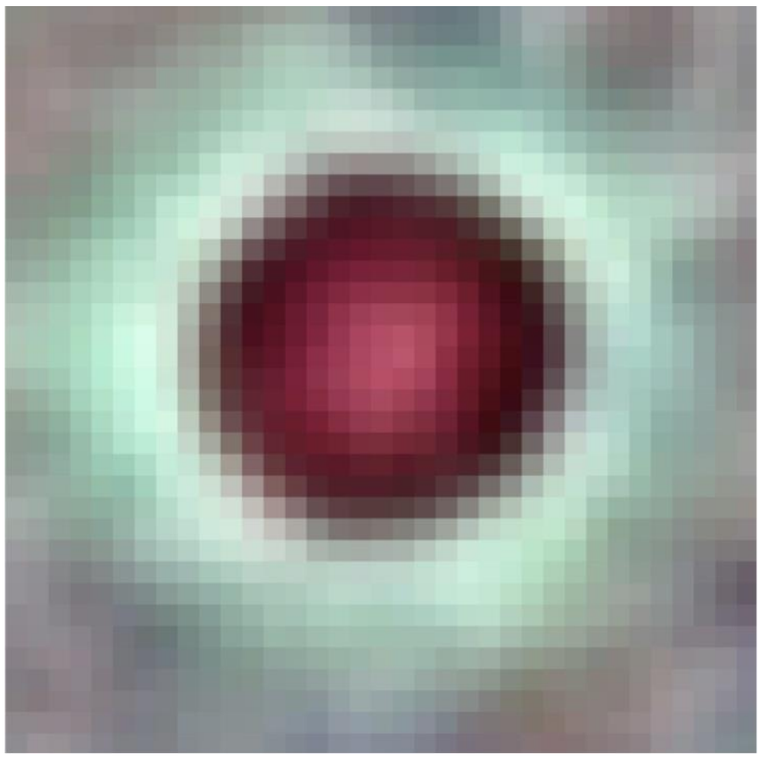
Linear Cell Classifier



Initialization

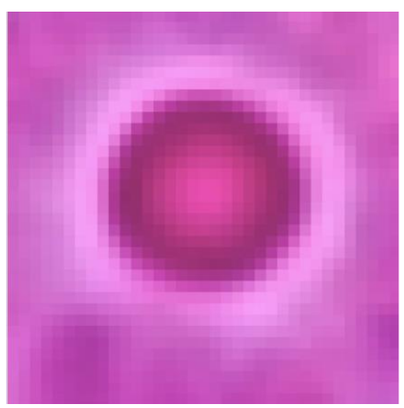


50 iterations

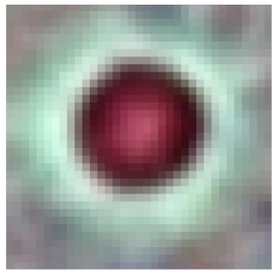


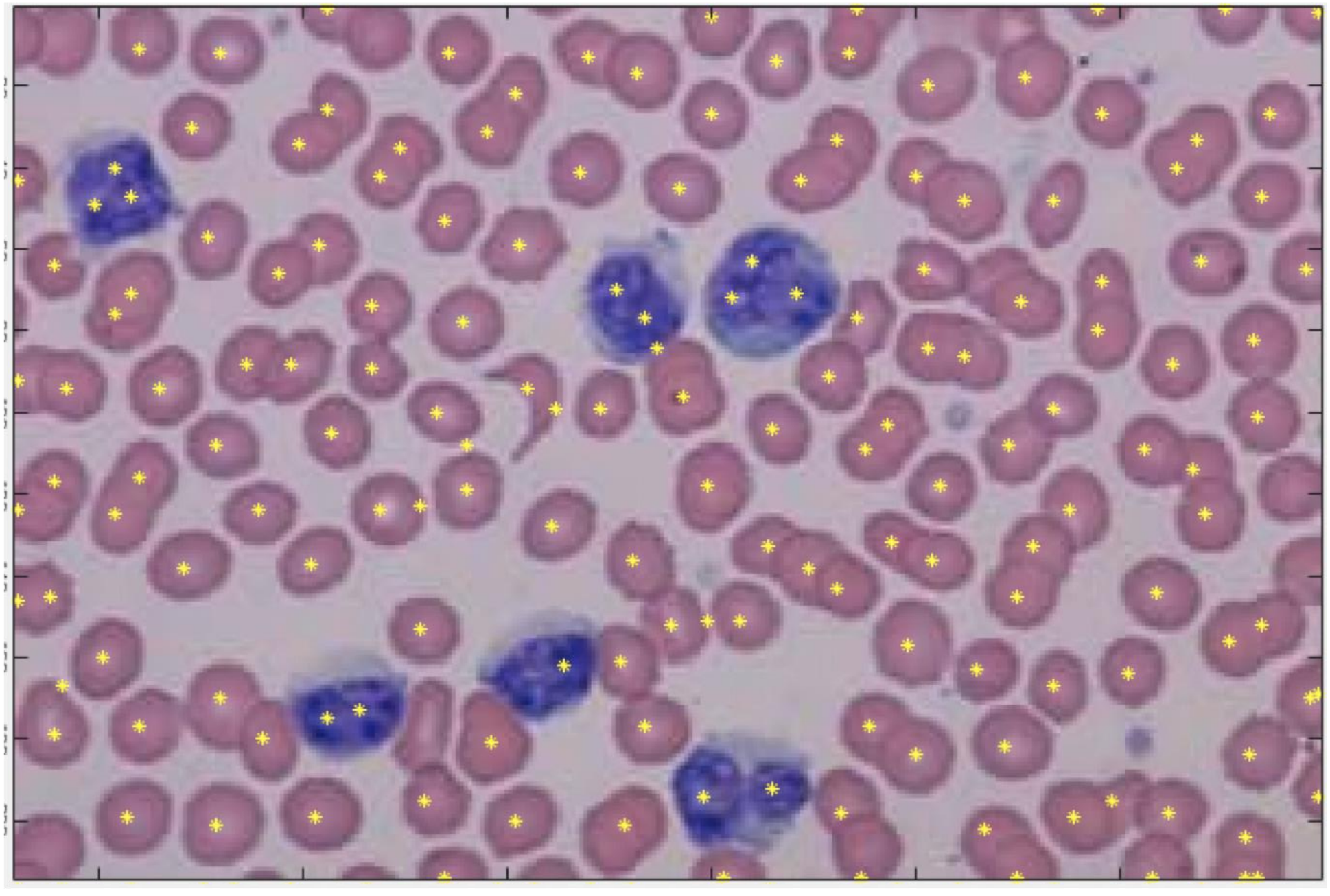
1000 iterations

Mean cell
(normalized)

