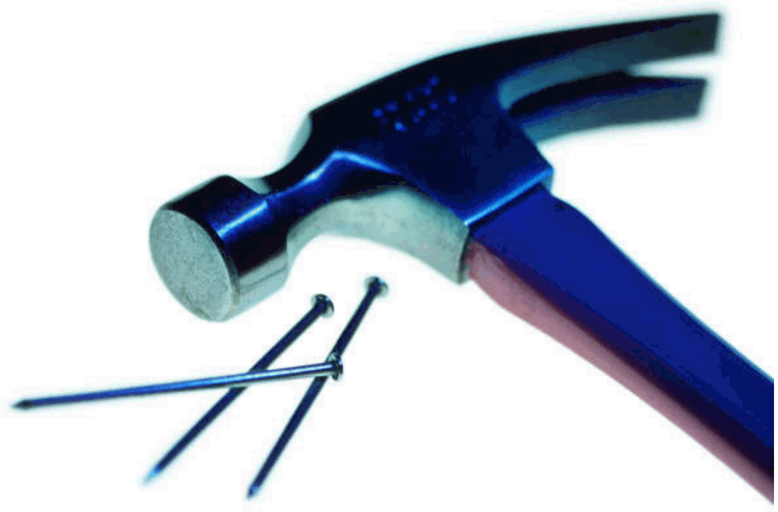


Machine Learning: The Optimization Perspective

Konstantin Tretyakov
http://kt.era.ee

AACIMP Summer School 2015

STACC Software Technology and Applications Competence Center

Award Medallion BIOS v6.0, An Energy Star Ally
Copyright (C) 1984-2001, Award Software, Inc.

ASUS P4T533-C ACPI BIOS Revision 1007 Beta 001

Intel(R) Pentium(R) 4 2000 MHz Processor
Memory Test : 262144K OK

Award Plug and Play BIOS Extension v1.0A
Initialize Plug and Play Cards...
PNP Init Completed

Detecting Primary Master ... MAXTOR 6L040J2
Detecting Primary Slave ... ASUS CD-S520/A
Detecting Secondary Master... Skip
Detecting Secondary Slave ... None_

EPA POLLUTION PREVENTER

Press DEL to enter SETUP, Alt-F2 to enter EZ flash utility
08/20/2002-I850E/ICH2/W627-P4T533-C

Quiz

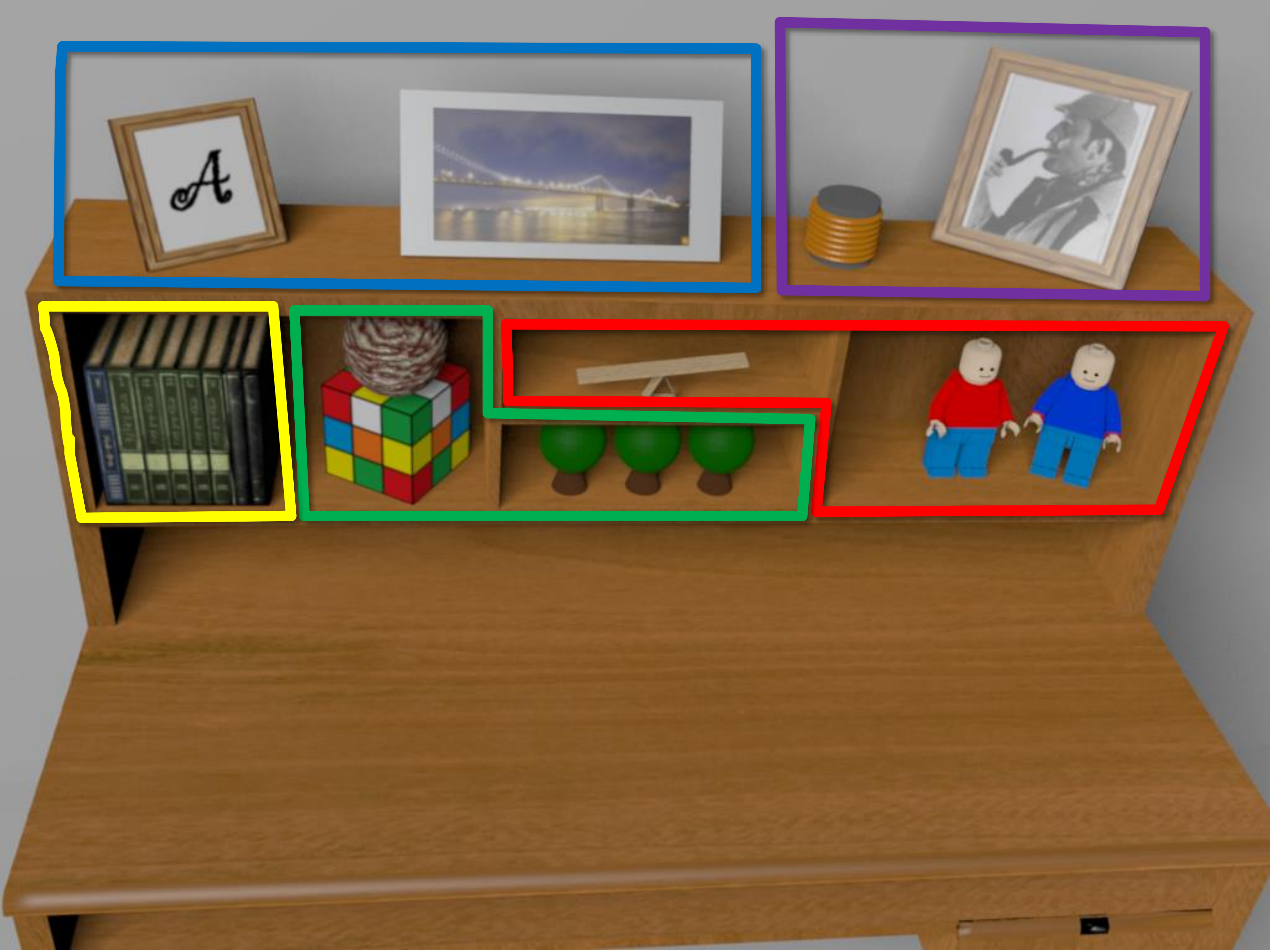
- Machine learning is ________________________.

