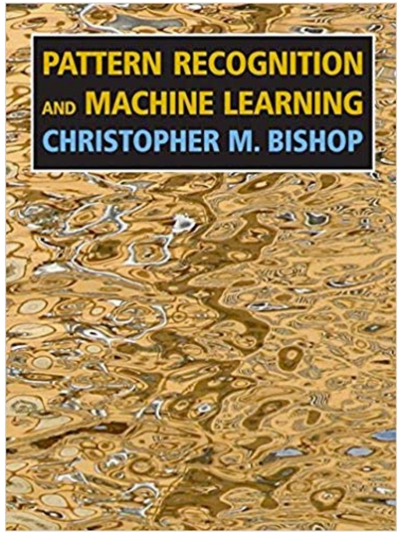
Course material at:
https://github.com/lyeskhalil/mlbootcamp2022

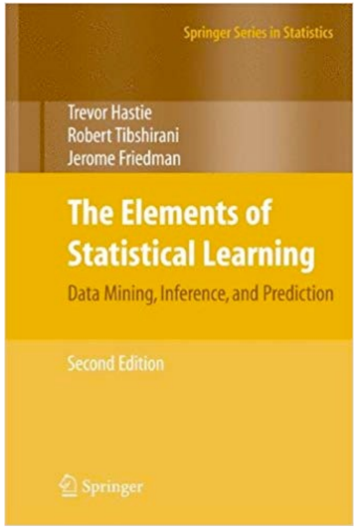


Regression, Overfitting and Optimization

Monday, Lecture 2

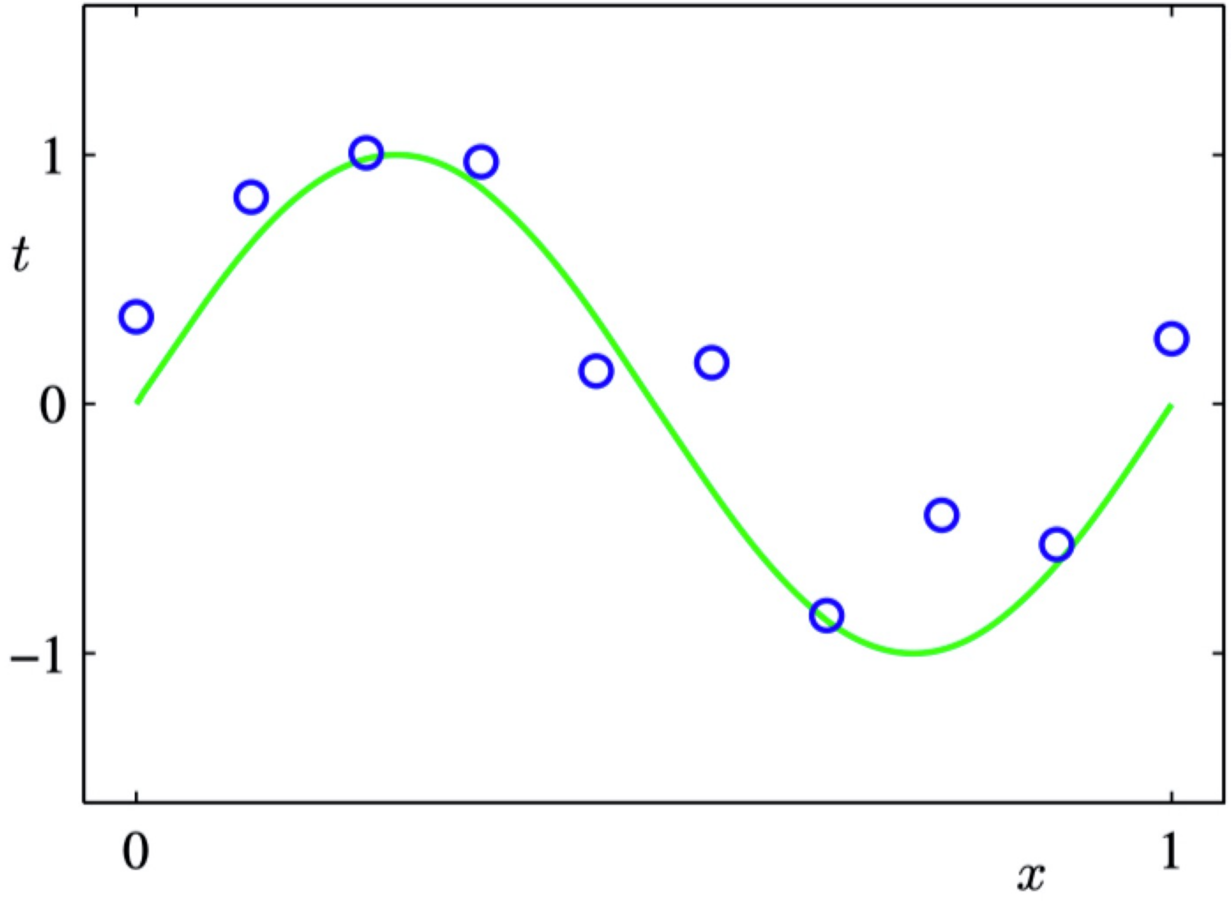
FASE ML Bootcamp

Based on material from Joseph Redmon, Christopher Bishop, Polo Chau
Elements of Statistical Learning by Hastie, Tibshirani, Friedman



Regression

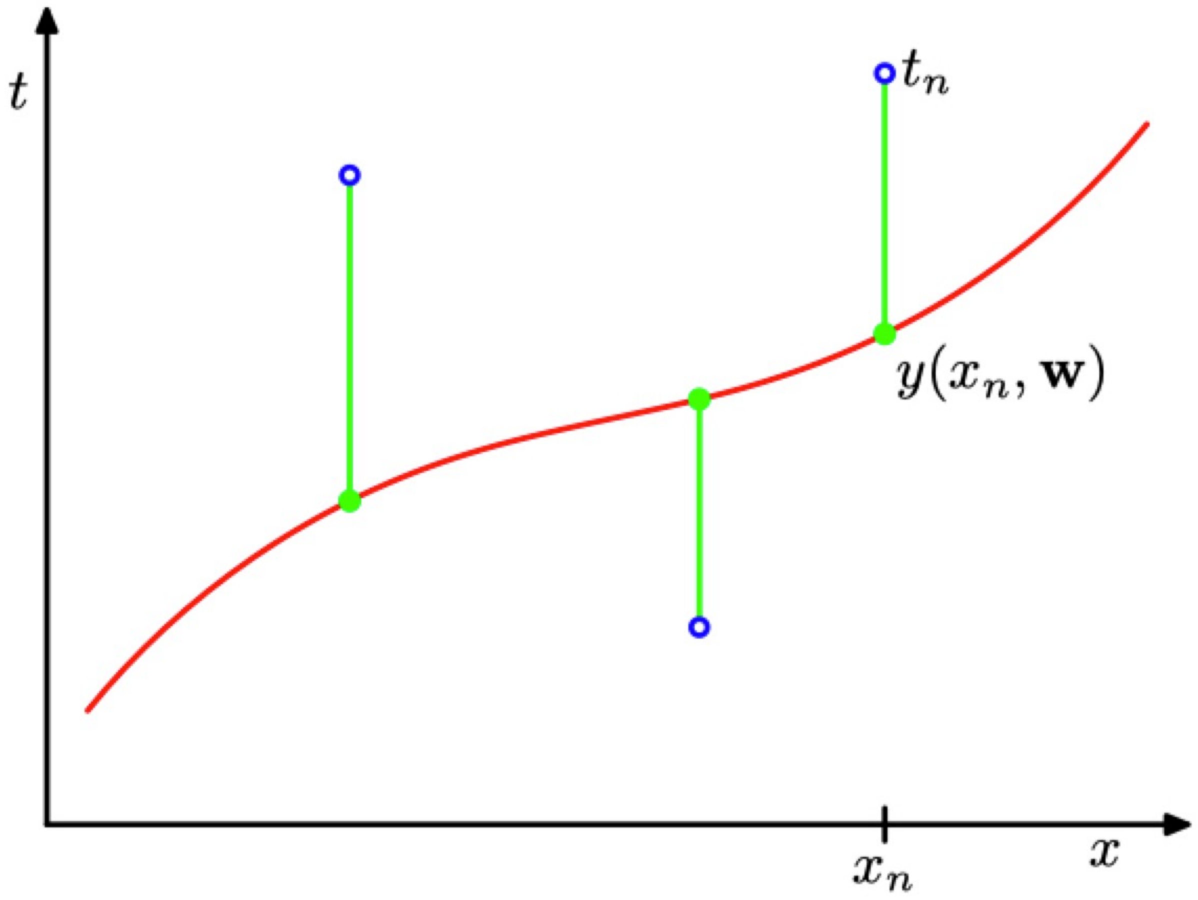
Data points **(x,t)** generated by adding noise to $\boldsymbol{sin(2\pi x)}$

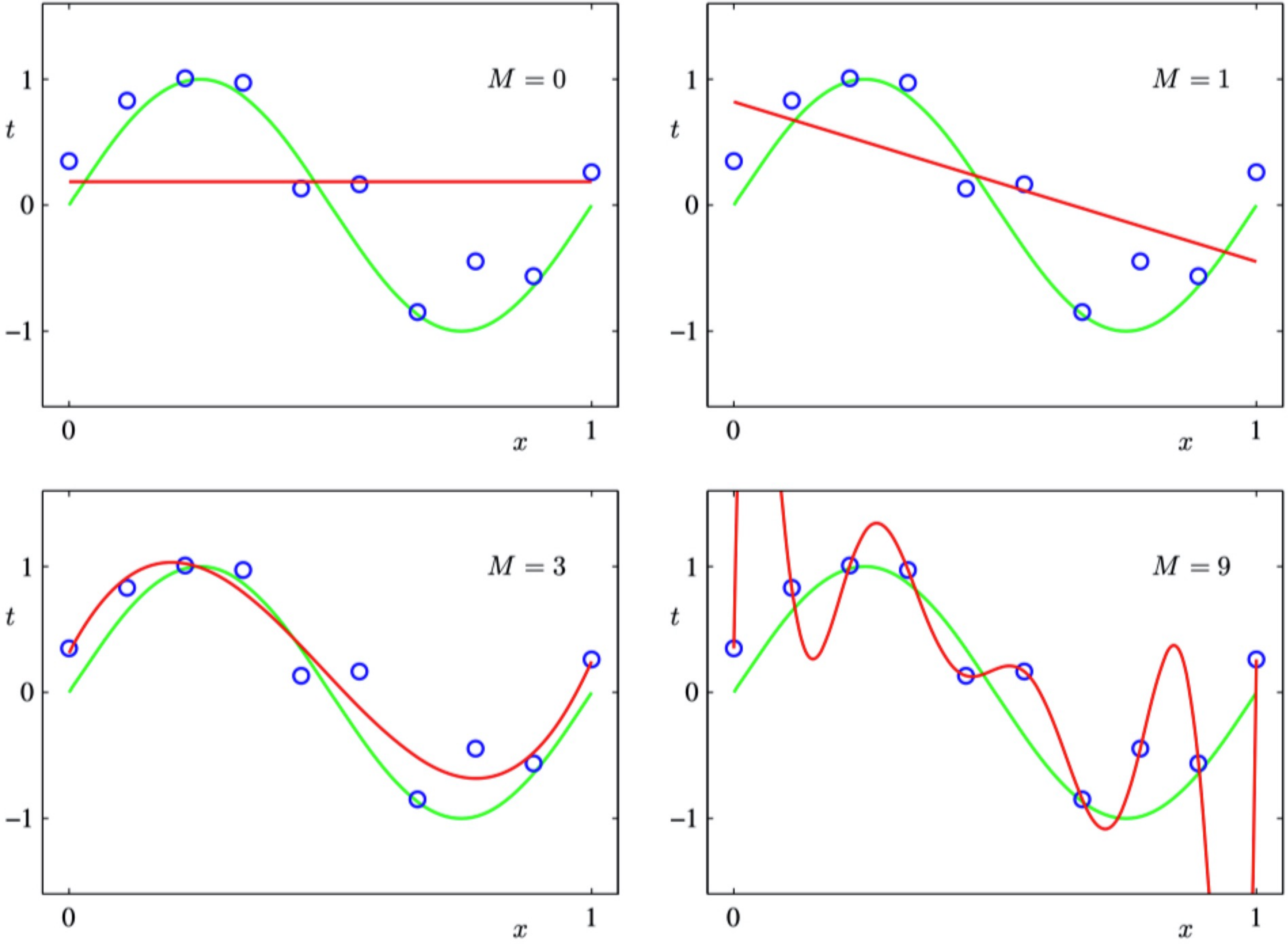


Loss Function!