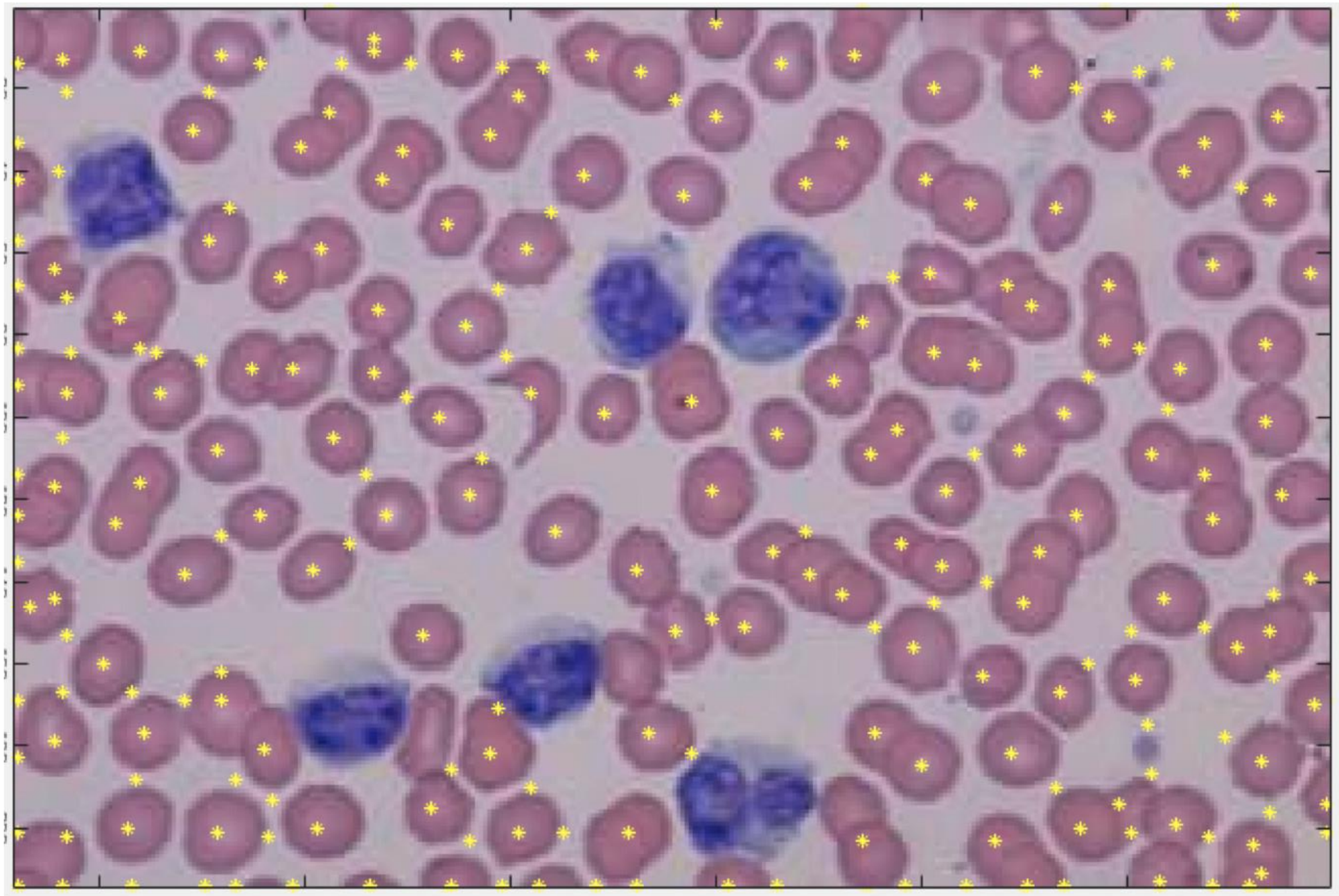


Importance of Sampling





Importance of Sampling

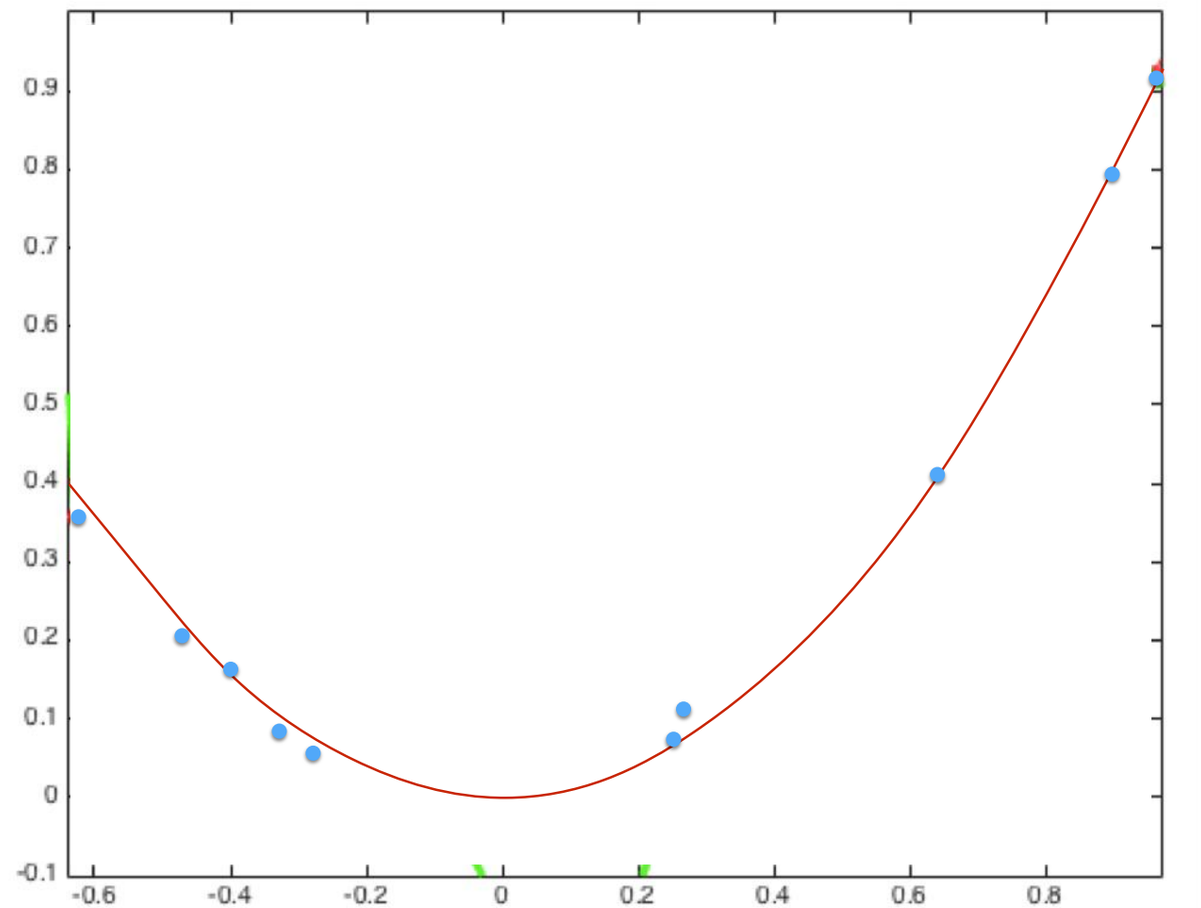


Overfitting

Training a model with many parameters on little data.

Overfitting

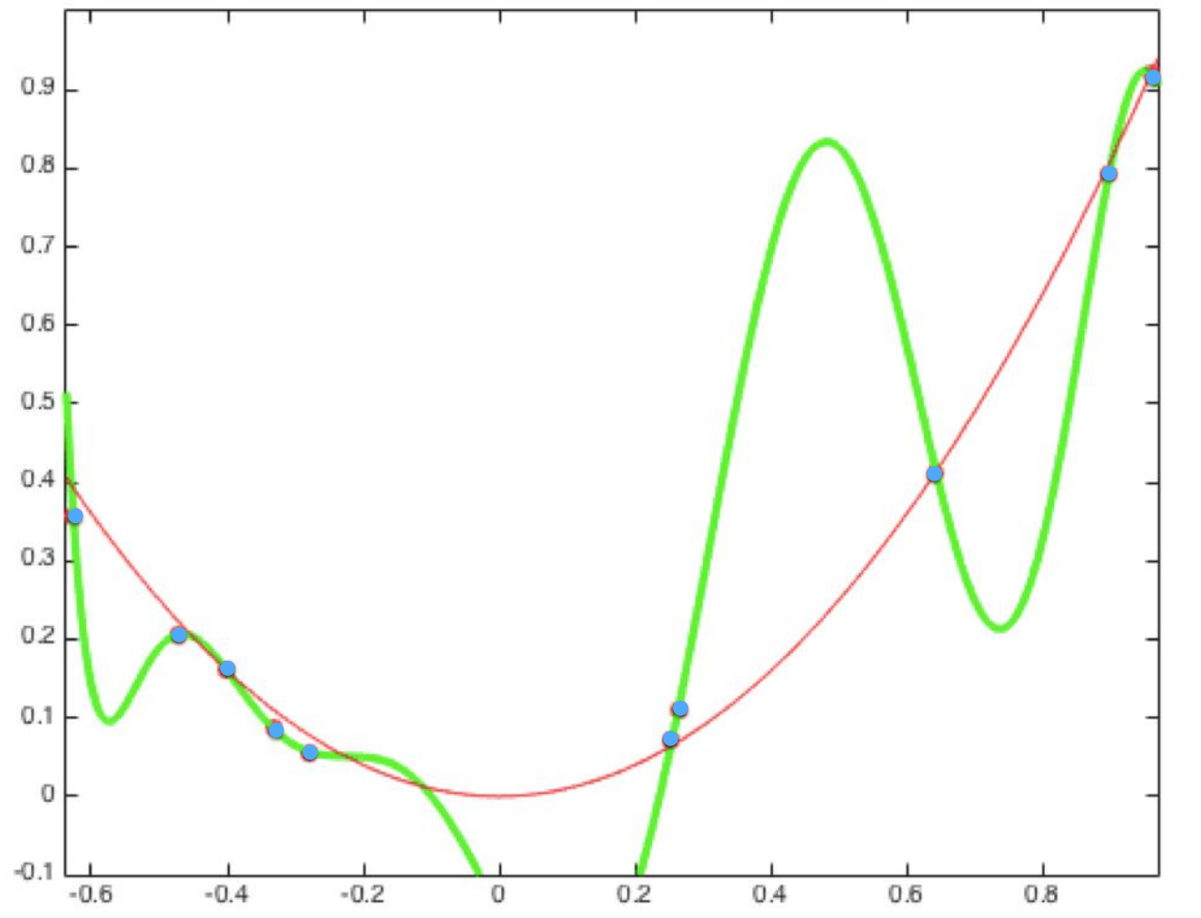
10 data points



$$y = x^2$$

Overfitting

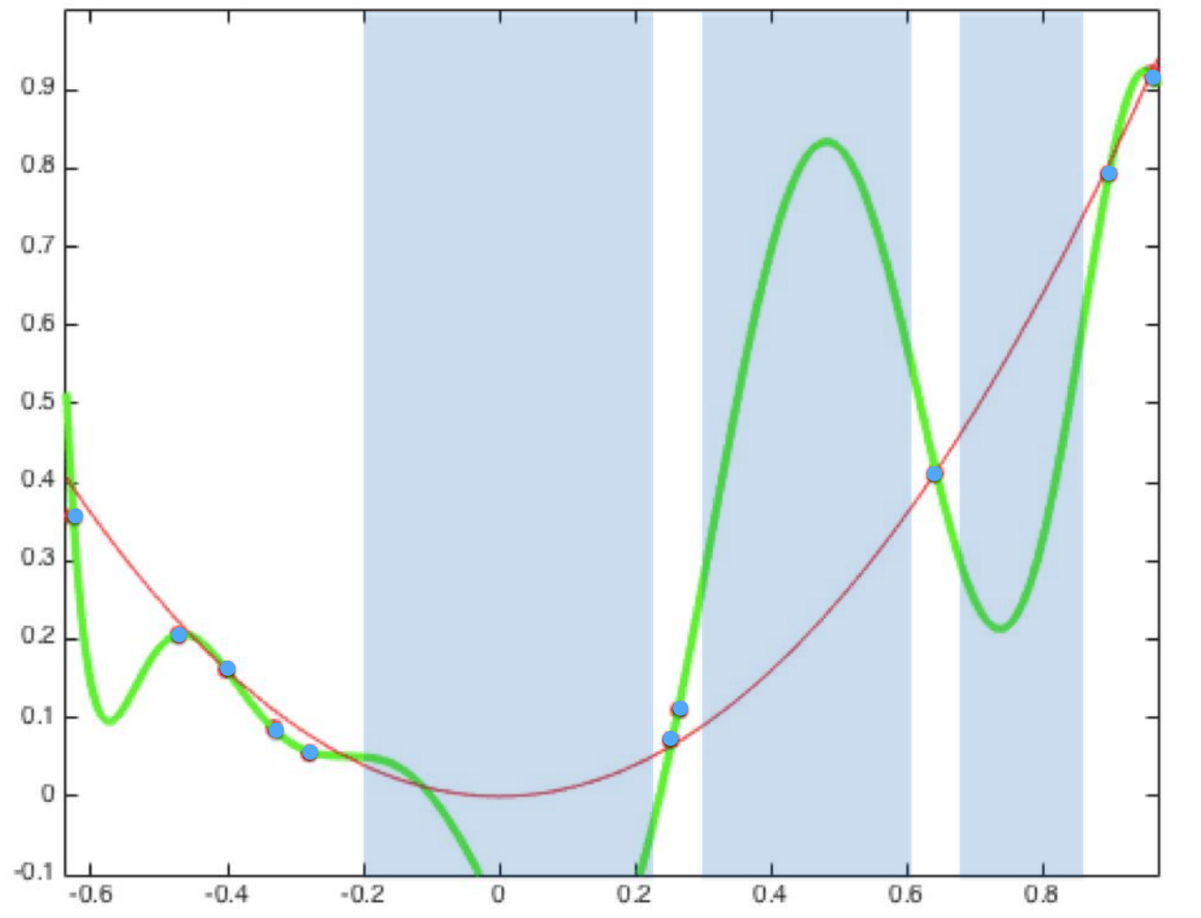
10 data points - Fitting 10^{th} degree polynomial