- Two important components of machine learning are __________ and __________.

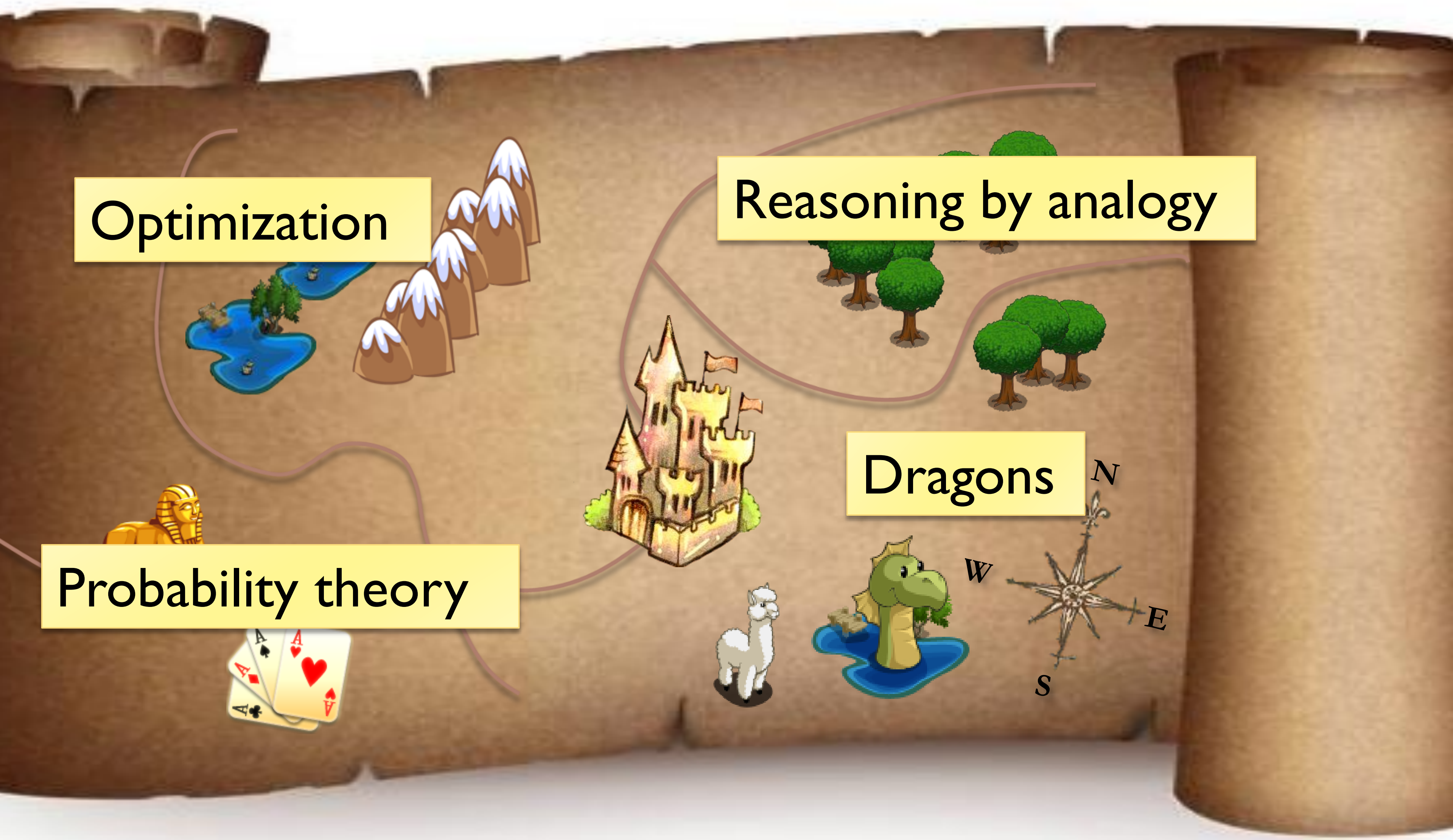
- The parameters of a machine learning model are estimated using a _______________. The quality of the model is measured using a ______________.