Errors in Regression

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$

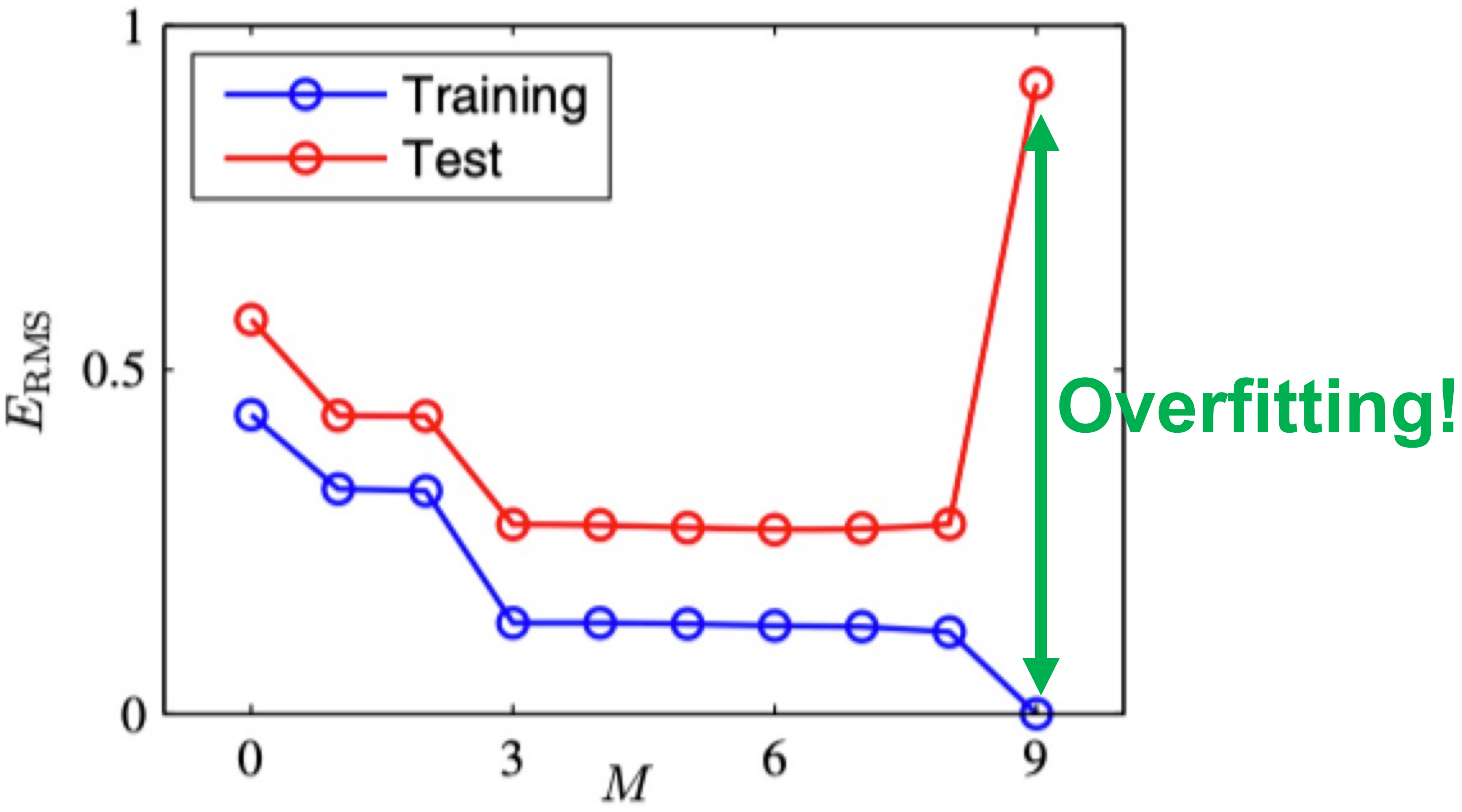




M = 0
M = 1
M = 3
M = 9
t
x
1
0
−1

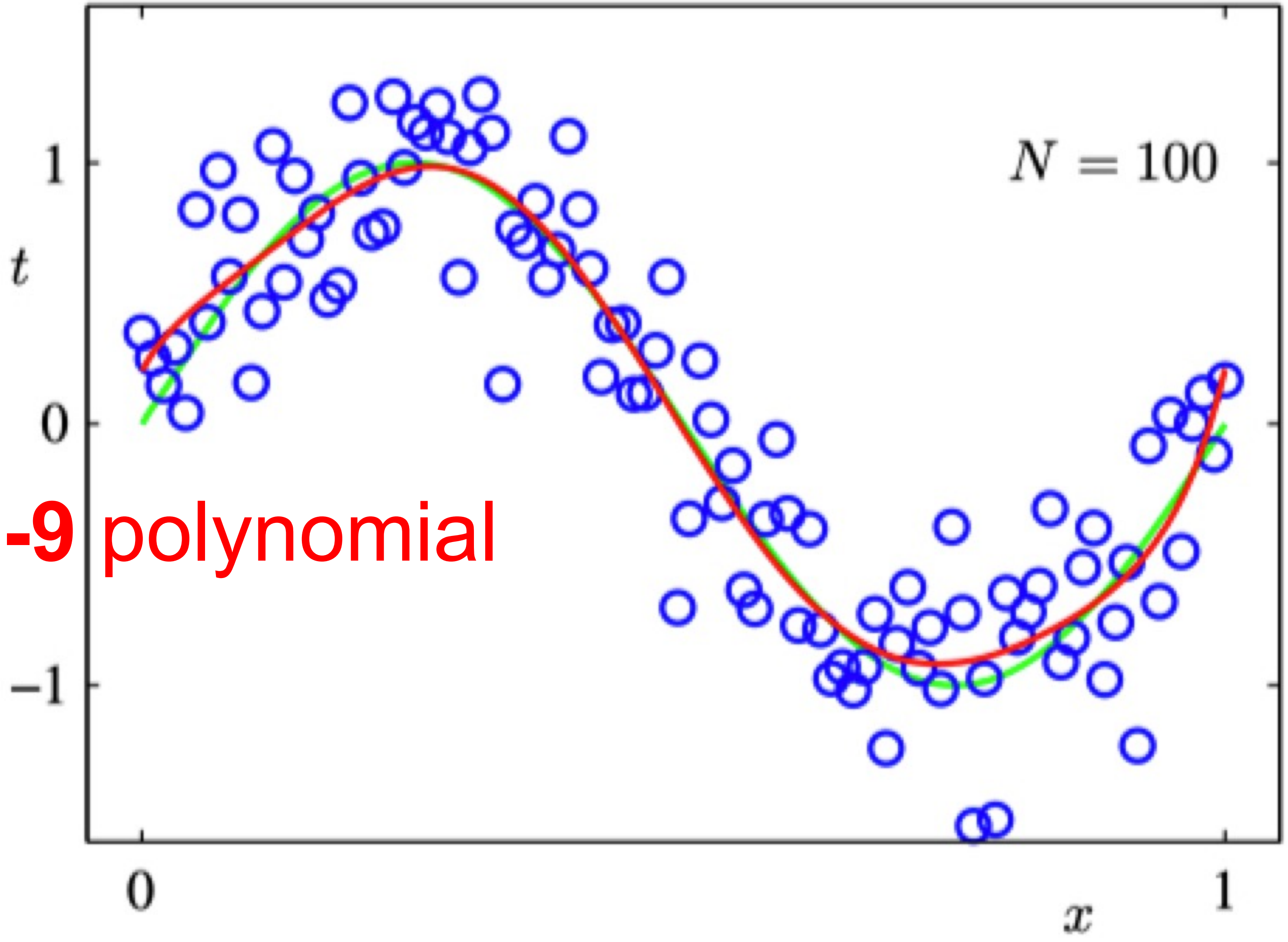
Error as a function of degree of polynomial

Error



Degree of polynomial

Dealing with Overfitting: (1) more data!



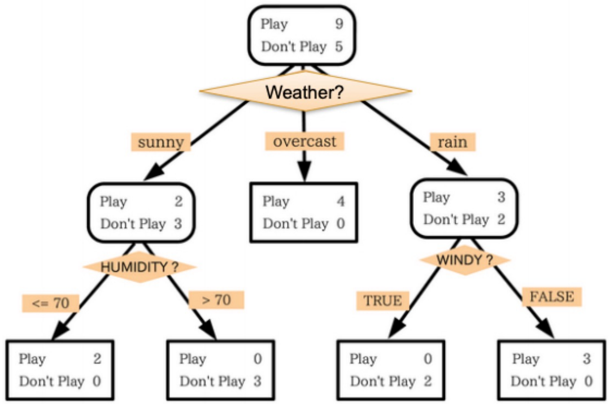
Degree-9 polynomial

Dealing with Overfitting: (2) regularization

minimize:

$$\widetilde{E}(\mathbf{w}) = \underbrace{\frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2}_{\text{Model's squared error}} + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2}_{\text{Penalty on parameters}}$$

Regularization in Classification

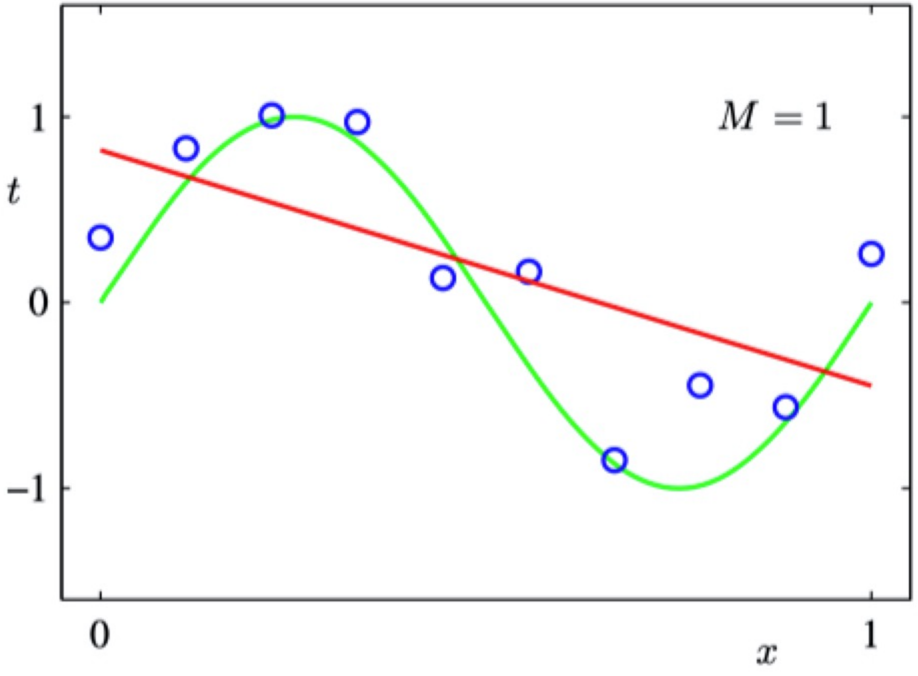


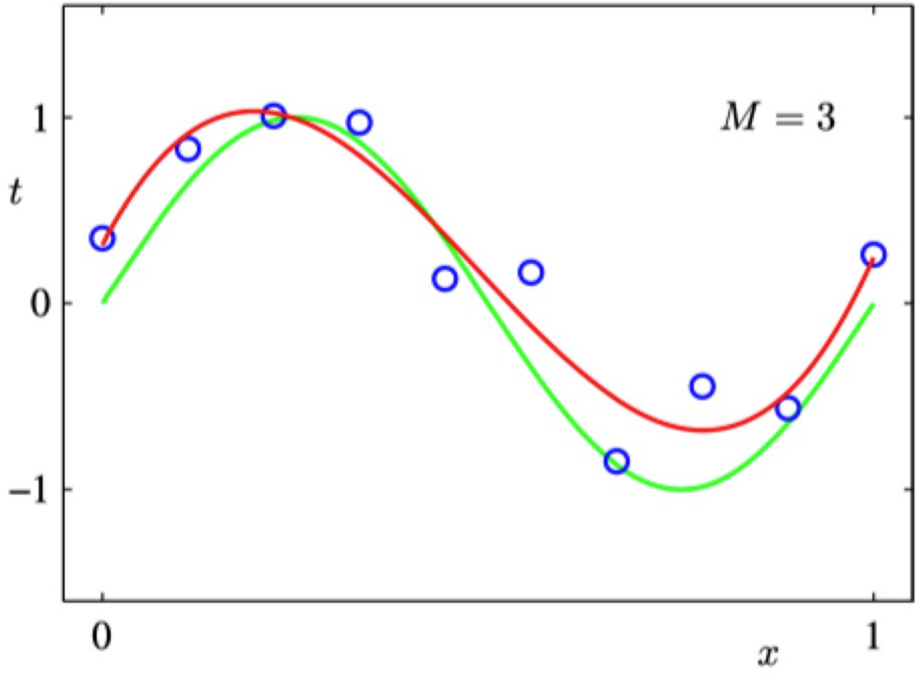
- Decision Trees
  - Limit the depth of the tree
  - Prune subtrees after training
- Support Vector Machines
  - Built-in, tunable regularization term
- Logistic Regression
  - Can add a tunable regularization term to MLE objective

Penalize large parameter values!

Underfitting: model is too simplistic!