$$y = x^2$$

Overfitting

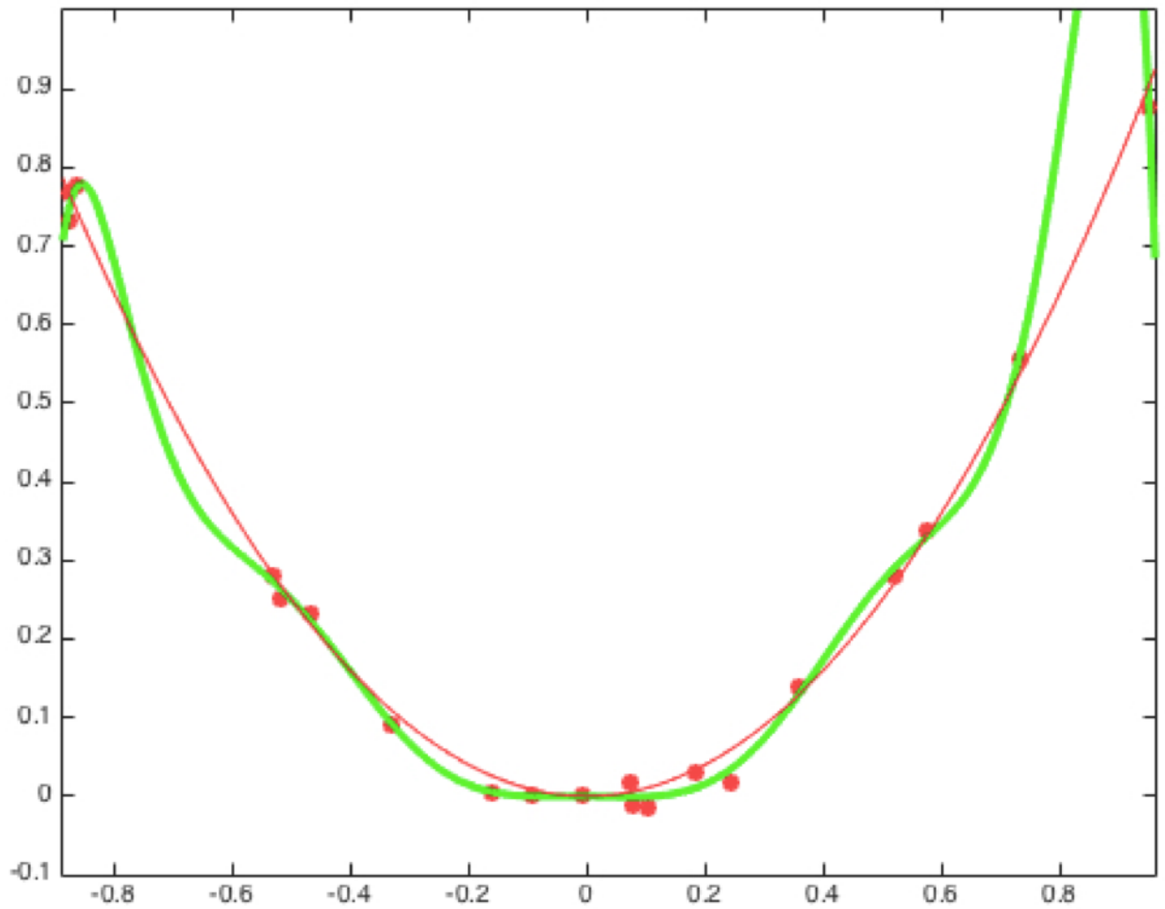
10 data points - Fitting 10^{th} degree polynomial



$$y = x^2$$

Overfitting

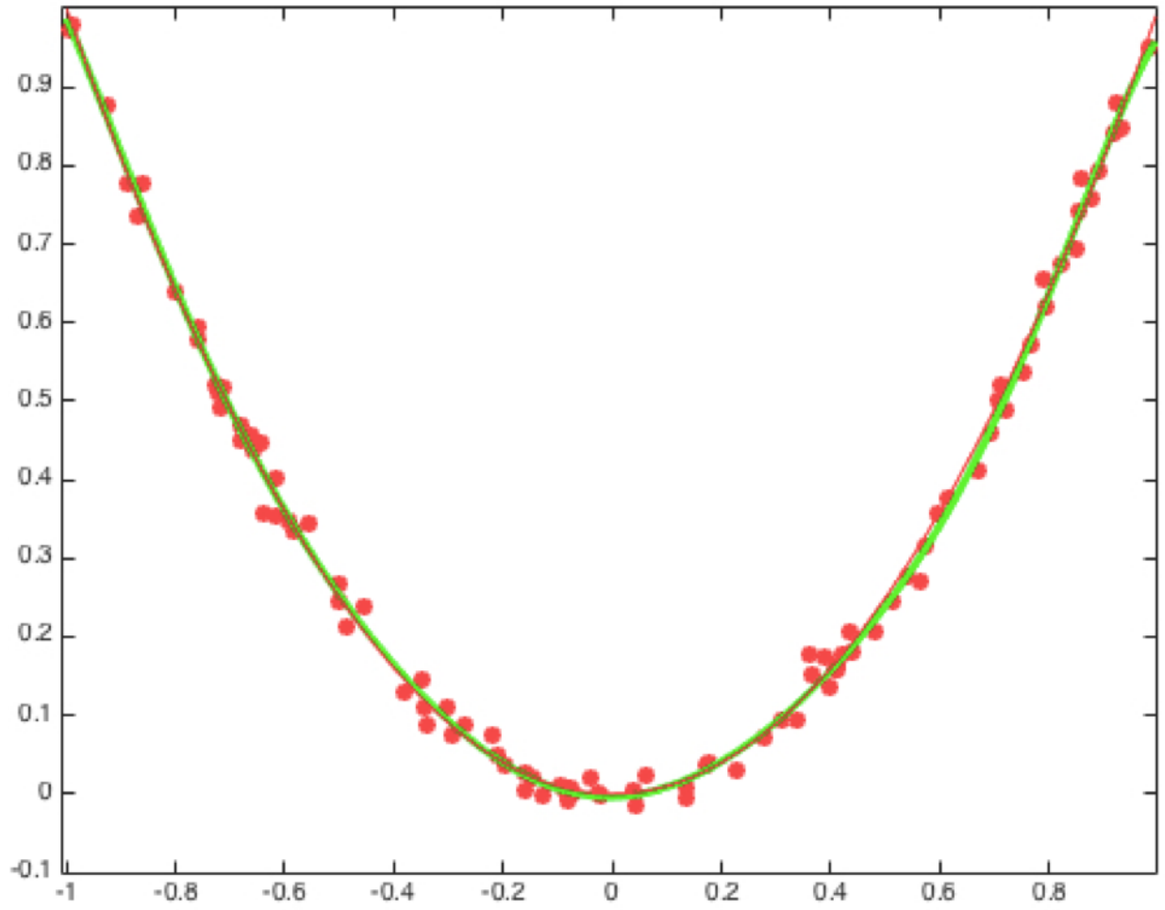
20 data points - Fitting 10^{th} degree polynomial



$$y = x^2$$

Overfitting

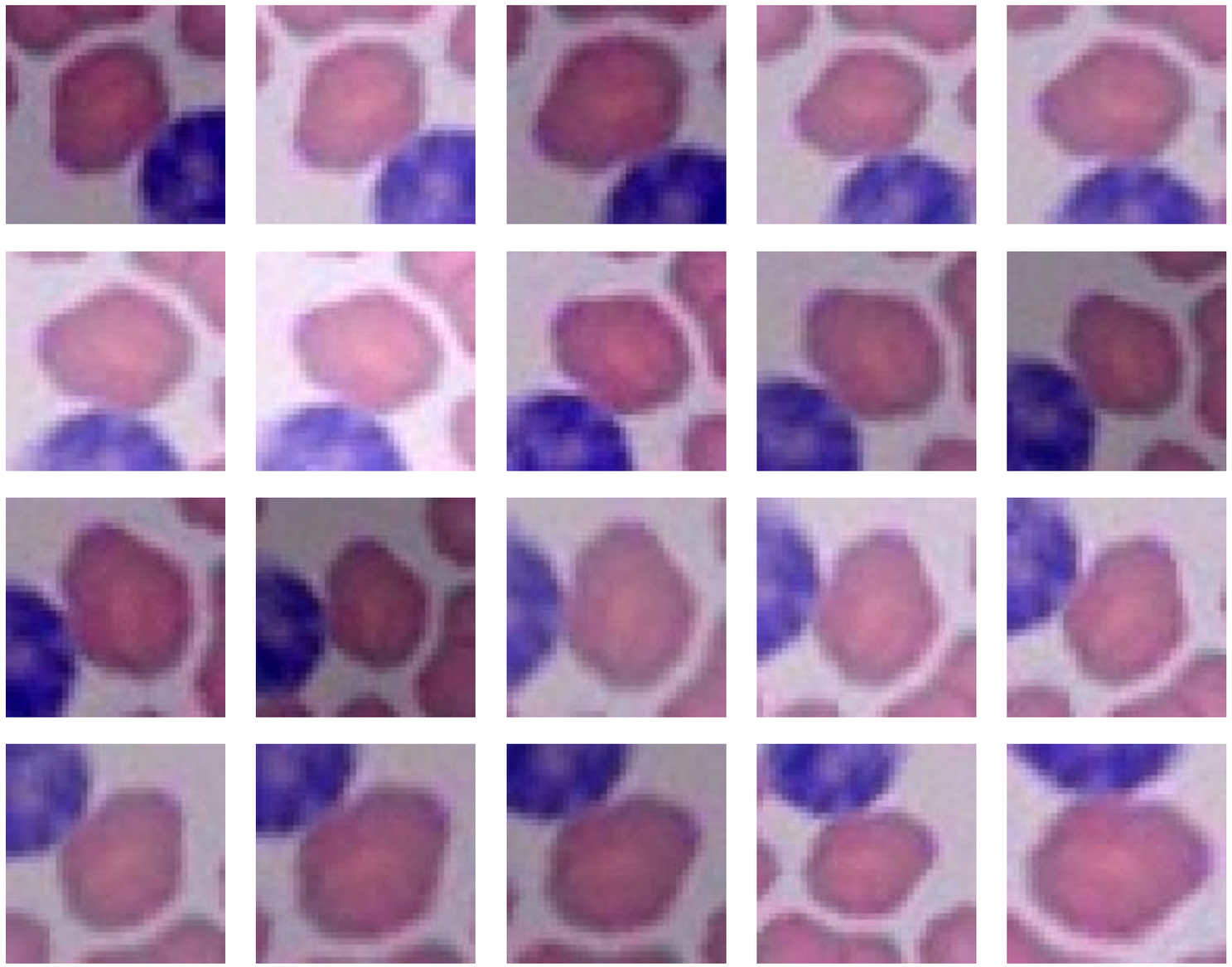
100 data points - Fitting 10^{th} degree polynomial