Quiz

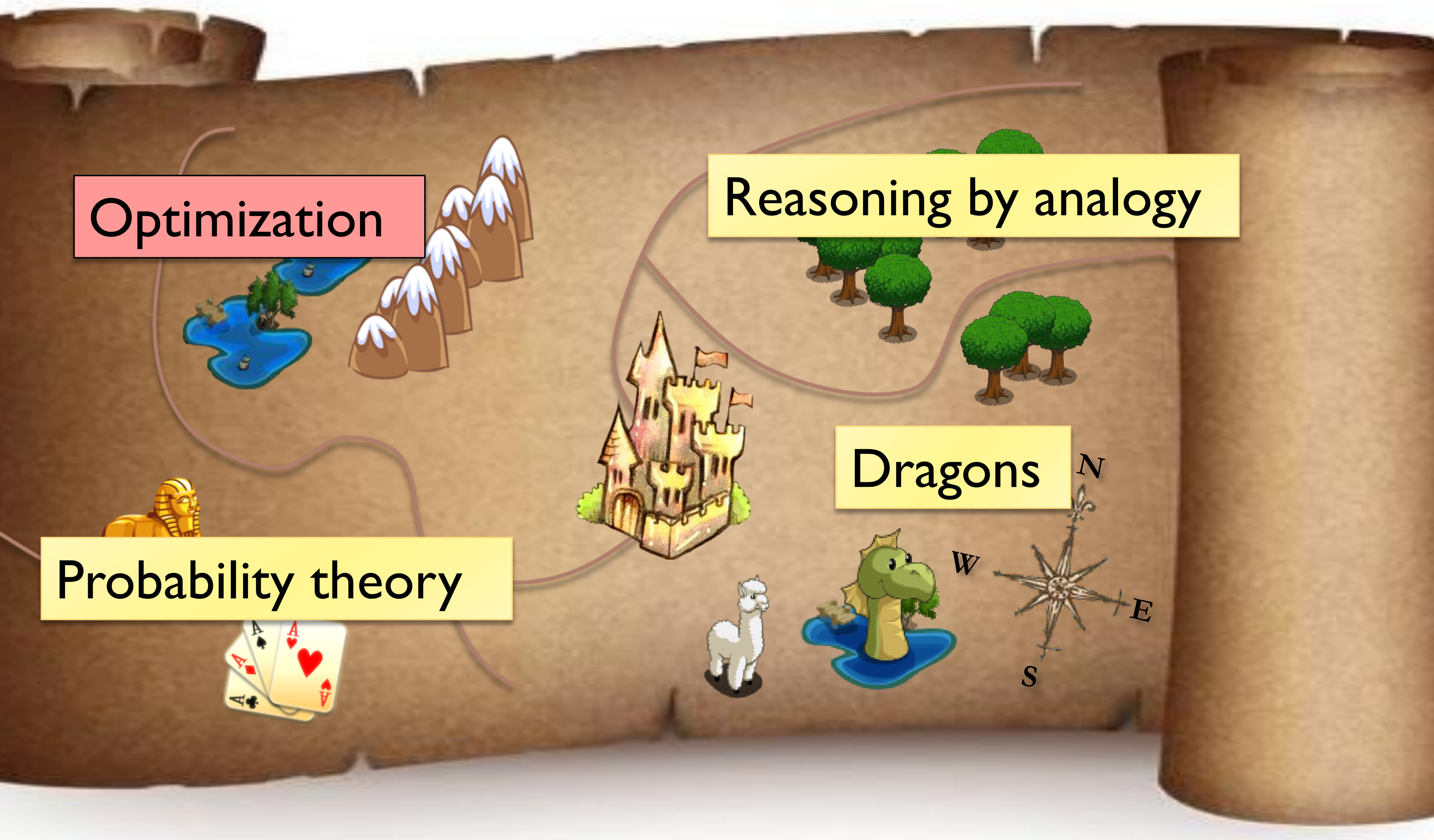
- Parameter estimation methods: ____, ____, ____.
- *Supervised learning* denotes the problem of inferring a ________________ from _____ data.
- The following supervised learning methods were mentioned yesterday: __________


A

The Land of Machine Learning



The Land of Machine Learning



Optimization

Given **a function**

$$f(\mathbf{x}) : \mathbf{x} \rightarrow \mathbb{R}$$

find the argument x resulting in the **optimal** value.