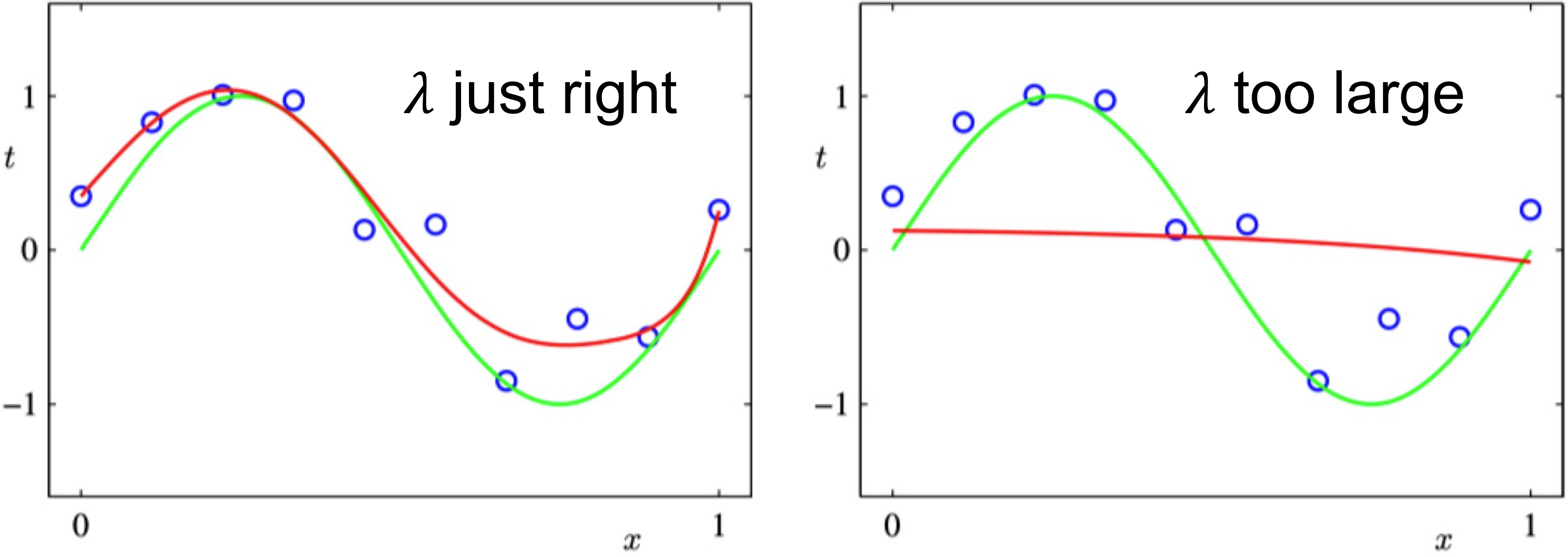
Training loss too high $\longrightarrow$ Use a more complex model





Dealing with Overfitting: (2) regularization

minimize: $$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$



Parameters vs Hyper-parameters

Example **hyper-parameters**

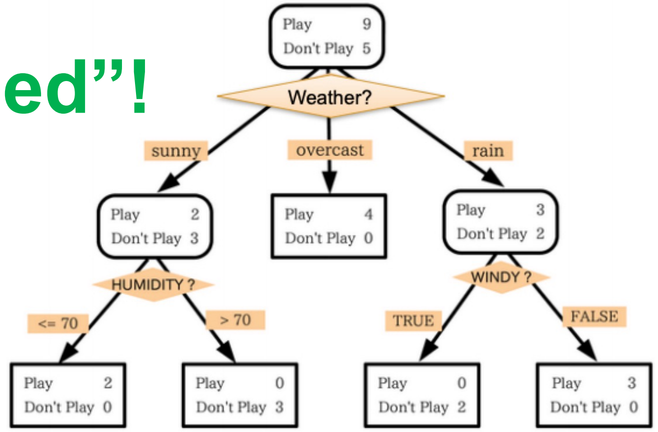
- Decision tree: maximum depth of the tree
- Polynomial regression: regularization coefficient $\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$

- Hyper-parameters are **determined through trial-and-error / cross validation**

Example **parameters: directly optimized, “learned”!**

- Decision tree: attributes you split on
- Logistic regression: weights $\boldsymbol{\beta}$

$$p(x;\boldsymbol{\beta}) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_d x_d)}}$$



Regression

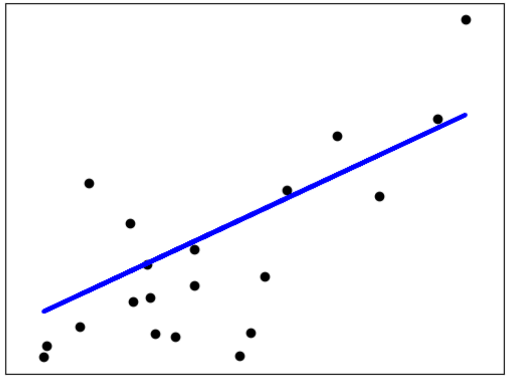
Examples:

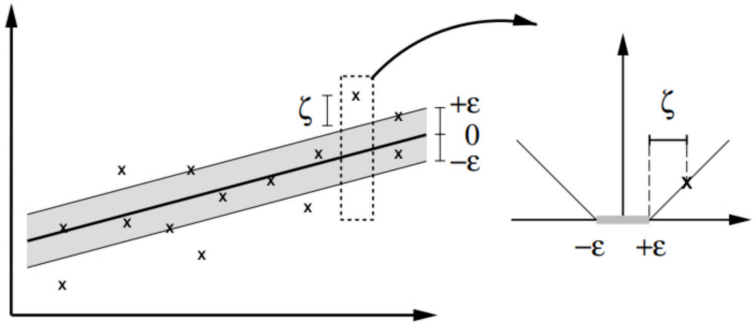
- Stock price prediction
- Forecasting epidemics
- Weather prediction

We will look at:

- Linear Regression
- Ridge Regression
- LASSO

https://medium.com/@ali_88273/regression-vs-classification-87c224350d69





Linear Regression

Data: $S = \{(x_i, y_i)\}_{i=1,\ldots,n}$

x_i: data example with **d** attributes

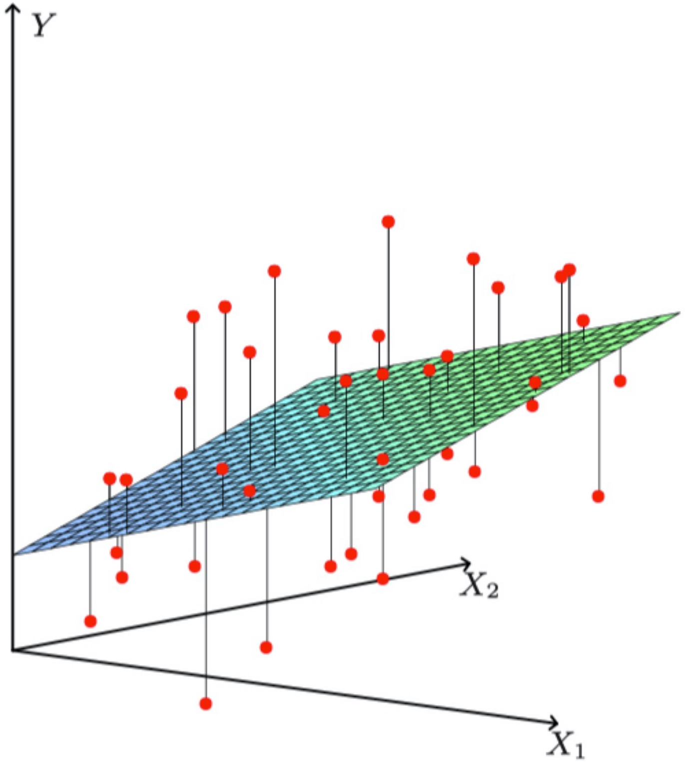
y_i: target of example (what you care about)

Model:

$$f(x;\boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

Loss function: Residual Sum of Squares

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - f(x_i;\boldsymbol{\beta})\right)^2$$



Linear Regression

Minimizing RSS to find $\boldsymbol{\beta}^*$:

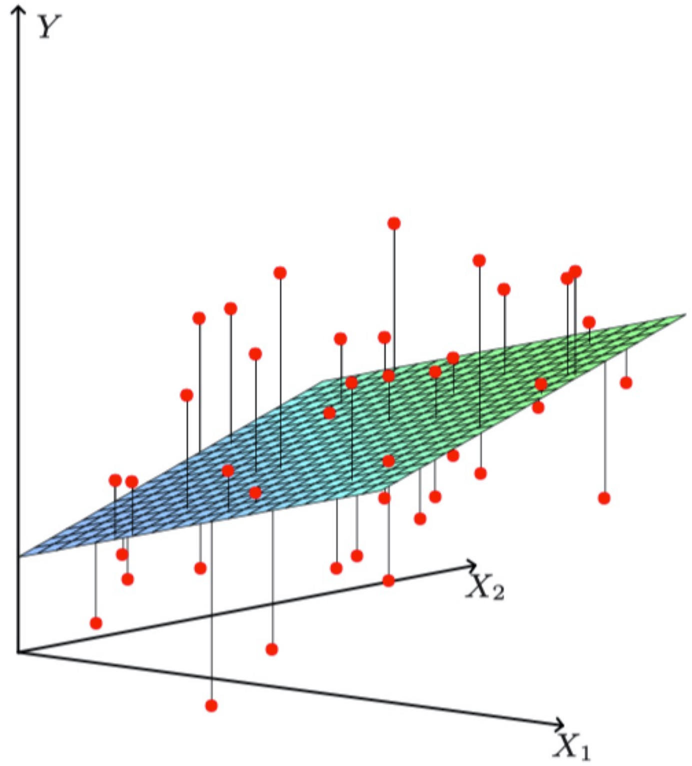
- Closed-form solution!
- If closed-form fails, can use standard optimization methods

Model:

$$f(x; \boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

Loss function: Residual Sum of Squares

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - f(x_i; \boldsymbol{\beta})\right)^2$$



Ridge Regression

- Linear Regression uses all features; model may be complicated
- **Ridge Regression** penalizes large parameter values

Model:

$$f(x;\boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

Loss function: Residual Sum of Squares + penalty term

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \big(y_i - f(x_i;\boldsymbol{\beta})\big)^2 + \boldsymbol{\lambda} \sum_{j=0}^{d} \beta_j^2$$

Lasso Regression

- **As in Ridge Regression, Lasso** penalizes large parameters
- Penalizes **absolute** instead of squared coefficient values
- **Zeroes out** more coefficients **BUT** optimization is more involved

Model:

$$f(x;\boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

Loss function: Residual Sum of Squares + penalty term

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \big(y_i - f(x_i;\boldsymbol{\beta})\big)^2 + \lambda \sum_{j=0}^{d} |\beta_j|$$

Example: Prostate Cancer

Stamey et al. (1989)

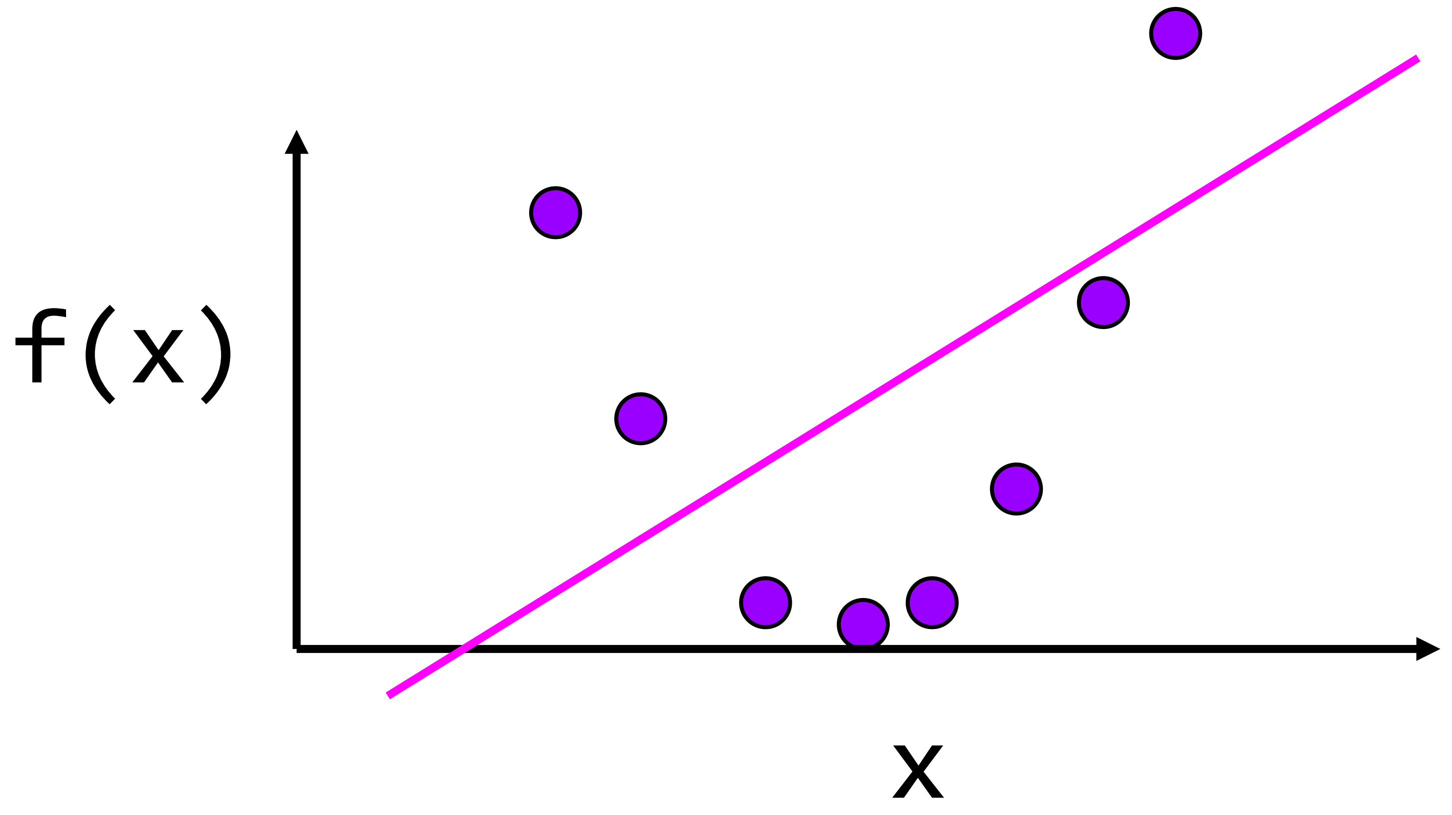
- x: cancer volume, prostate weight, age, …
- y: amount of prostate-specific antigen