$$y = x^2$$

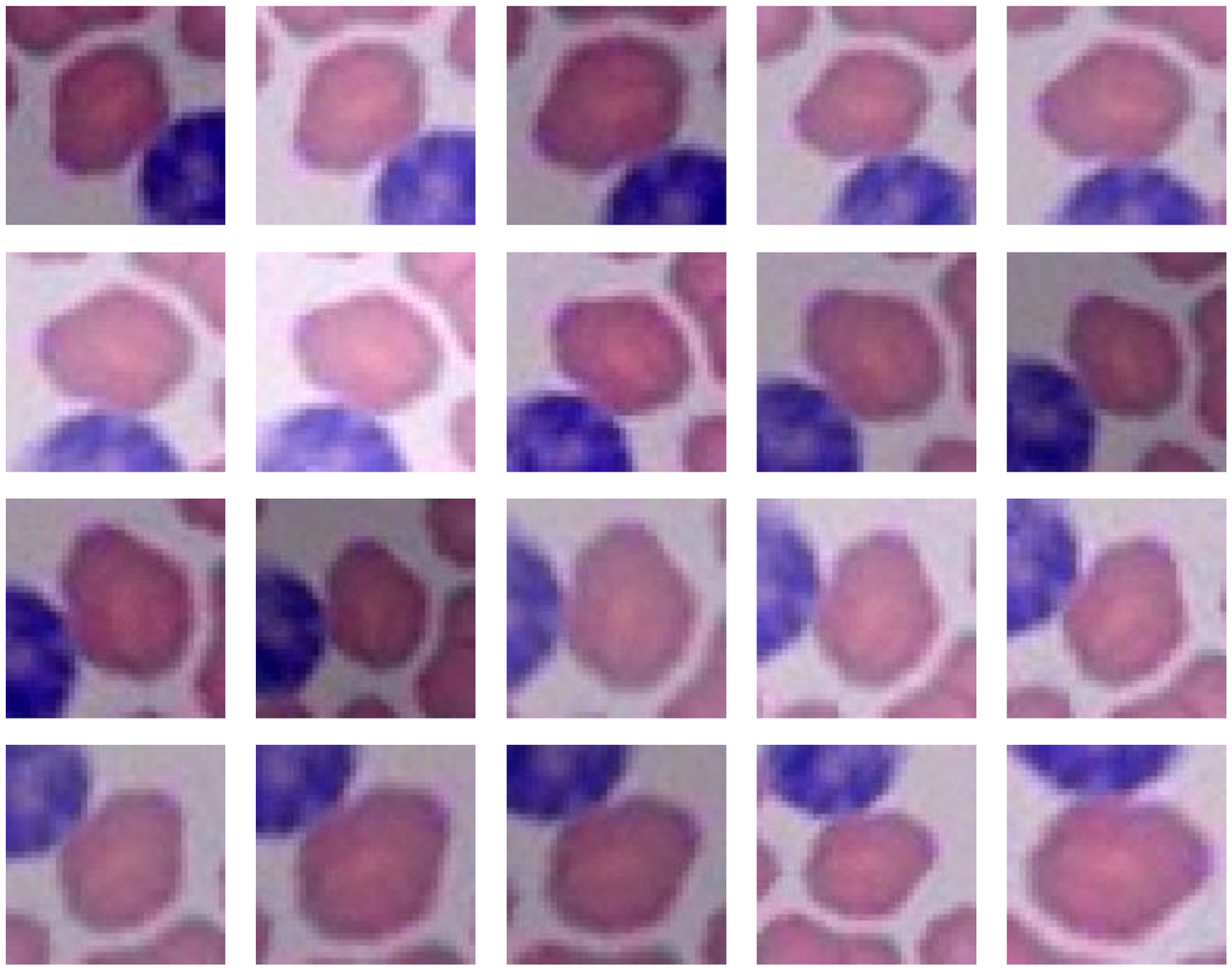
Data Augmentation

- Rotate
- Scale
- Change brightness
- Add noise



Data Augmentation

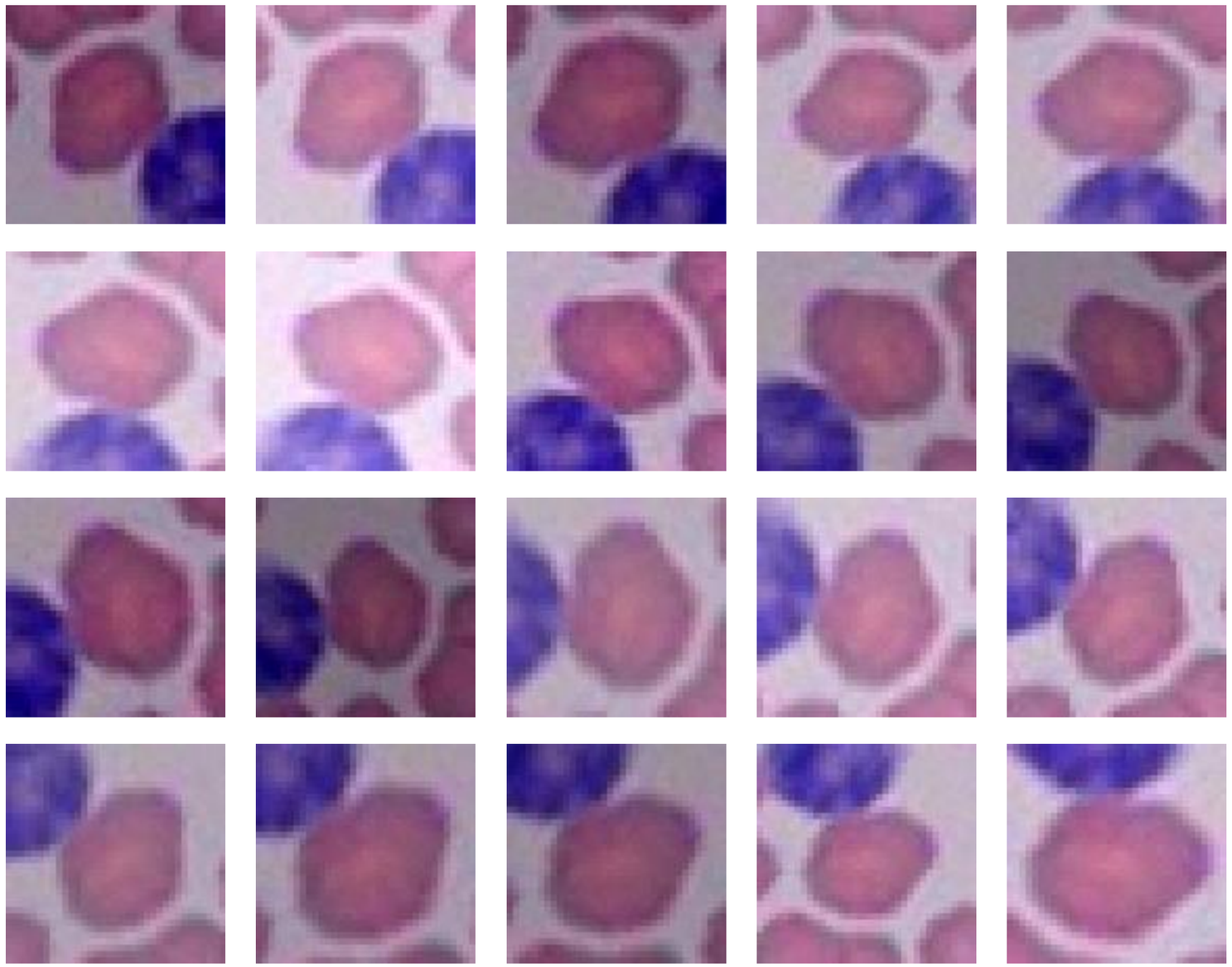
- Rotate
- Scale
- Change brightness
- Add noise



Can add a (small) amount of diversity to small datasets.

Data Augmentation

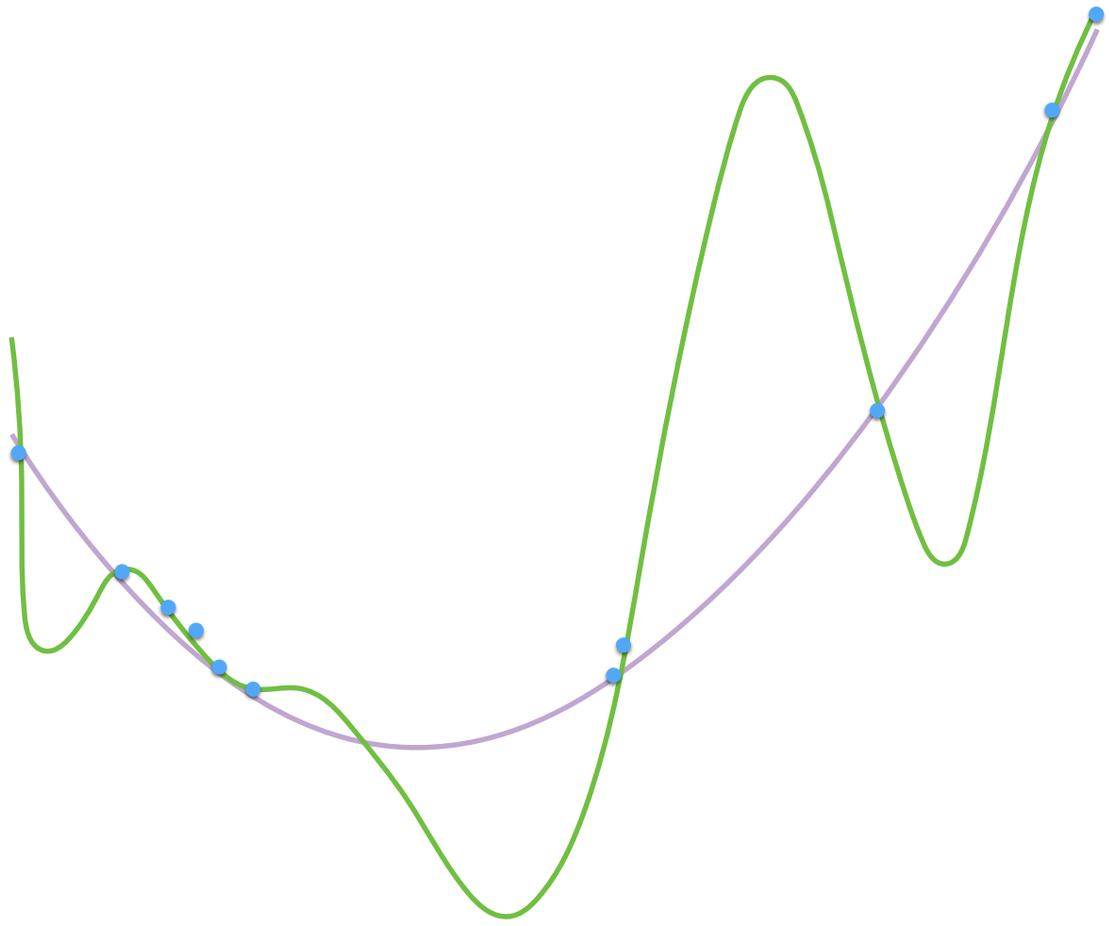
- Rotate
- Scale
- Change brightness
- Add noise



Average: (normalized)