Special cases of optimization

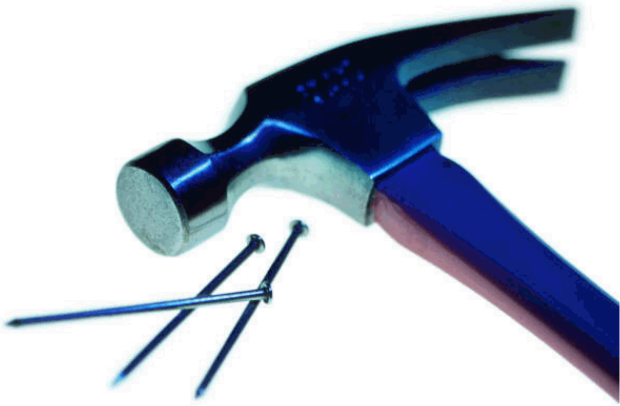
- Machine learning
- …

Special cases of optimization

- Machine learning
- Algorithms and data structures
- General problem-solving
- Management and decision-making

Special cases of optimization

- Machine learning
- Algorithms and data structures
- General problem-solving
- Management and decision-making
- Evolution
- *The Meaning of Life?*



What do you need to know about optimization?

What do you need to know about optimization?

1. Optimization is **important**
2. Optimization is **possible**

What do you need to know about optimization?

1. Optimization is **important**
2. Optimization is **possible***

* Basic **techniques**

- Constrained / Unconstrained
- Analytic / Iterative
- Continuous / Discrete

Optimization task

Given **a function**

$$f(\mathbf{x}) : \mathbf{x} \to \mathbb{R}$$

find the argument x resulting in the **optimal** value.

Constrained optimization task

Given **a function**

$$f(\mathbf{x}) : \mathbf{x} \to \mathbb{R}$$

find the argument x resulting in the **optimal** value, subject to

$$\mathbf{x} \in \mathcal{C}$$

Optimization methods

In principle, **x** can be anything:

- **Discrete**
  - Value (e.g. a name)
  - Structure (e.g. a graph, plaintext)
  - Finite / infinite
- **Continuous***
  - Real-number, vector, matrix, …
  - Complex-number, function, …

Optimization methods

In principle, **f** can be anything:

- **Random oracle**
- **Structured**
- **Continuous**
- **Differentiable**
- **Convex**
- **…**

Optimization methods

		Knowledge about f	
		Not much	A lot
Type of x	Discrete	Combinatorial search: Brute-force, Stepwise, MCMC, Population-based, …	Algorithmic
	Continuous	Numeric methods: Gradient-based, Newton-like, MCMC, Population-based, …	Analytic

Optimization methods

Finding a **weight-vector** w, minimizing the model error

		Knowledge about f	
		Not much	**A lot**
Type of x		**Combinatorial search:** Brute-force, Stepwise, MCMC, Population-based, …	**Algorithmic**
	Continuous	**Numeric methods:** Gradient-based, Newton-like, MCMC, Population-based, …	**Analytic**

Optimization methods

Finding a **weight-vector** w, minimizing the model error, in a **fairly general case**

		Knowledge about f	
		Not much	**A lot**
		Combinatorial search: Brute-force, Stepwise, MCMC, Population-based, …	**Algorithmic**
	Continuous	**Numeric methods:** Gradient-based, Newton-like, MCMC, Population-based, …	**Analytic**

Optimization methods

Finding a **weight-vector** w, minimizing the model error, in a **very general case**

		Knowledge about f	
		Not much	**A lot**
		Combinatorial search: **Brute-force, Stepwise, MCMC, Population-based, …**	**Algorithmic**
	Continuous	**Numeric methods:** **Gradient-based, Newton-like, MCMC, Population-based, …**	**Analytic**

Optimization methods

Finding a **weight-vector** w, minimizing the model error, in **many practical cases**