Term	LR	Best Subset	Ridge	Lasso
Intercept	2.465	2.477	2.452	2.468
lcavol	0.680	0.740	0.420	0.533
lweight	0.263	0.316	0.238	0.169
age	−0.141		−0.046	
lbph	0.210		0.162	0.002
svi	0.305		0.227	0.094
lcp	−0.288		0.000	
gleason	−0.021		0.040	
pgg45	0.267		0.133	
Test Error	0.521	0.492	0.492	0.479
Std Error	0.179	0.143	0.165	0.164

Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

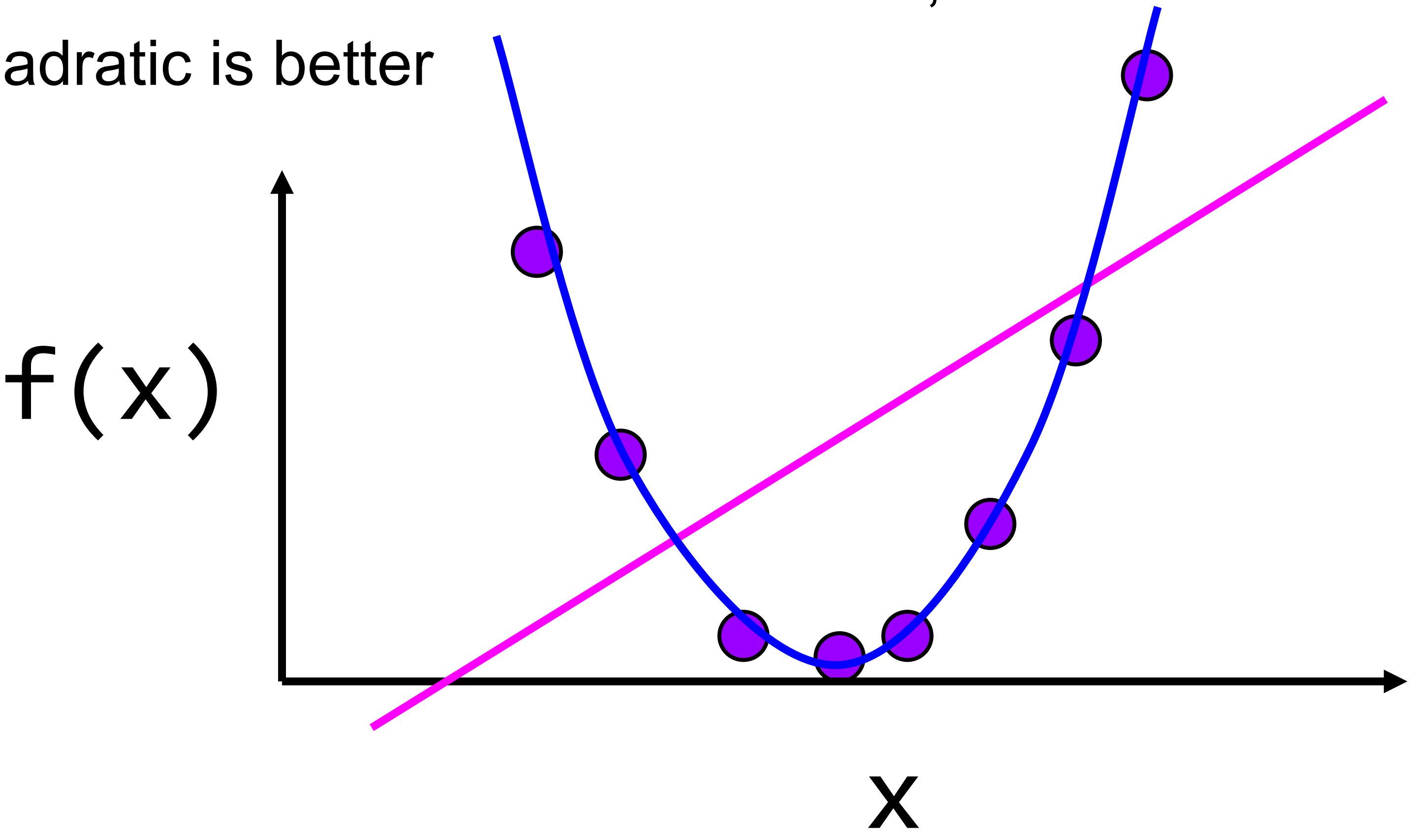
- Bias
    - Error from assumptions model makes about data
    - Linear model assumes data is linear, bad for data that isn't



Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

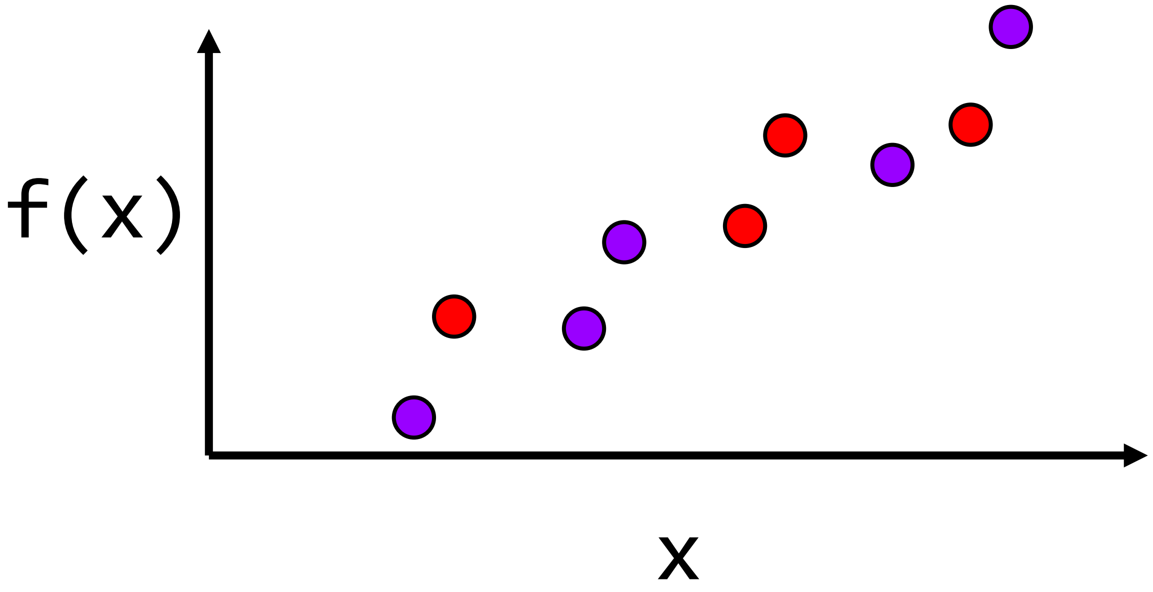
- Bias
    - Error from assumptions model makes about data
    - Linear model assumes data is linear, bad for data that isn't
    - Quadratic is better



Bias / Variance tradeoff

Based on Joseph Redmon's slides,
UW Computer Vision class

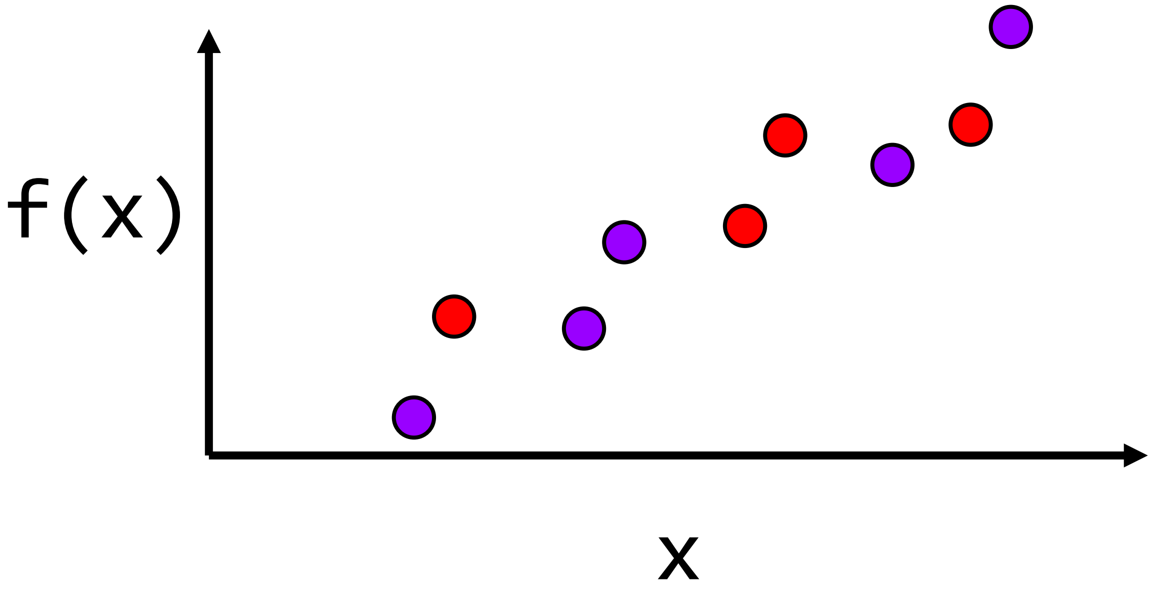
- Variance
  - Algorithm's sensitivity to noise
  - More complex algorithms are more sensitive!

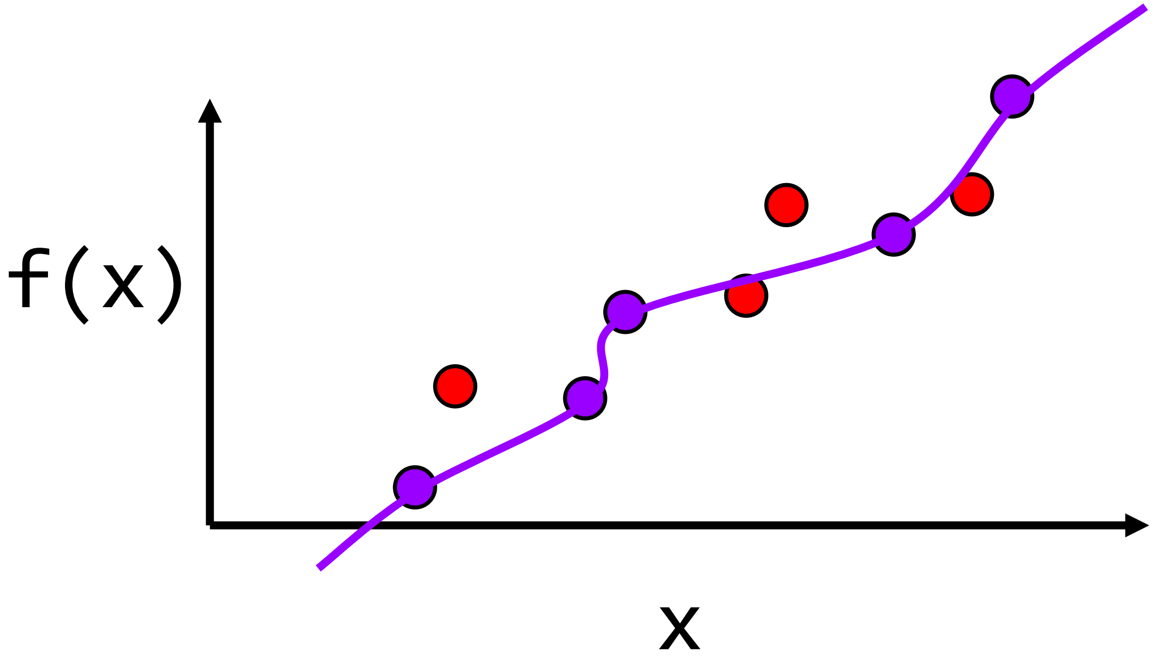


Bias / Variance tradeoff

Based on Joseph Redmon's slides,
UW Computer Vision class

- Variance
  - Algorithm's sensitivity to noise
  - More complex algorithms are more sensitive!