Validation Data

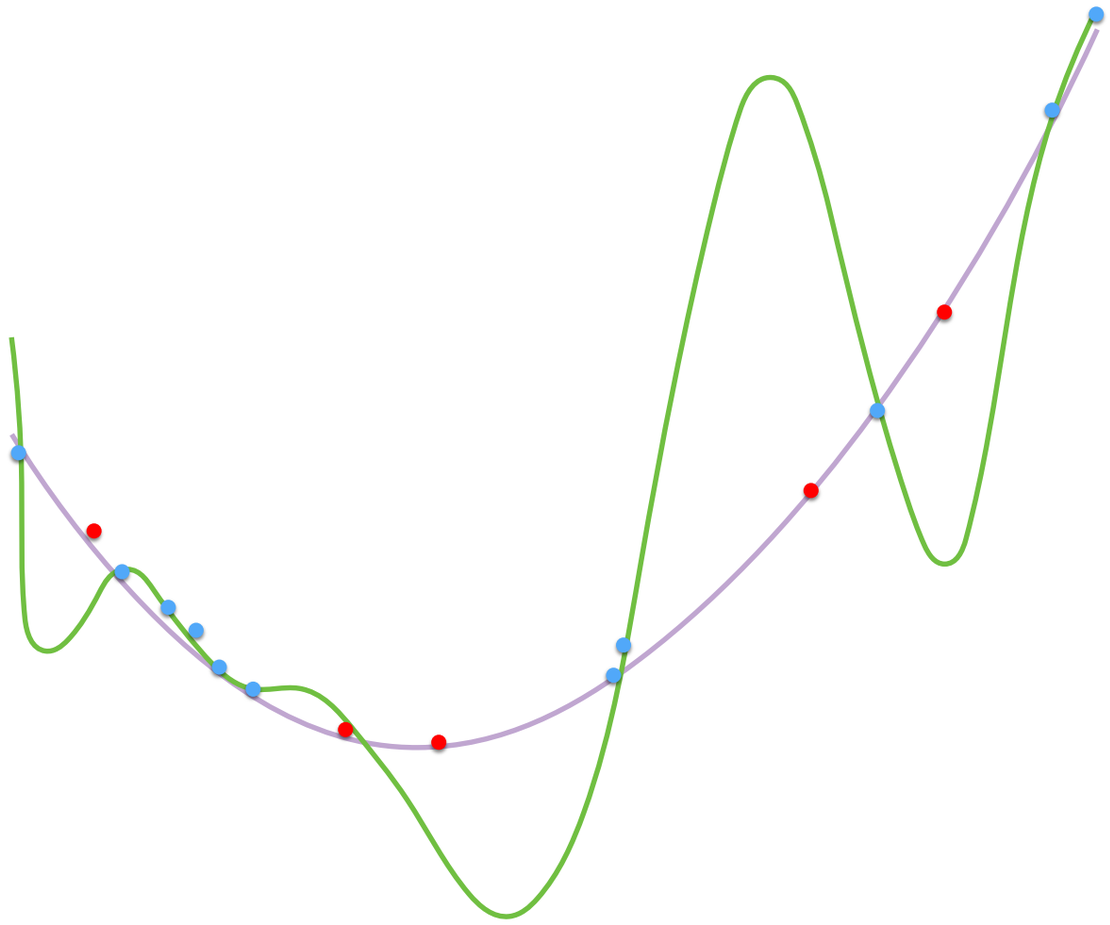
- Fitting 10^{th} degree polynomial



$$y = x^2$$

Validation Data

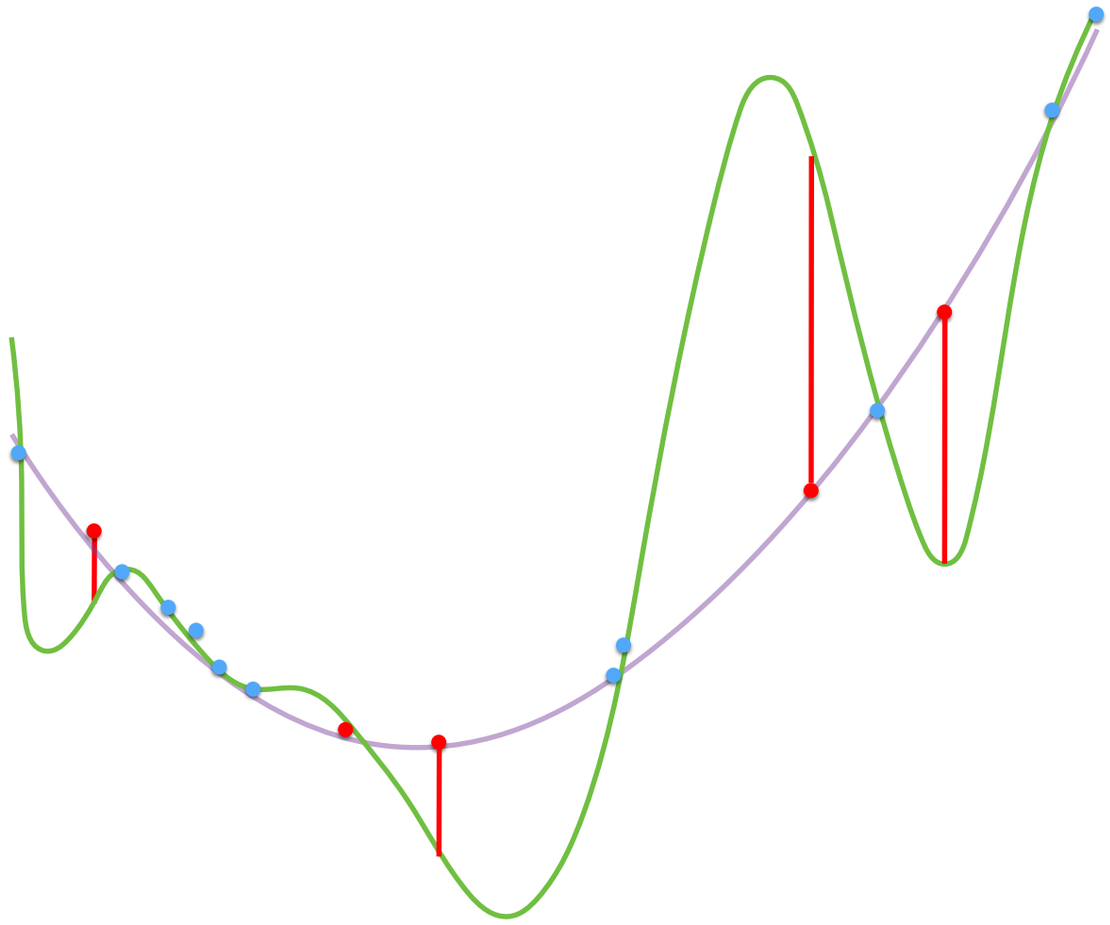
- Fitting 10^{th} degree polynomial



$$y = x^2$$

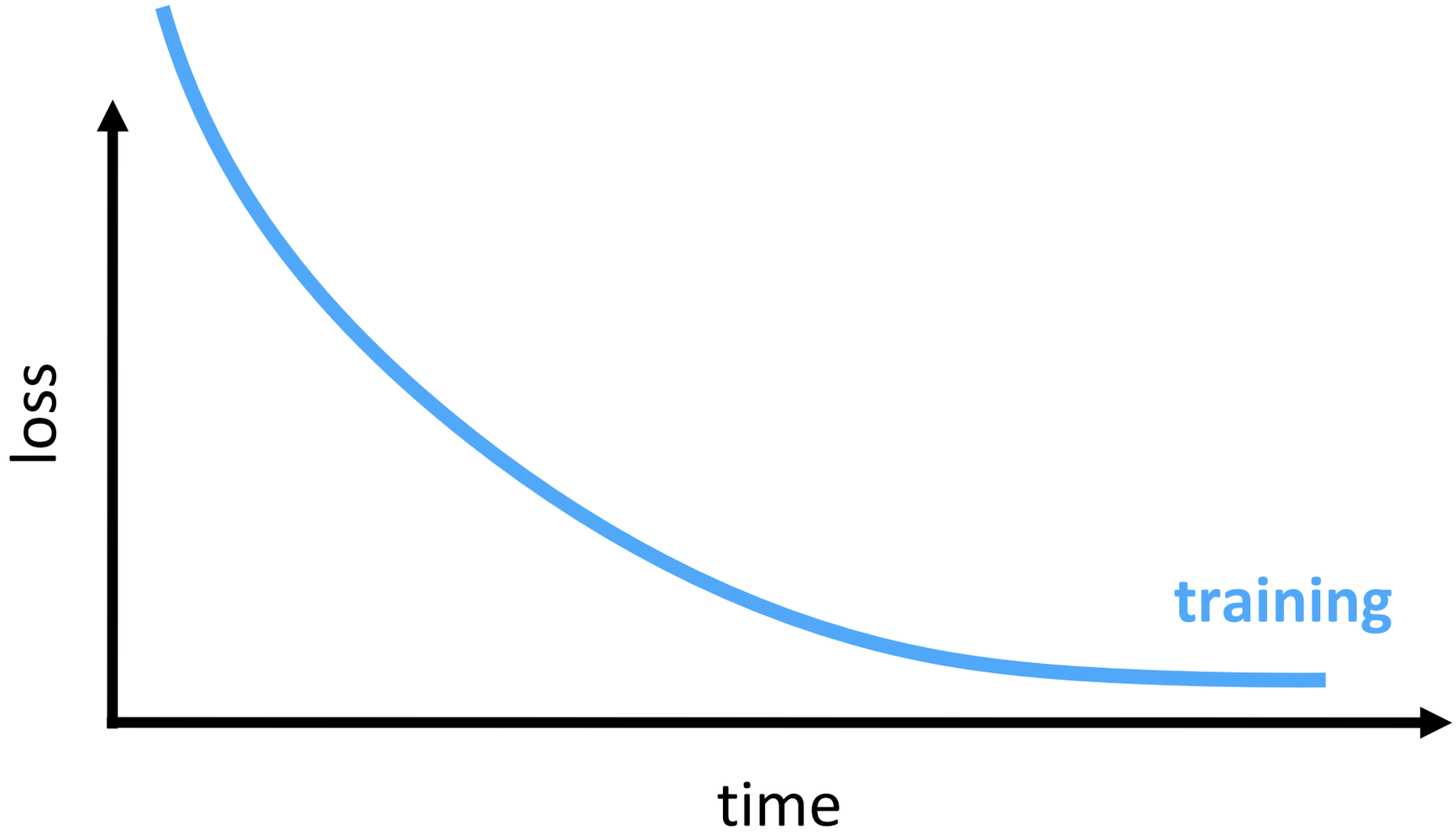
Validation Data

- Fitting 10^{th} degree polynomial

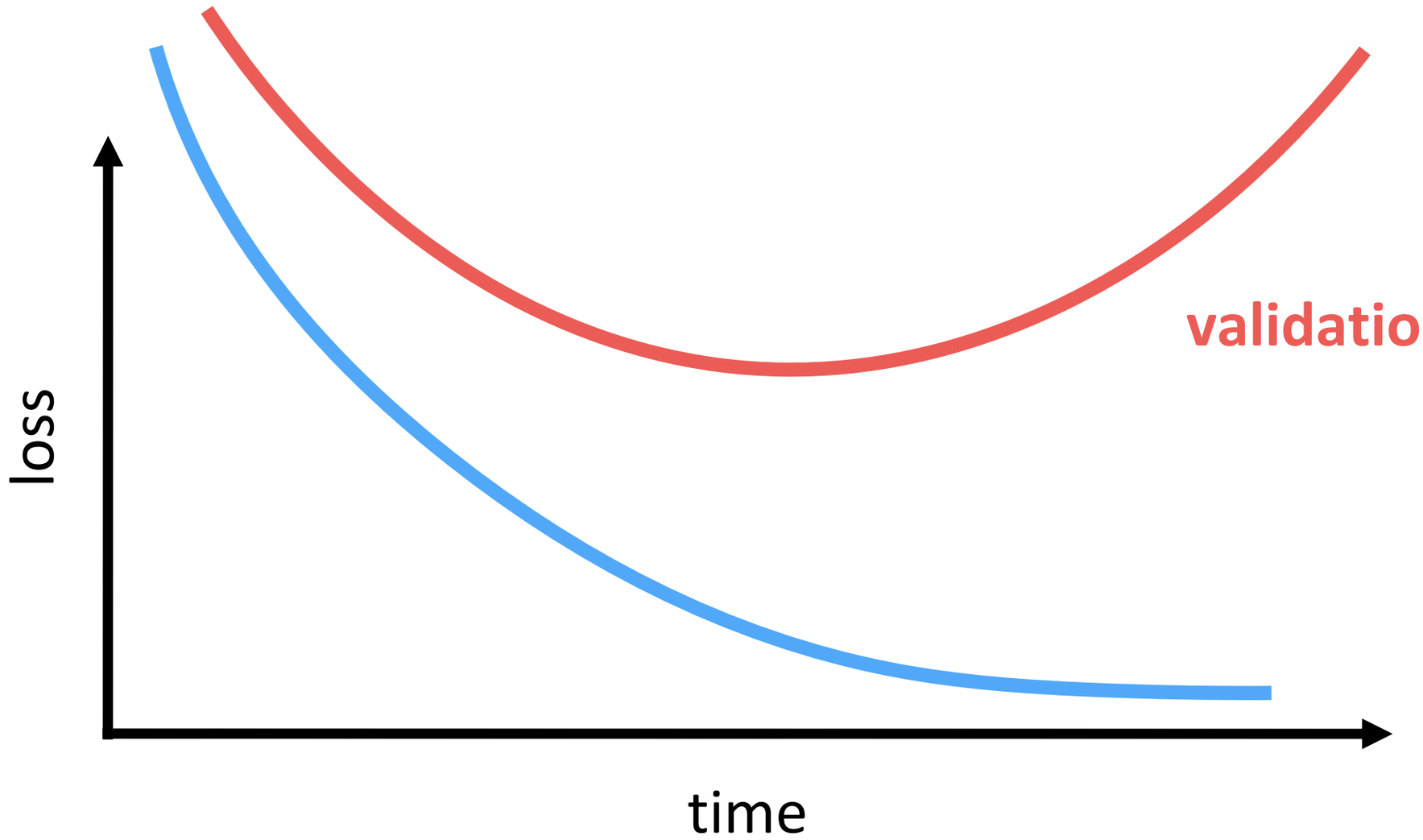


$$y = x^2$$

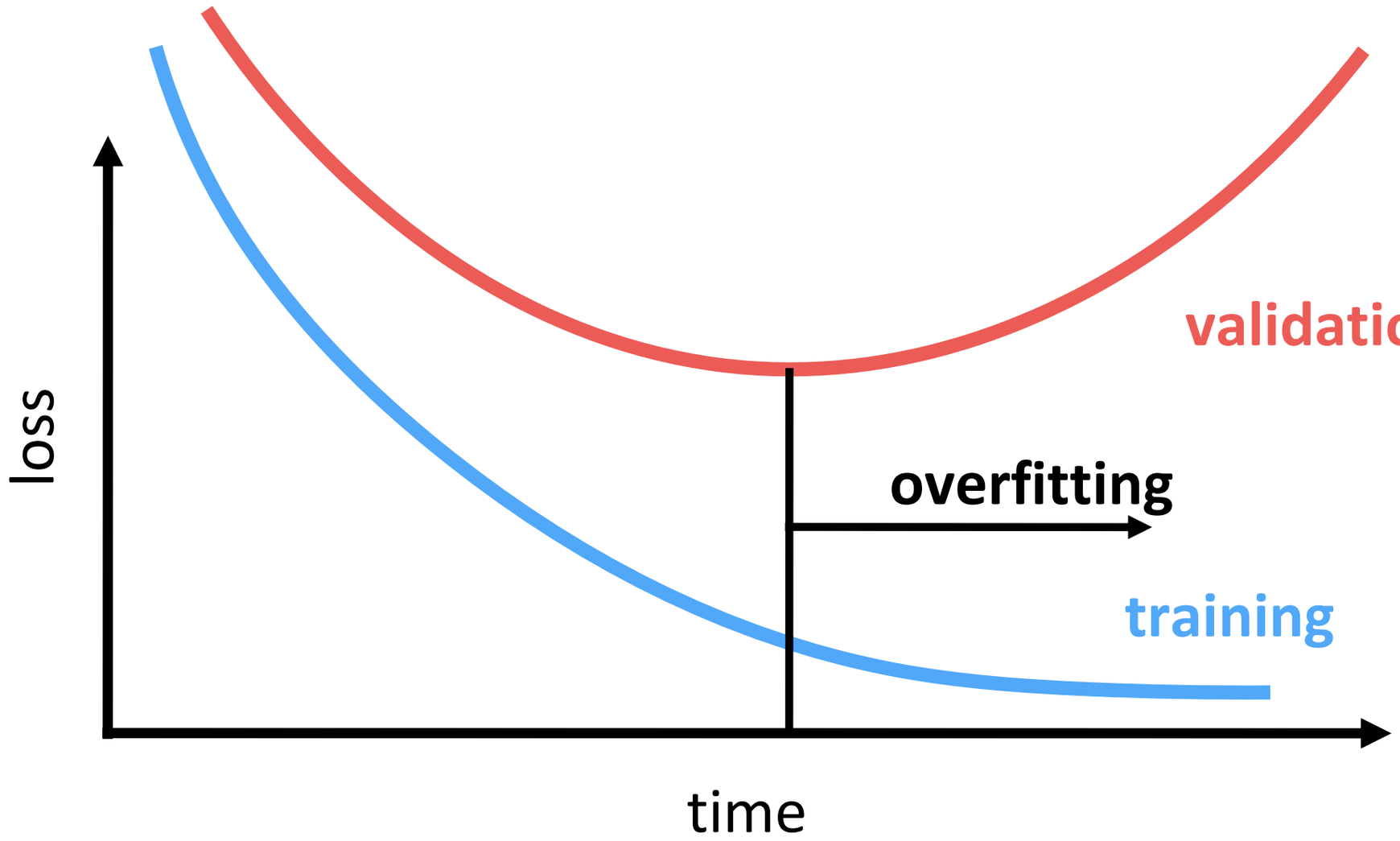
Detecting Overfitting



Detecting Overfitting

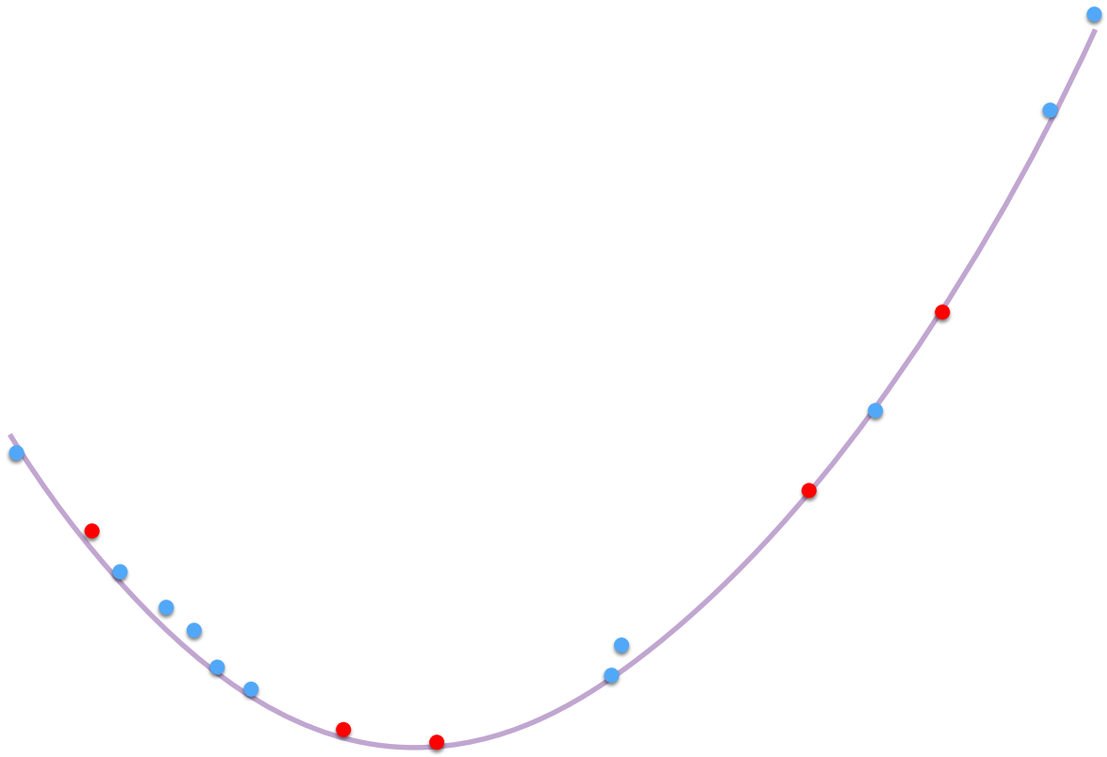


Detecting Overfitting



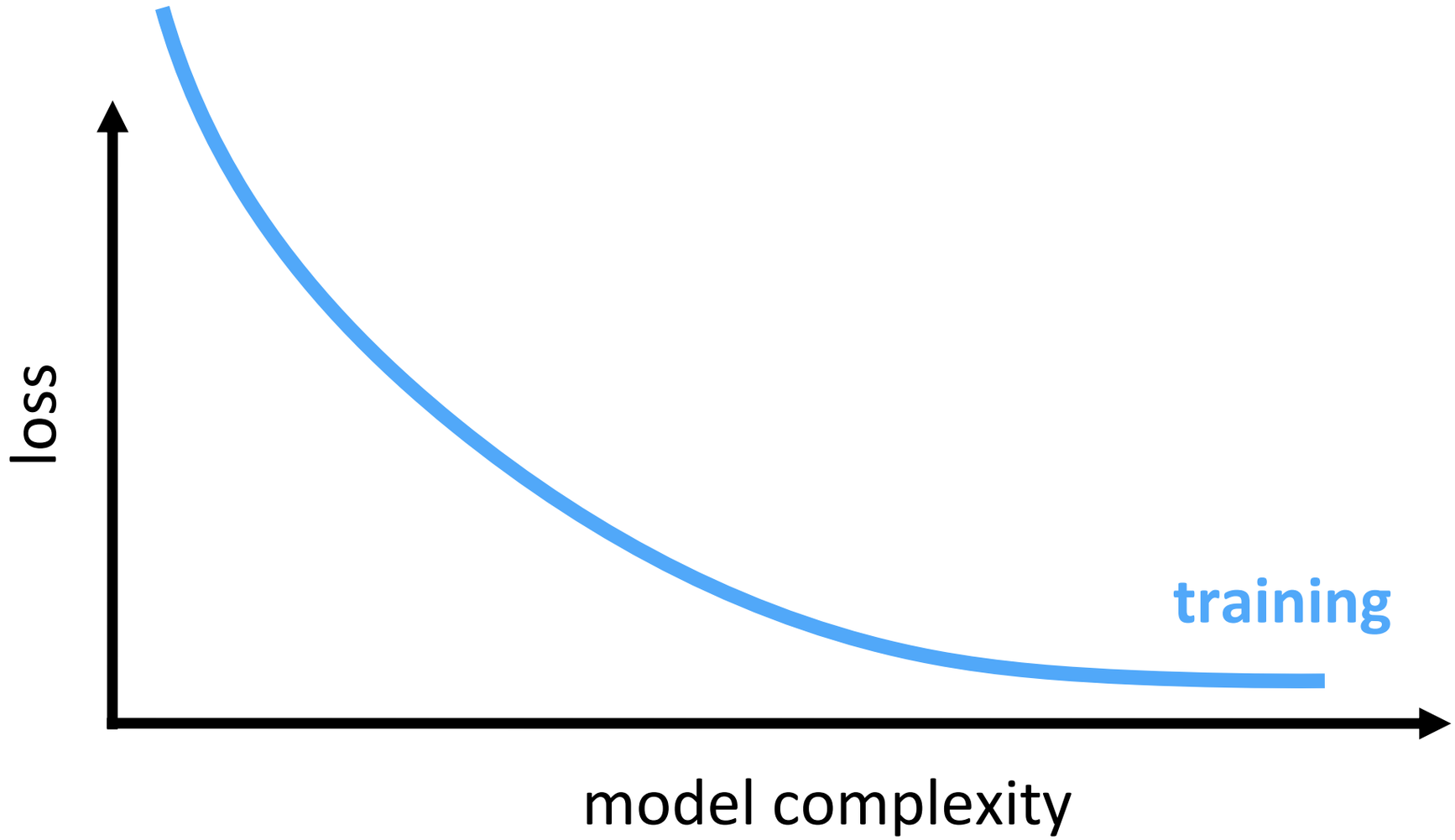
Validation Data

- Fitting 2nd degree polynomial

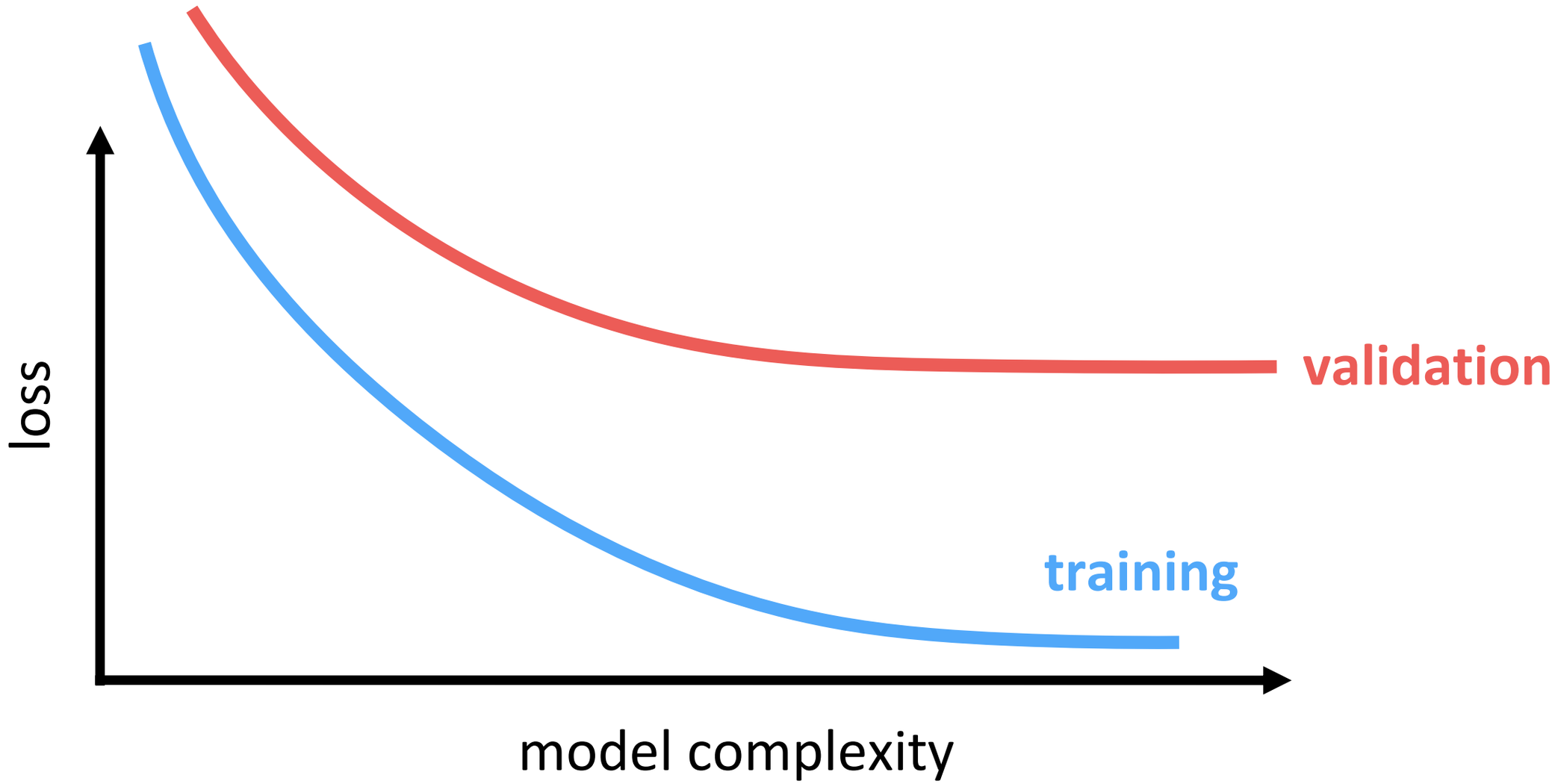


$$y = x^2$$

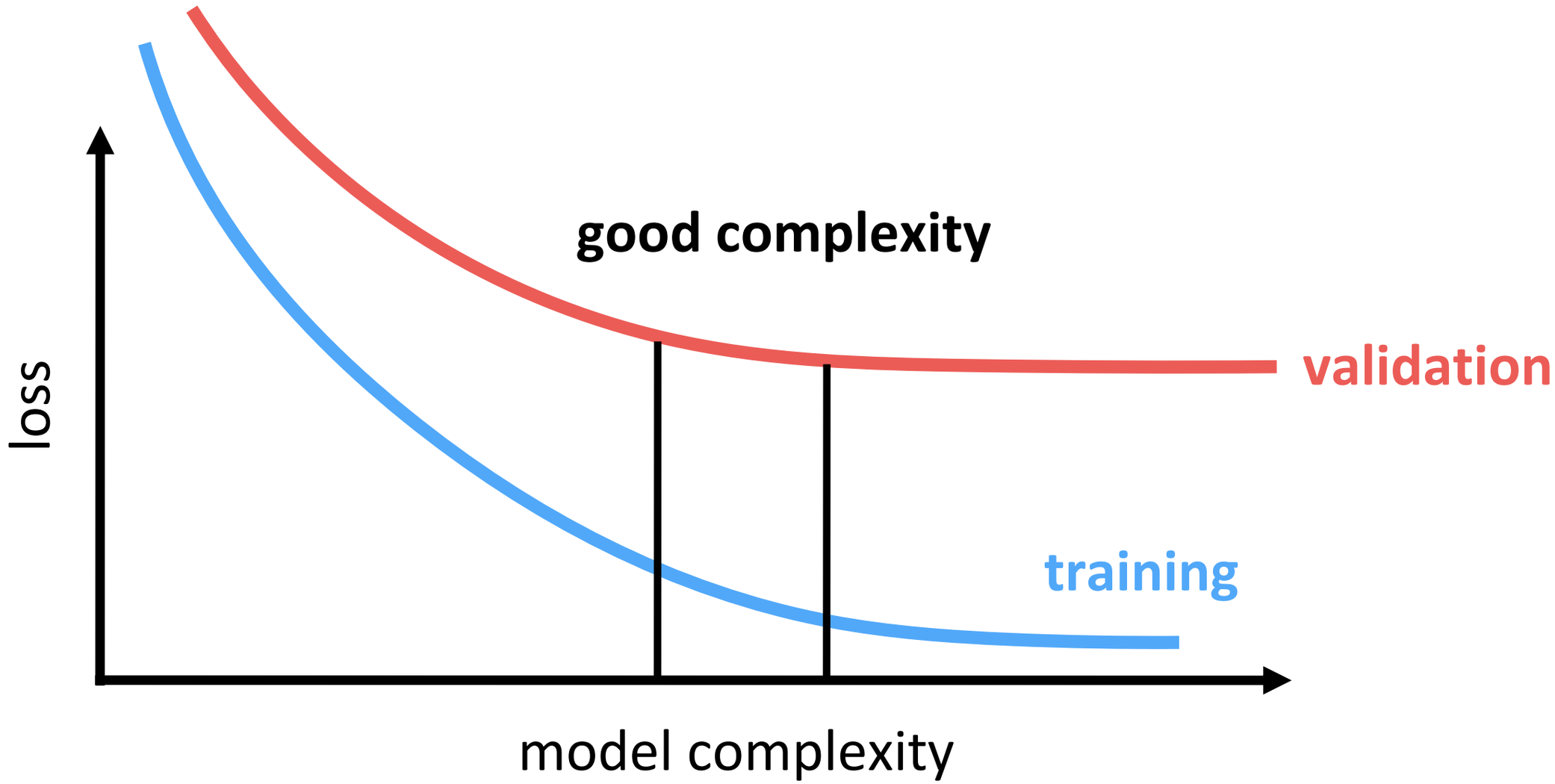