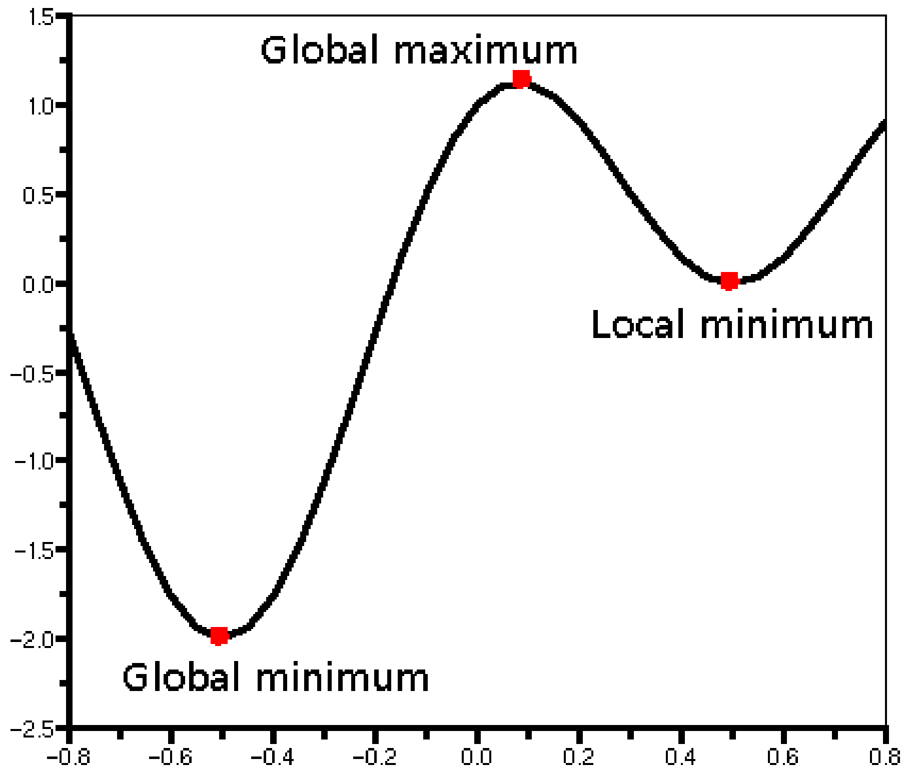
	Knowledge about **f**	
	Not much	**A lot**
	Combinatorial search: Brute-force, Stepwise, MCMC, Population-based, …	**Algorithmic**
Continuous	**Numeric methods:** Gradient-based, Newton-like, MCMC, Population-based, …	**Analytic**