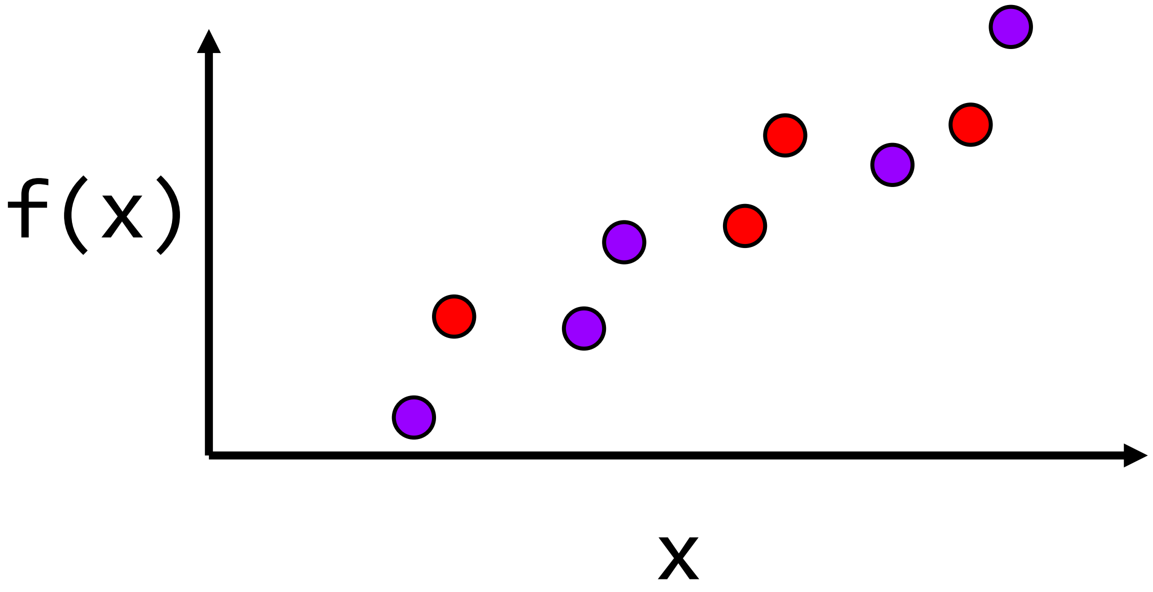


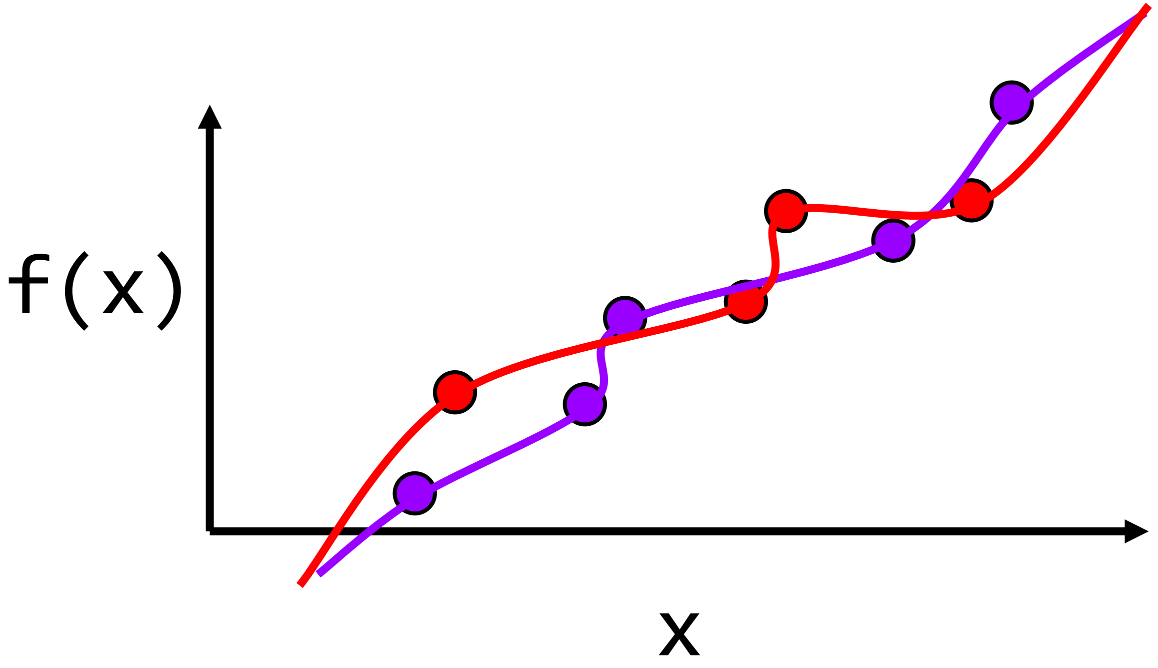


Bias / Variance tradeoff

Based on Joseph Redmon's slides,
UW Computer Vision class

- Variance
  - Algorithm's sensitivity to noise
  - More complex algorithms are more sensitive!
  - High variance hurts generalization, *overfitting*





Bias / Variance tradeoff

Based on Joseph Redmon's slides, UW Computer Vision class

Error = Noise + Bias + Variance

- Noise
  - Random variations in data
- Bias
  - **Error from assumptions** model makes about data
  - Less complex algorithms -> more assumptions about data
- Variance
  - Algorithm's **sensitivity to noise**
  - More complex algorithms are more sensitive!
  - High variance hurts generalization

Linear regression

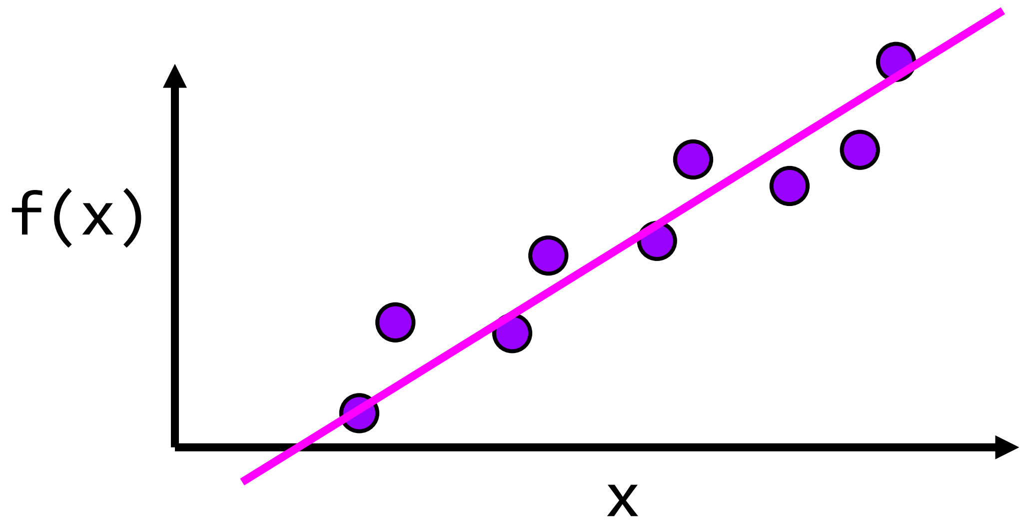
Based on Joseph Redmon's slides, UW Computer Vision class

- $f^*(x) = ax + b$

- **High bias**: linear assumption

- **Low variance**

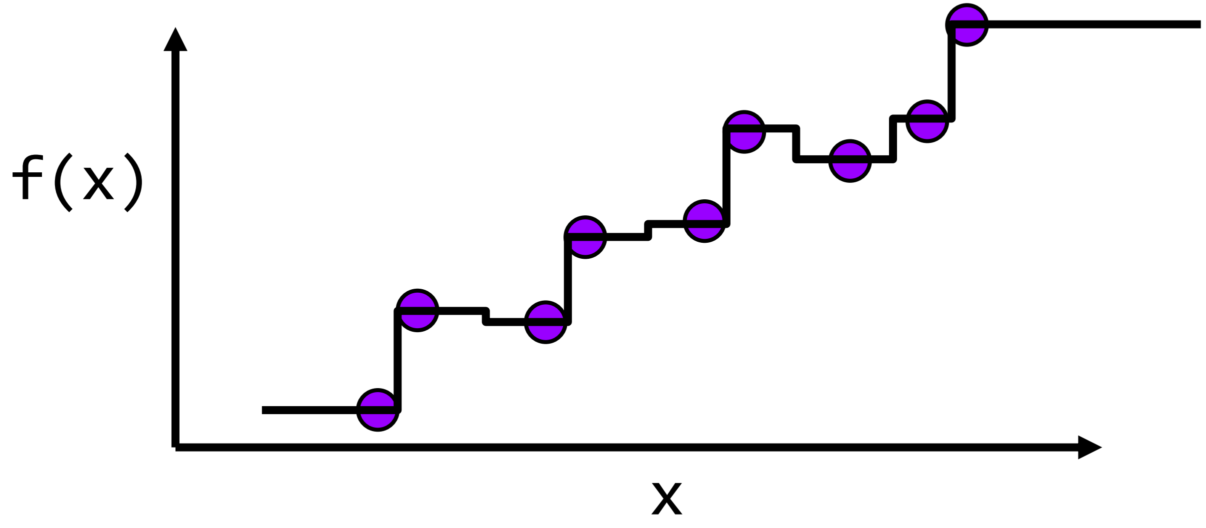
- Benefits:
  - Closed form solution
  - Fast to compute for new data
- Weaknesses:
  - Not very powerful, assumes linear
  - Underfit more interesting data



Nearest neighbor regression

Based on Joseph Redmon's slides, UW Computer Vision class

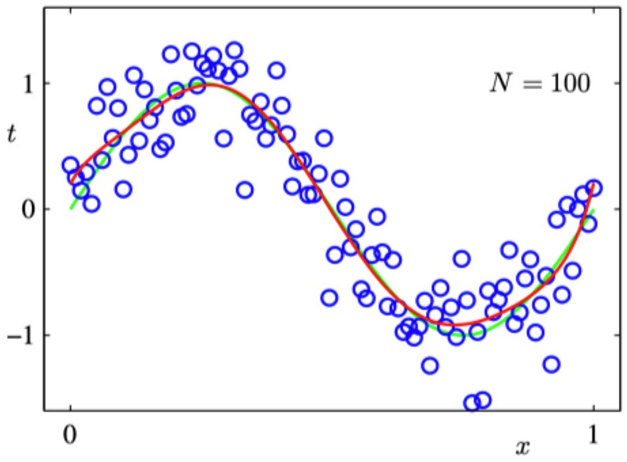
- $f^*(x) = f(x')$ for nearest x' in training set
- **Low bias**: no assumptions about data
- **High variance**: very sensitive to training set
- Benefits:
    - Super easy to implement
    - Easy to understand
    - Arbitrarily powerful, esp with lots of data
- Weaknesses:
    - Hard to scale
    - Prone to **overfitting** to noise



ML as an Optimization problem: Regression

minimize:

$$\widetilde{E}(\mathbf{w}) = \underbrace{\frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2}_{\text{Model's squared error}} + \underbrace{\frac{\lambda}{2}\|\mathbf{w}\|^2}_{\text{Penalty on parameters}}$$



Model's squared error

Penalty on parameters

ML as an Optimization problem: Logistic Regression

$$p(x;\boldsymbol{\beta}) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_d x_d)}}$$

Data: $S = \{(x_i, y_i)\}_{i=1,\ldots,n}$

x_i: example with d attributes (age, #pregnancies, …)
y_i: cervical cancer diagnosis (0 or 1)

Maximum Likelihood Estimation (MLE)

Likelihood of observing the data for a given $\boldsymbol{\beta}$:

$$\prod_{i=1}^{n} p(x_i;\boldsymbol{\beta})^{y_i} \times \left(1-p(x_i;\boldsymbol{\beta})\right)^{1-y_i}$$

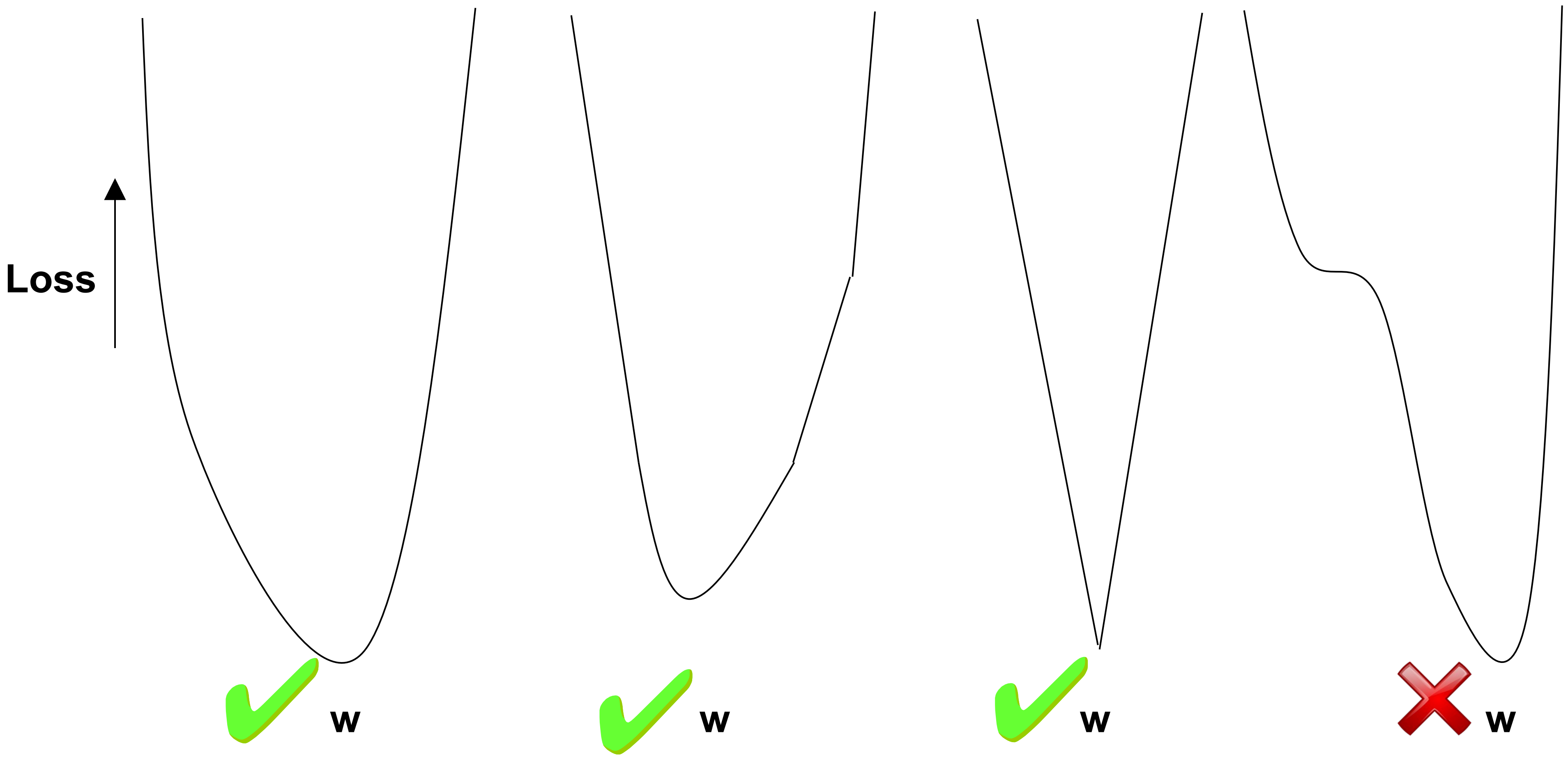
MLE seeks parameters $\boldsymbol{\beta}$ that maximize the likelihood