Determining Model Complexity



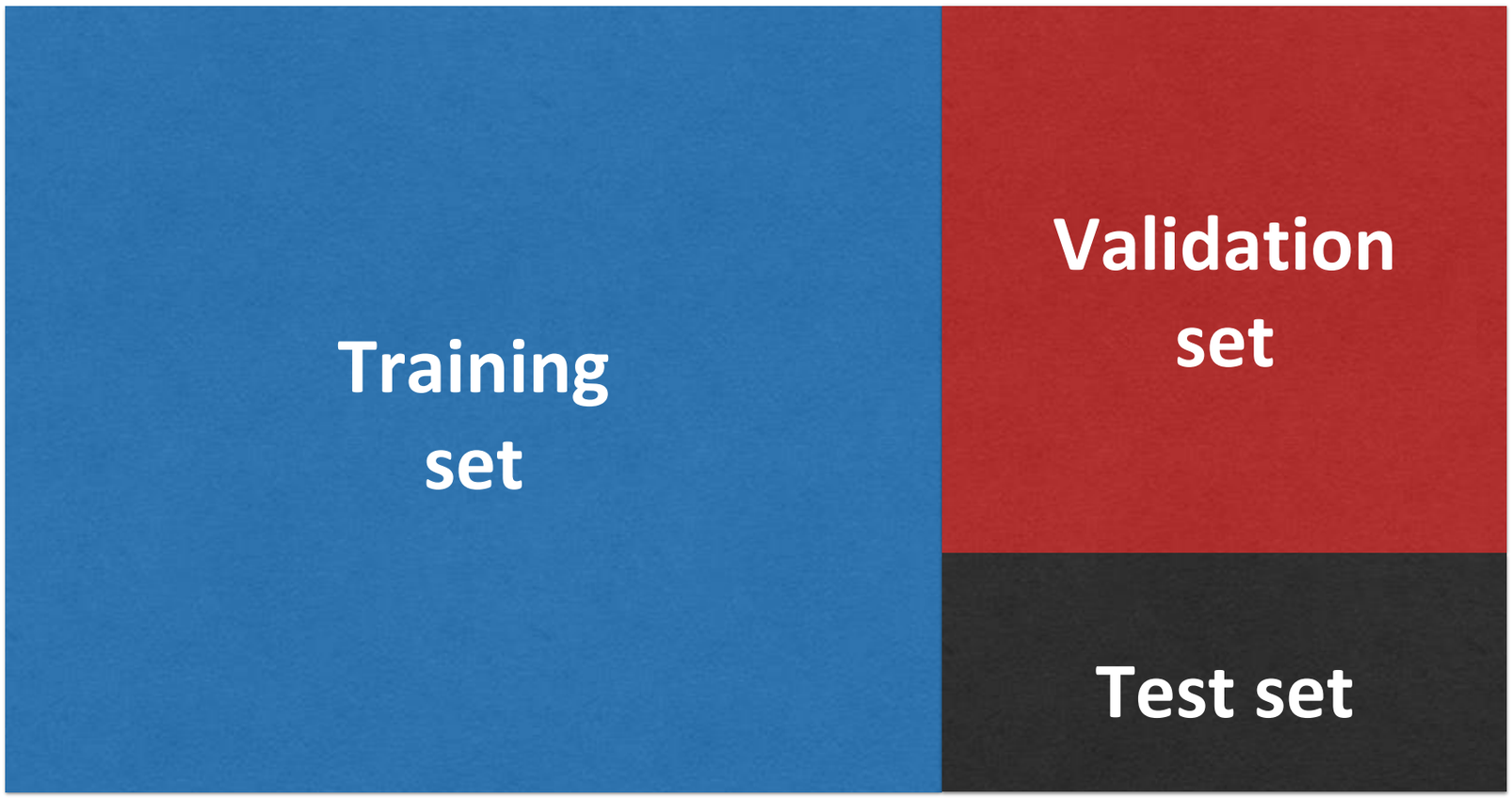
Determining Model Complexity



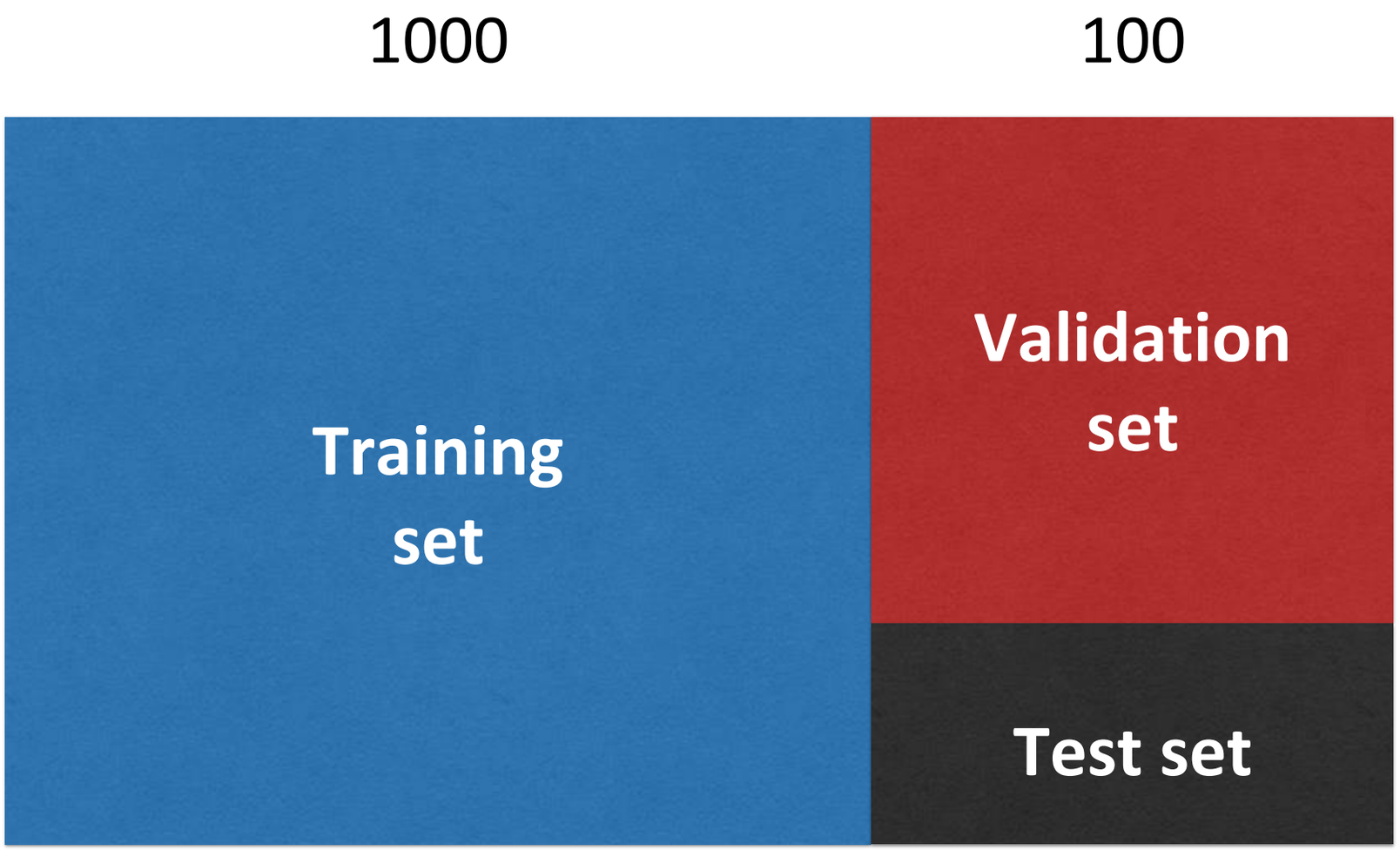
Determining Model Complexity



Training, Validation and Test

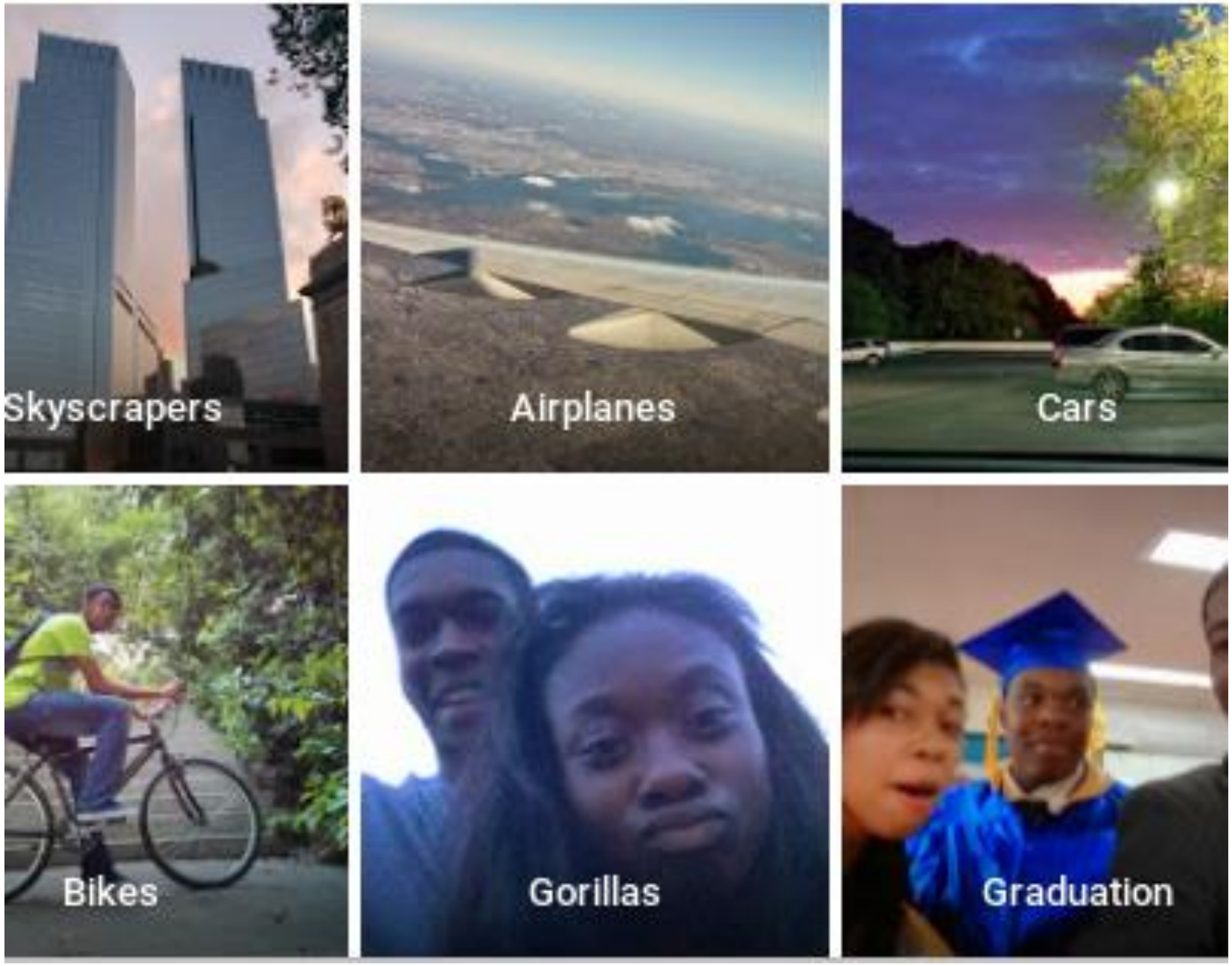


Training, Validation and Test



1

Dataset Bias



@jackyalcine 2015

Dataset Bias

REUTERS

Business Markets World Politics TV More

TECHNOLOGY NEWS OCTOBER 10, 2018 / 3:12 AM / A YEAR AGO

Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin 8 MIN READ

SAN FRANCISCO (Reuters) - Amazon.com Inc's (AMZN.O) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.

The team had been building computer programs since 2014 to review job applicants' resumes with the aim of mechanizing the search for top talent, five people familiar with the effort told Reuters.

source: Reuters

Dataset Bias

That is because Amazon's computer models were trained to vet applicants by observing patterns in resumes submitted to the company over a 10-year period. Most came from men, a reflection of male dominance across the tech industry.

In effect, Amazon's system taught itself that male candidates were preferable. It penalized resumes that included the word "women's," as in "women's chess club captain." And it downgraded graduates of two all-women's colleges, according to people familiar with the matter. They did not specify the names of the schools.

source: Reuters

Lessons Learned

- Main lessons from this lecture
  - Learning a classifier via statistical learning
  - Minimizing negative log-likelihood via stochastic gradient descent
  - Importance of validation set
- Next lecture: Convolutional Neural Networks

Last Lecture

Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	Filtering, gradients, scale	Lab 1
Jan. 27	Local features	
Jan. 30	Learning a classifier	Lab 2
Feb. 3	**Convolutional neural networks**	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
Feb. 17	Camera Geometry	Lab 4
Feb. 20	More camera geometry	
Feb. 24	Generative neural networks	
Feb. 27	Generative neural networks	
Mar. 2	TBA	
Mar. 9	TBA	