Optimization methods

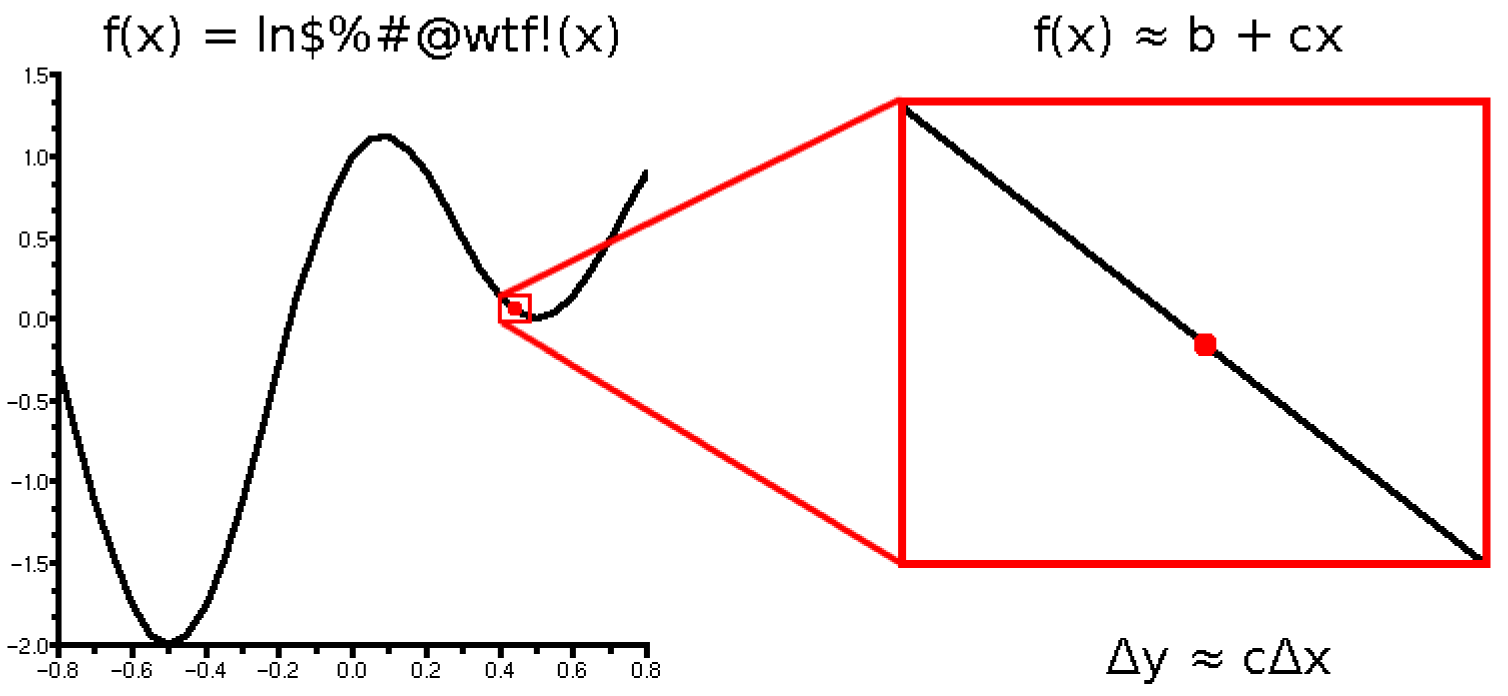
This lecture

		Knowledge about f	
		Not much	A lot
		Combinatorial search: Brute-force, Stepwise, MCMC, Population-based, …	Algorithmic
	Continuous	Numeric methods: Gradient-based, Newton-like, MCMC, Population-based, …	Analytic

Minima and maxima



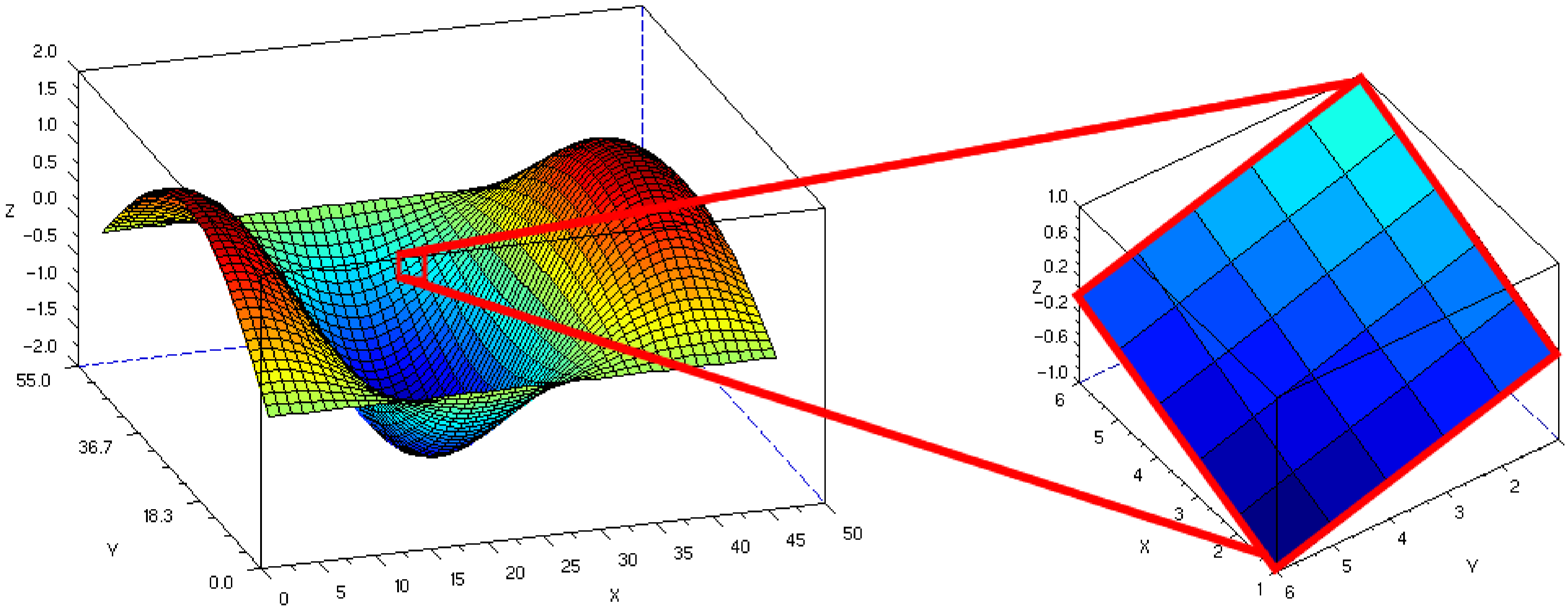
Differentiability



Differentiability

$$f(x_1,x_2) = \mathrm{wugaduga}(x_1,x_2)$$

$$f(x_1,x_2) \approx b + c_1x_1 + c_2x_2$$



$$\Delta y \approx c^T \Delta x$$

Differentiability

Definition. We call a function $f : \mathbb{R}^m \to \mathbb{R}$ differentiable at point $\mathbf{x}_0$ if there exists $\mathbf{c}(\mathbf{x}_0) \in \mathbb{R}^m$ such that:

$$\Delta f(\mathbf{x}_0) = f(\mathbf{x}_0 + \Delta\mathbf{x}) - f(\mathbf{x}_0) = \mathbf{c}(\mathbf{x}_0)^T \Delta\mathbf{x} + o(\Delta\mathbf{x})$$

We call $\mathbf{c}(\mathbf{x}_0)$ the gradient or derivative* of f (at point $\mathbf{x}_0$) and denote it by:

$$\frac{\partial f(\mathbf{x}_0)}{\partial \mathbf{x}} \quad \text{or} \quad f'(\mathbf{x}_0) \quad \text{or} \quad \nabla f(\mathbf{x}_0)$$

The Most Important Observation

Let f be differentiable and let $\nabla f(\mathbf{x}_0) = \mathbf{c} \neq \mathbf{0}$.
Take $\mathbf{\Delta x} = \ \ldots .$ Then:

$$f(\mathbf{x}_0 + \mathbf{\Delta x}) \approx f(\mathbf{x}_0) + \mathbf{c}^T \ldots < f(\mathbf{x}_0).$$

therefore $\mathbf{x}_0$ can't be a minimum of f.