The optimal parameters, $\boldsymbol{\beta}^*$, can be found by optimization

Convexity

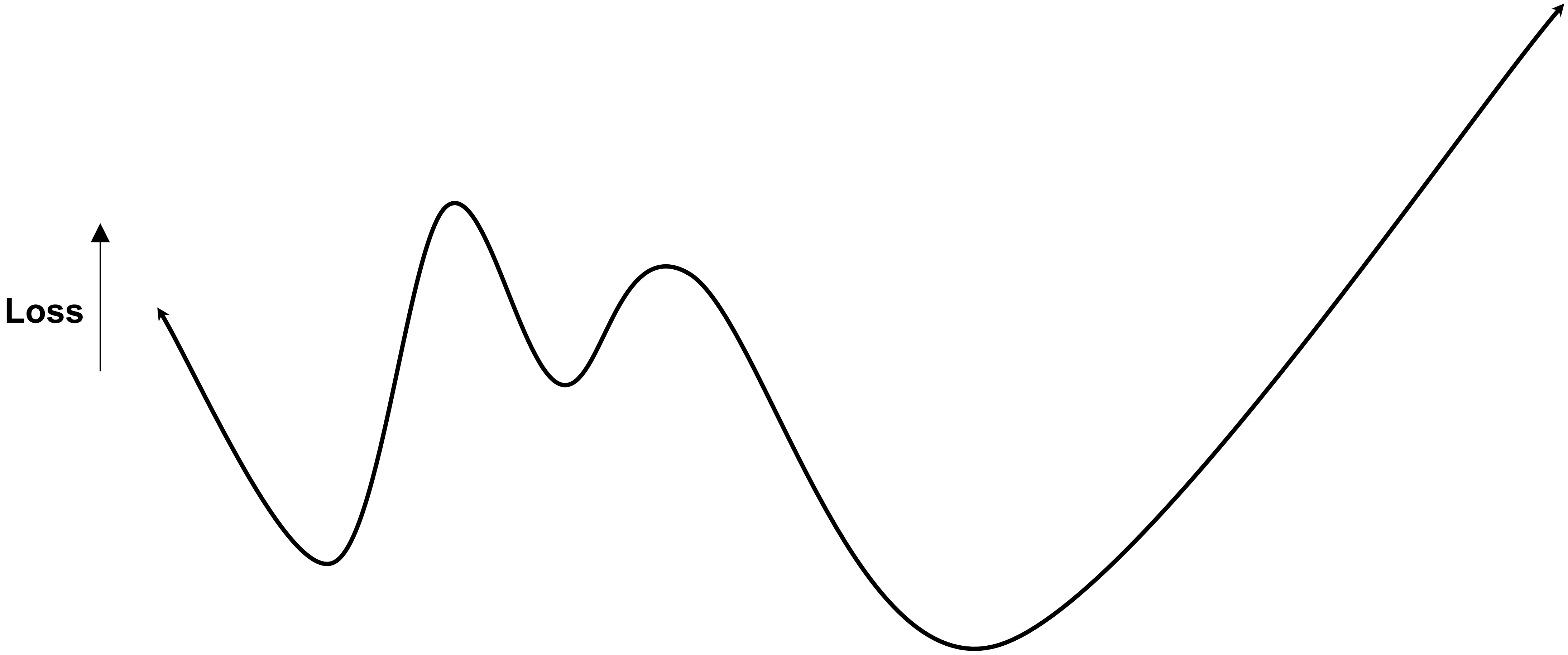
- Graphically… which of these is convex?

For convex problems, gradient descent gets us to **global minima**. It's that easy!



Non-convexity

Usually no easy way to find global or local optima, harder to optimize

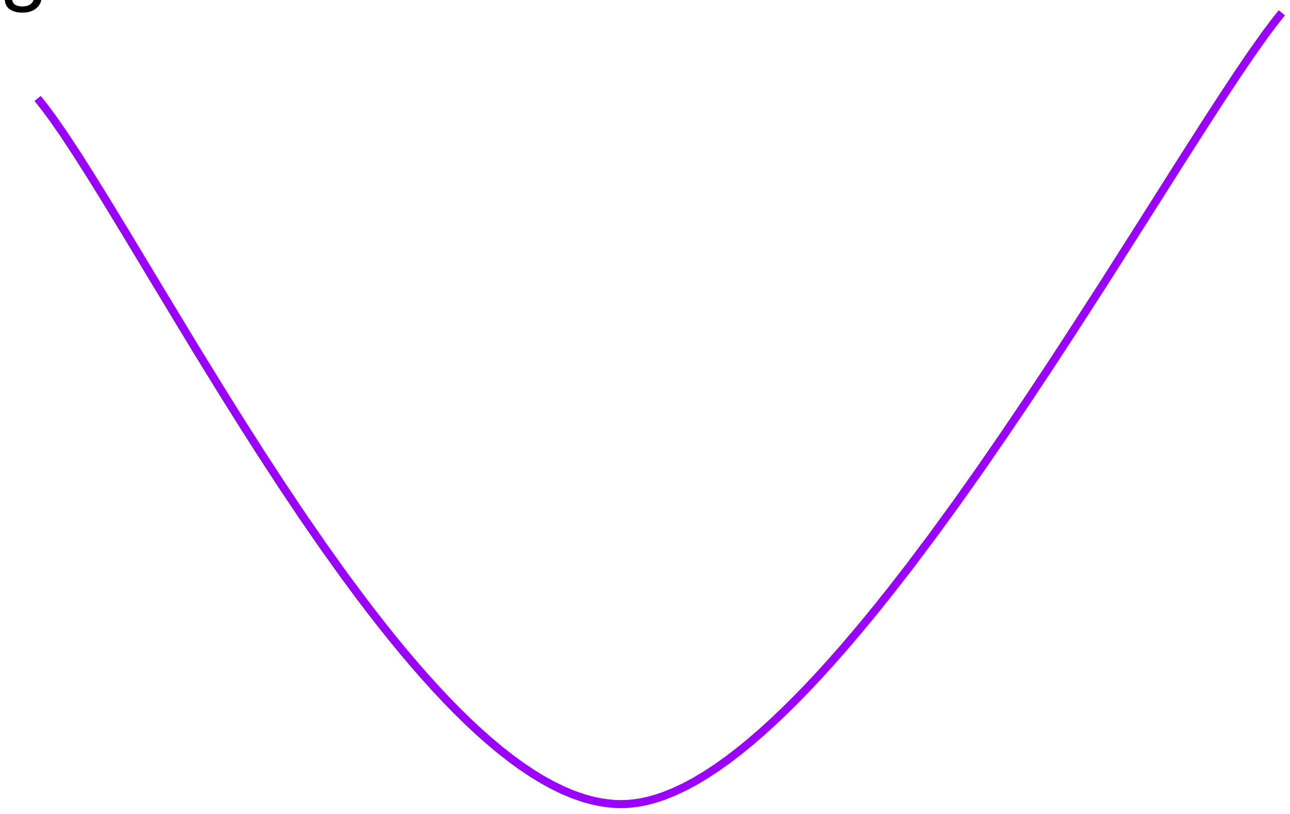


Gradient descent

Based on Joseph Redmon's slides, UW Computer Vision class

For some loss function L(**w**), gradient ∇L(**w**) points towards in direction of steepest ascent.

In 1d, either points left or right

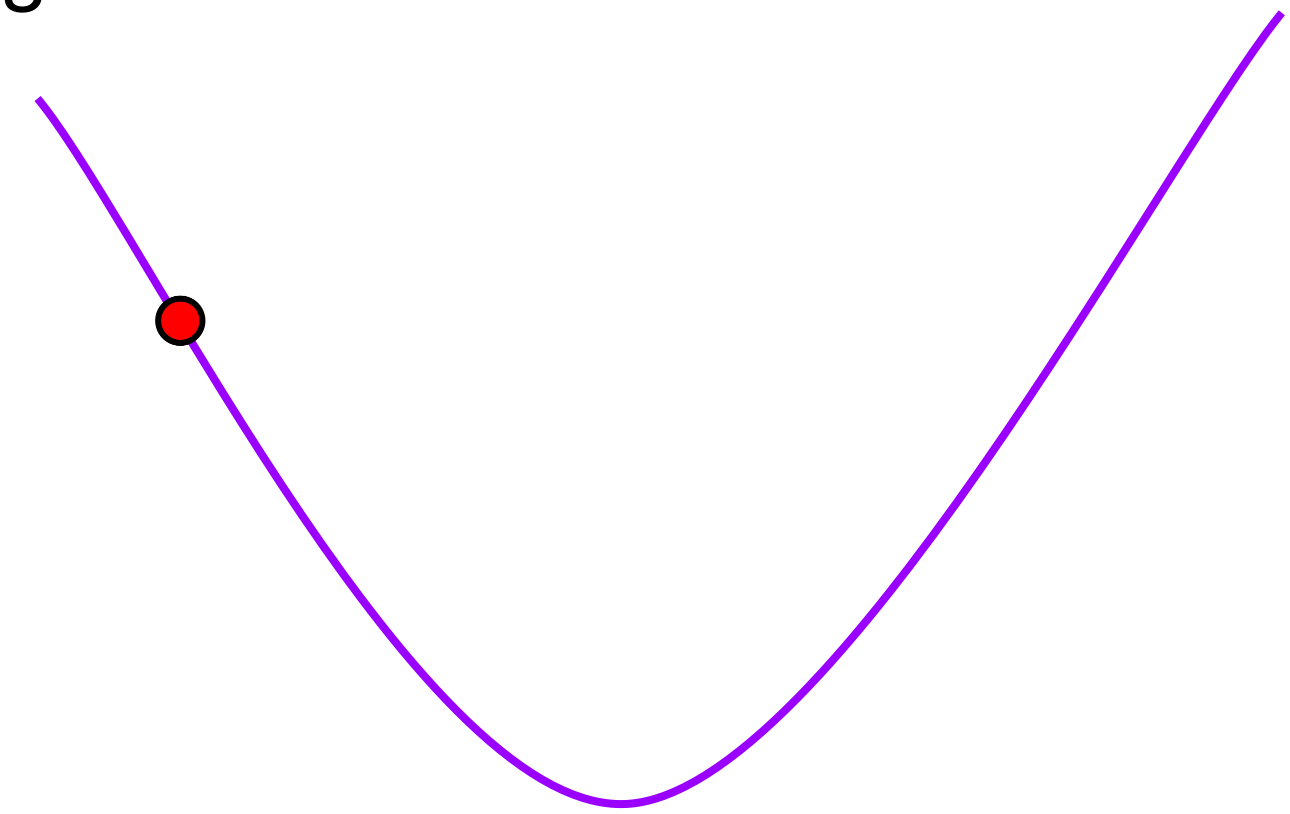


Gradient descent

Based on Joseph Redmon's slides,
UW Computer Vision class

For some loss function L(**w**), gradient ∇L(**w**) points towards in direction of steepest ascent.