The Most Important Observation

Let f be differentiable and let $\nabla f(\mathbf{x}_0) = \mathbf{c} \neq \mathbf{0}$.
Take $\mathbf{\Delta x} = -\mu\mathbf{c}$. Then:

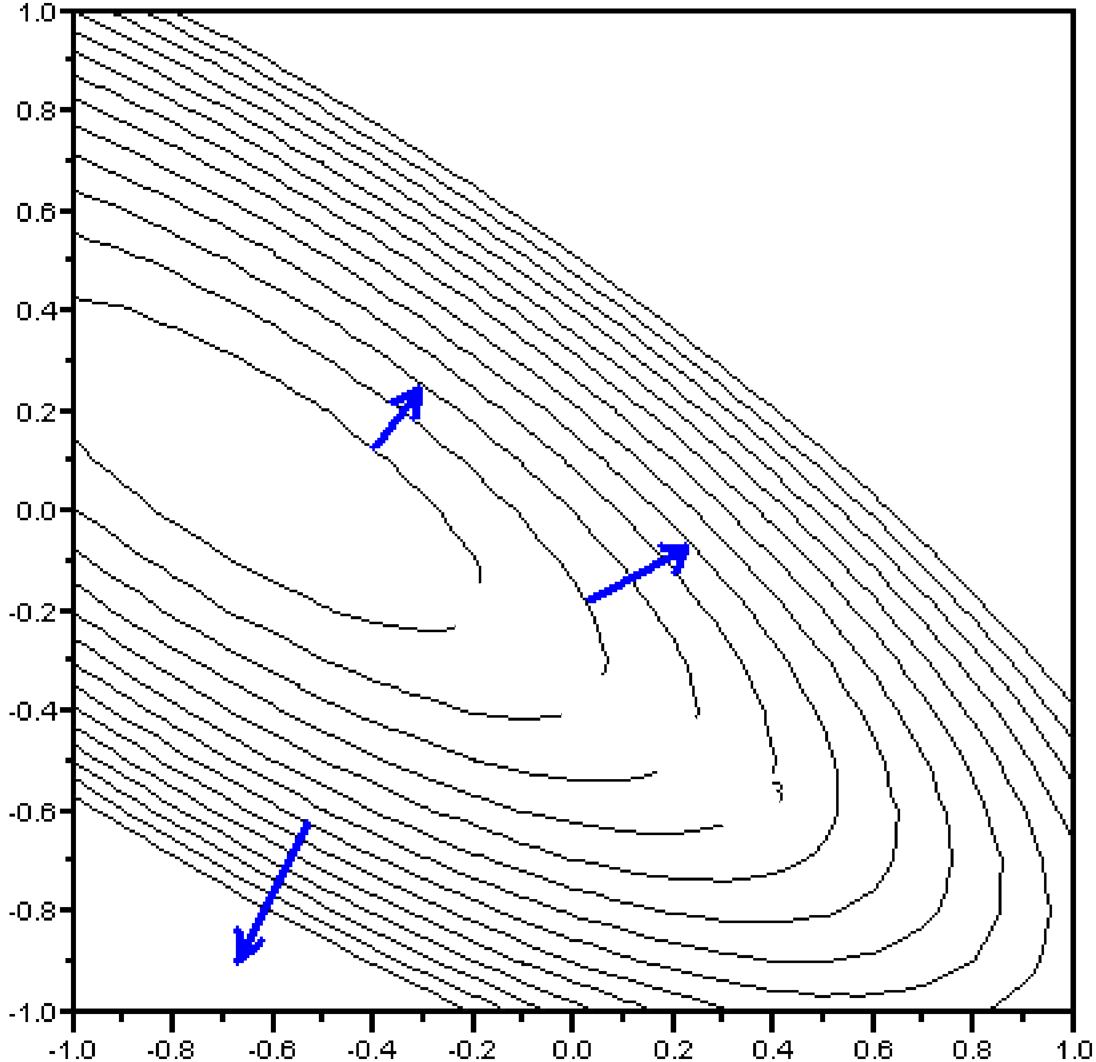
$$f(\mathbf{x}_0 + \mathbf{\Delta x}) \approx f(\mathbf{x}_0) + \mathbf{c}^T(-\mu\mathbf{c}) < f(\mathbf{x}_0).$$

therefore $\mathbf{x}_0$ can't be a minimum of f.