In 1d, either points left or right

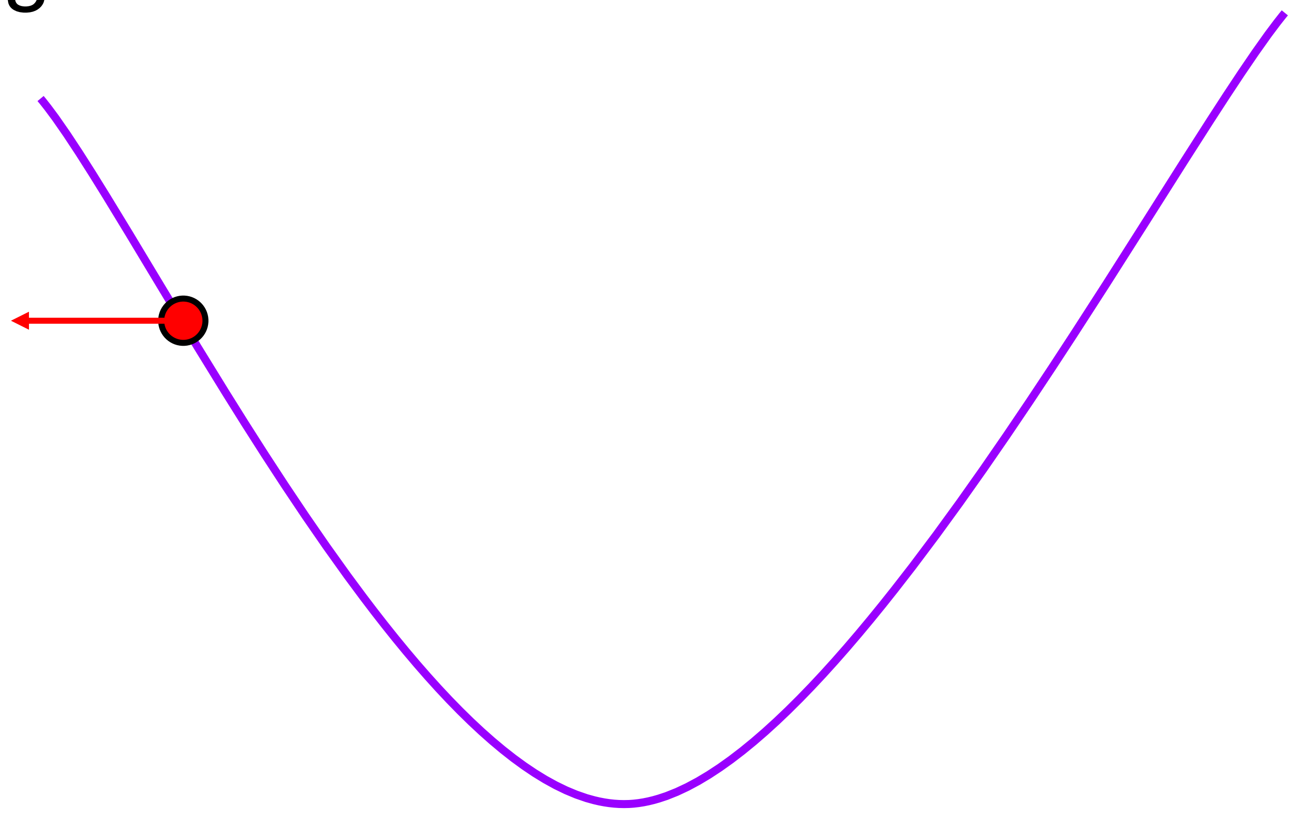


Gradient descent

Based on Joseph Redmon's slides,
UW Computer Vision class

For some loss function L(**w**), gradient ∇L(**w**) points towards in direction of steepest ascent.

In 1d, either points left or right

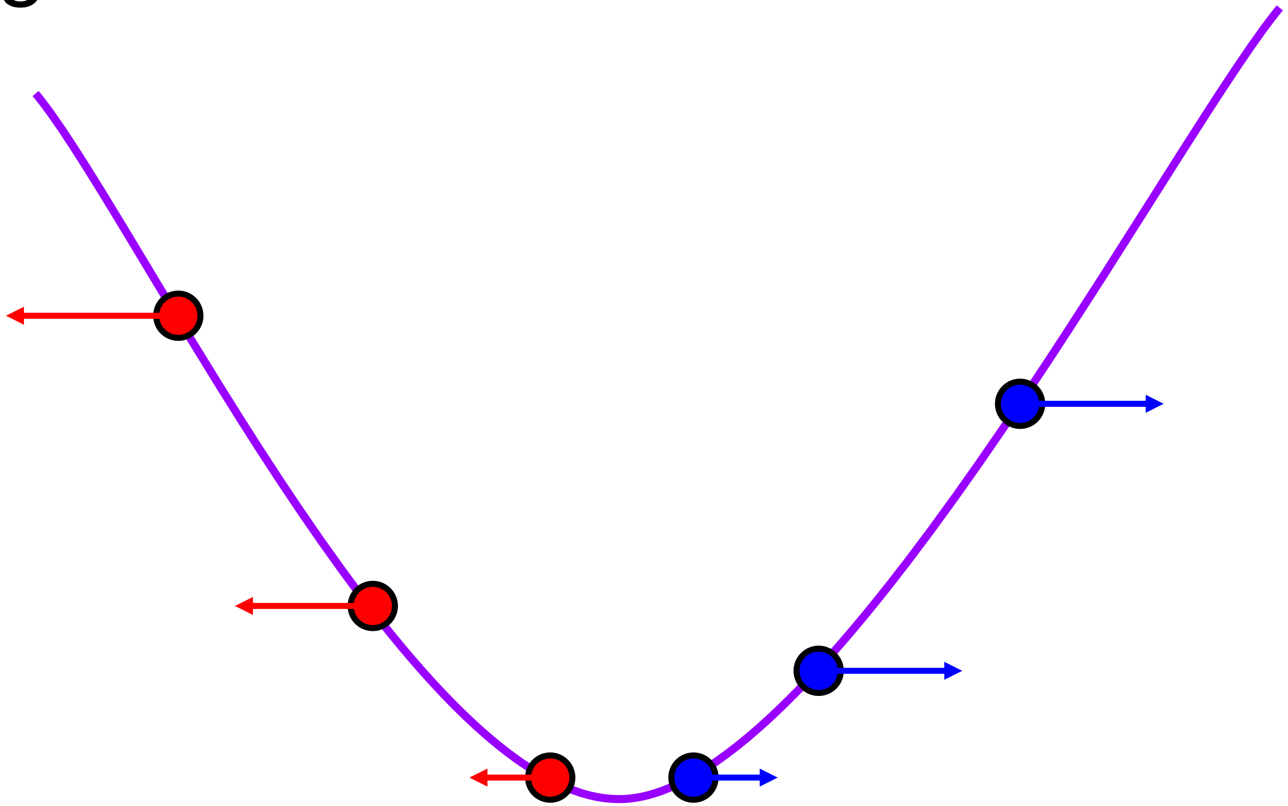


Gradient descent

Based on Joseph Redmon's slides, UW Computer Vision class

For some loss function L(**w**), gradient ∇L(**w**) points towards in direction of steepest ascent.

In 1d, either points left or right



Gradient descent

Based on Joseph Redmon's slides, UW Computer Vision class

For some loss function L(**w**), gradient ∇L(**w**) points towards in direction of steepest ascent.

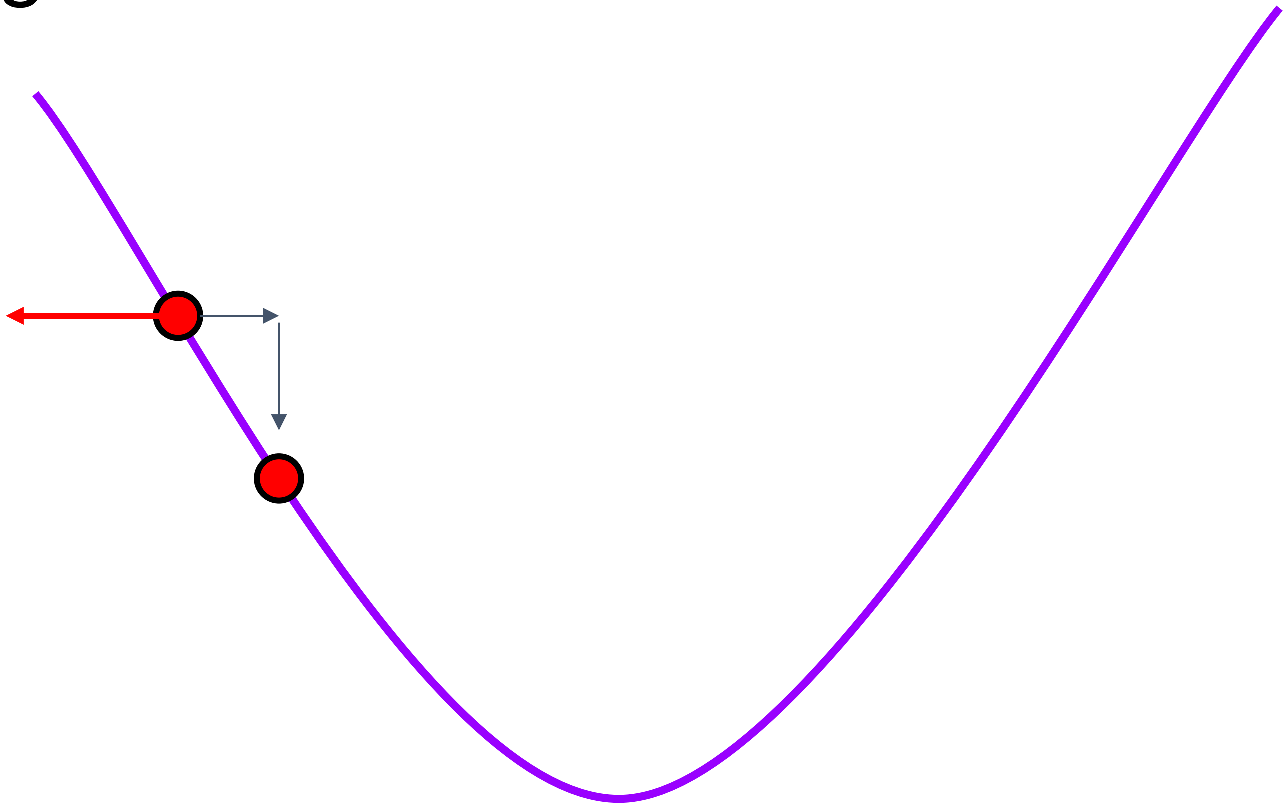
In 1d, either points left or right

Algorithm:

Take derivative

Move slightly in other direction

Repeat



Gradient descent

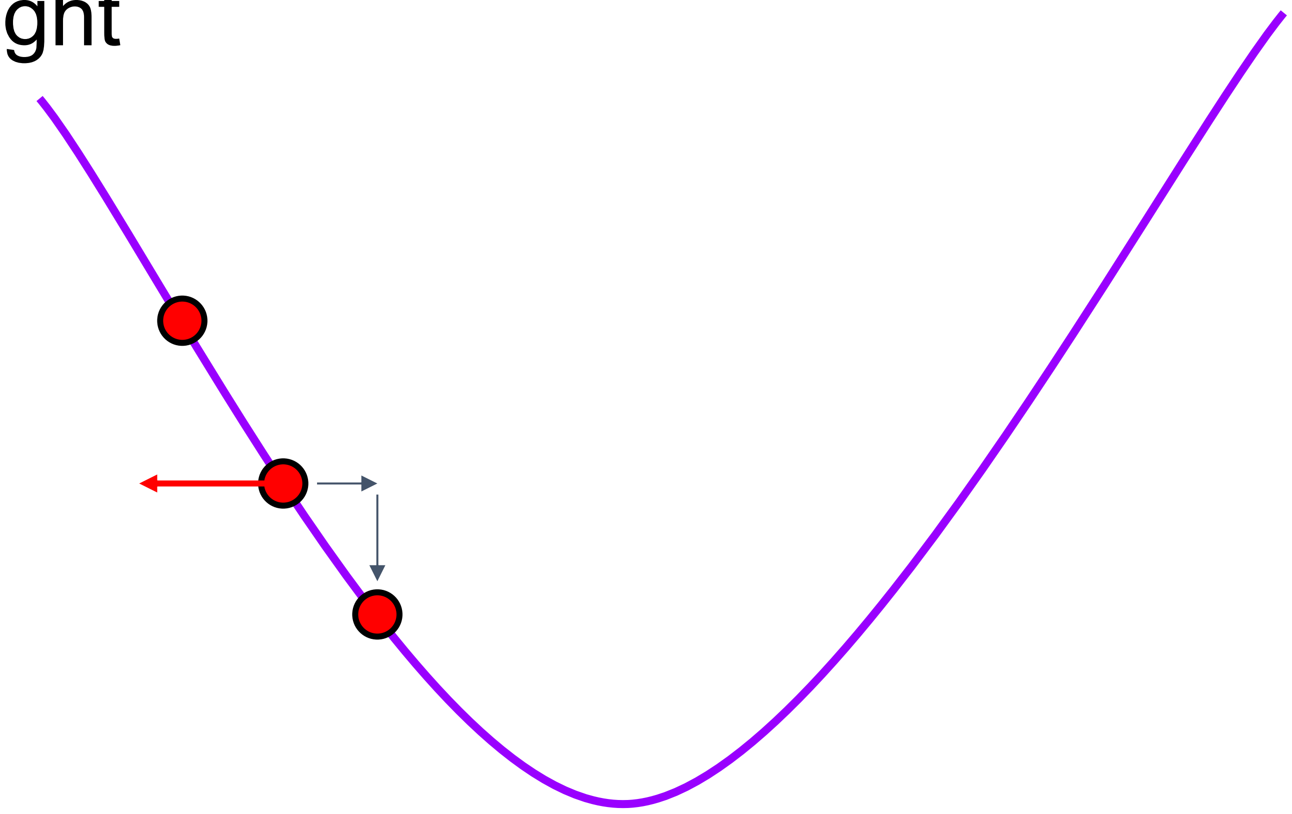
Based on Joseph Redmon's slides, UW Computer Vision class

For some loss function L(**w**), gradient ∇L(**w**) points towards in direction of steepest ascent.

In 1d, either points left or right

Algorithm:

Take derivative
Move slightly in other direction
Repeat



Gradient descent

Based on Joseph Redmon's slides,
UW Computer Vision class

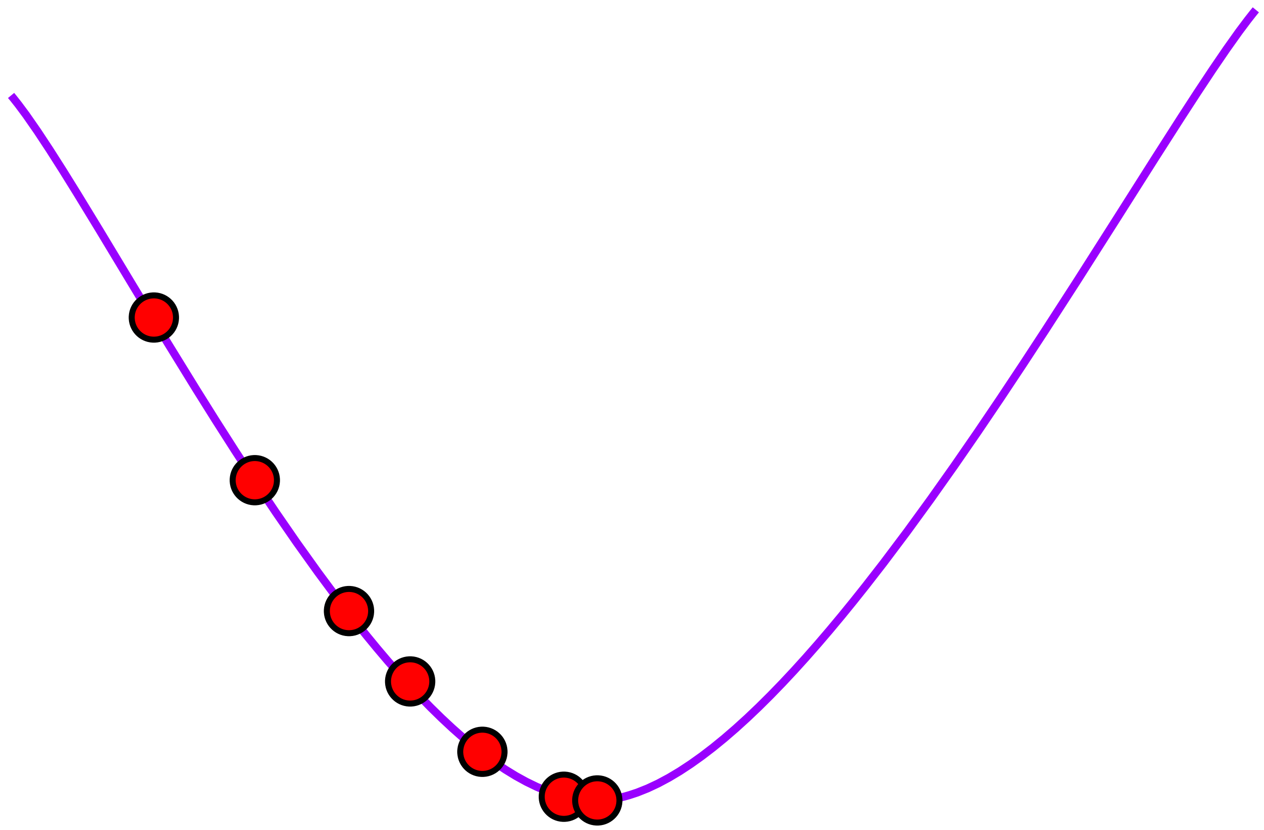
Algorithm:

Take derivative

Move slightly in other direction

Repeat

End up at local optima



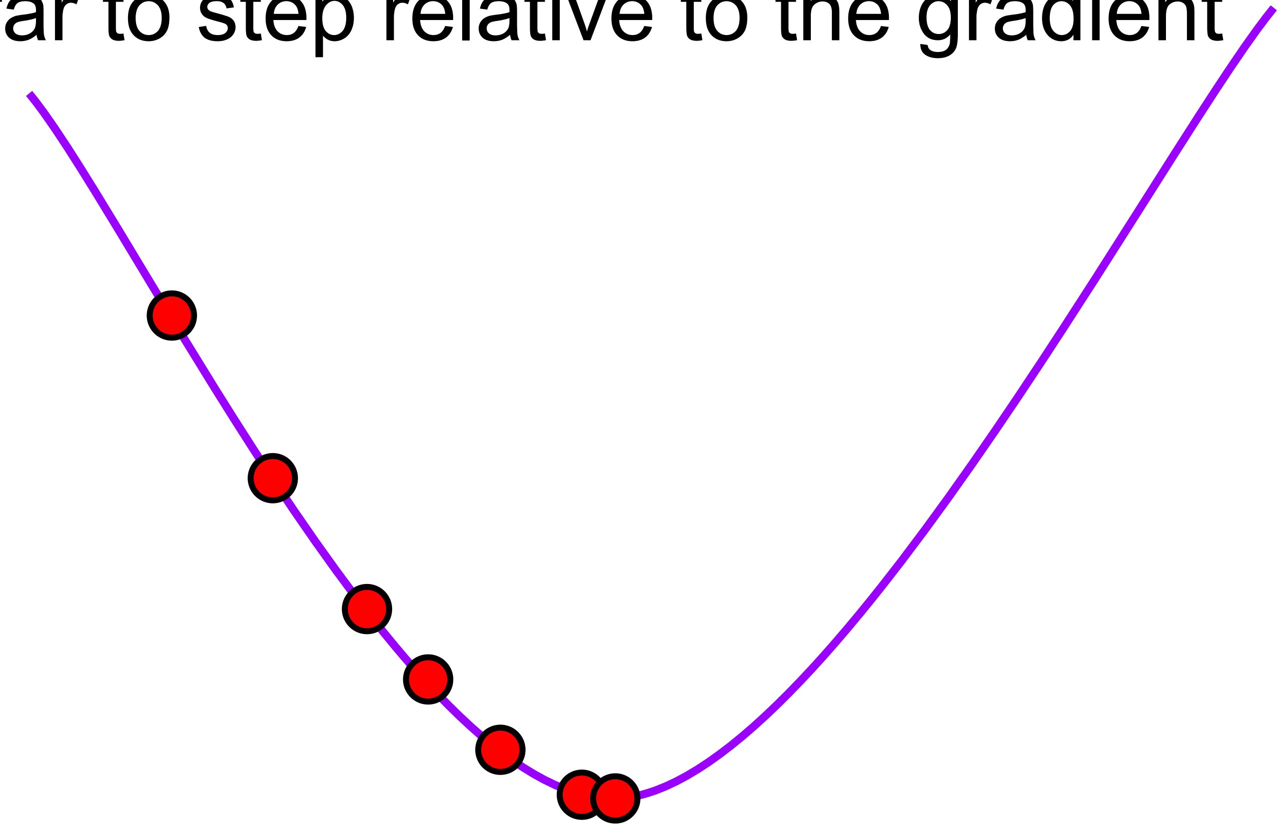
Gradient descent

Based on Joseph Redmon's slides,
UW Computer Vision class

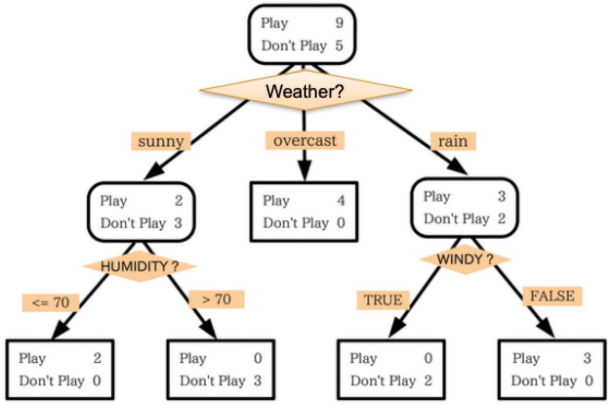
Formally:

$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla L(\mathbf{w})$

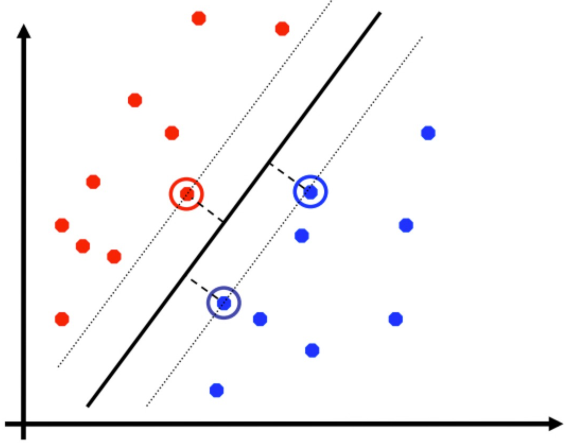
Where η is *step size*, how far to step relative to the gradient



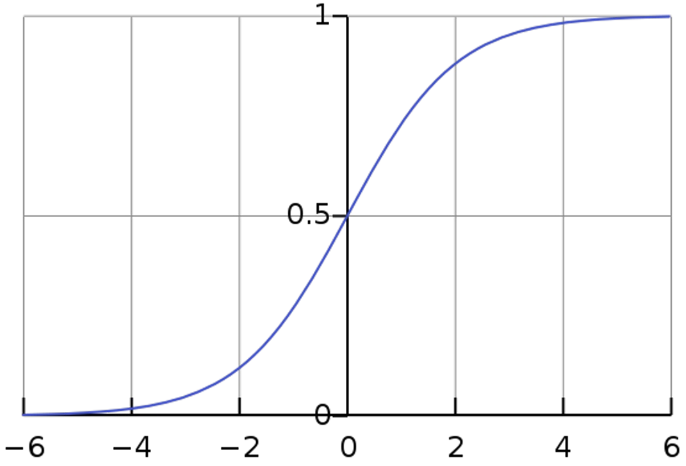
Optimization in Classification



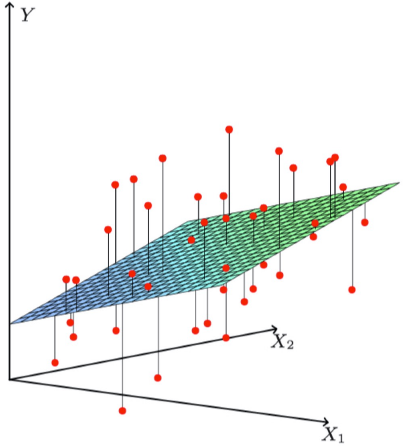
- **Decision Trees**
  - Very hard to find optimal tree (min. misclassification)
  - Top-down heuristic as shown in Lecture 1



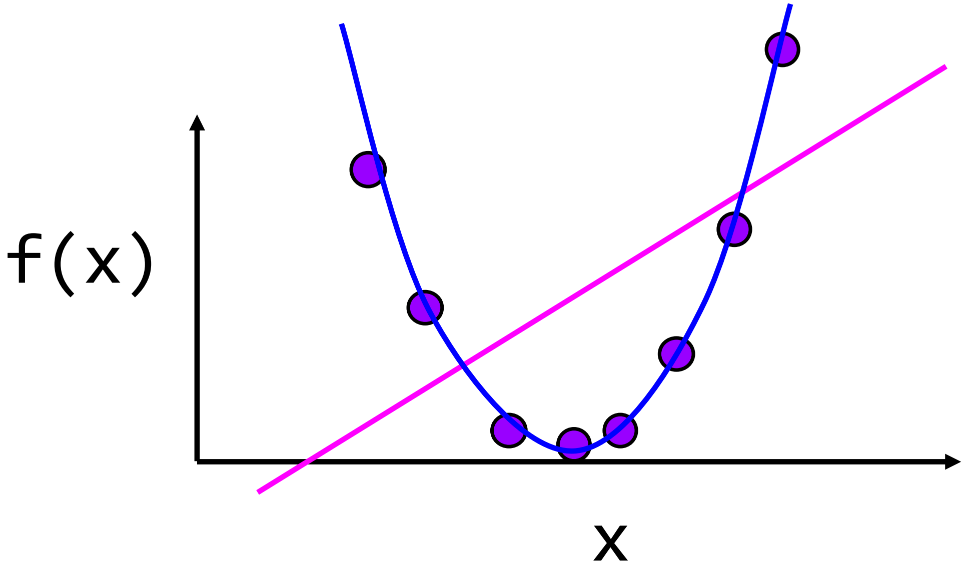
- **Support Vector Machines**
  - Convex optimization!
- **Logistic Regression**
  - Convex optimization!



Recap



- Regression models:
  - Linear Regression
  - Ridge Regression
  - Lasso
- Overfitting and Underfitting
- Bias-Variance Tradeoff
- Gradient Descent for ML



Error = Noise + Bias + Variance

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla L(\mathbf{w})$$