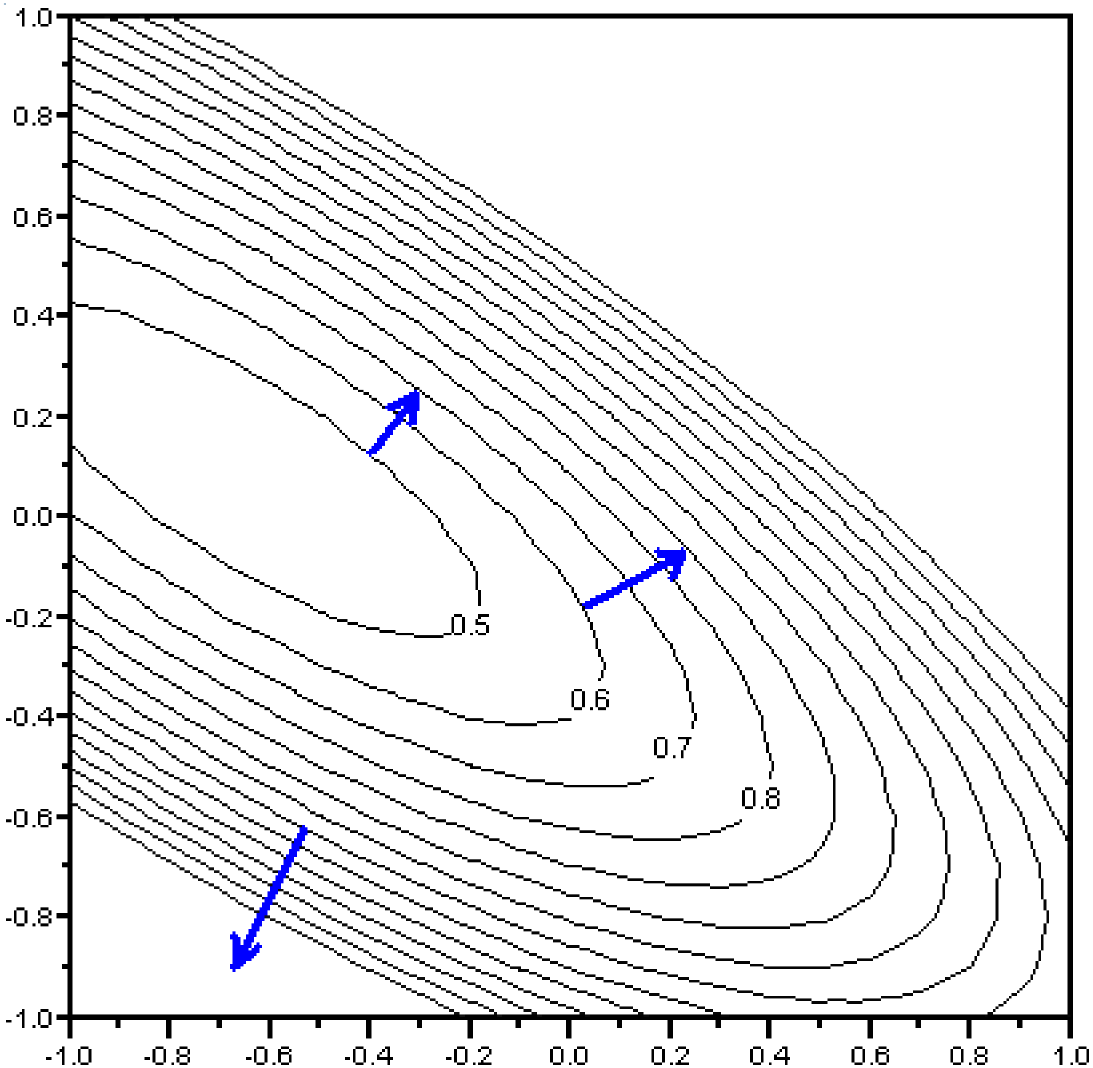
The Most Important Observation

- This small observation gives us everything we need for now
  - A nice **interpretation of the gradient**
  - An **extremality criterion**
  - An **iterative algorithm** for function minimization

Interpretation of the gradient



Interpretation of the gradient



Extremality criterion

Theorem (Fermat). Let f be differentiable. Then

$$\mathbf{x}_0 \text{ is an extremum } \Rightarrow \nabla f(\mathbf{x}_0) = \mathbf{0}.$$

The converse does not hold in general.

Gradient descent

1. Pick random point $\boldsymbol{x}_0$
2. If $\nabla f(\boldsymbol{x}_0) = \mathbf{0}$, then we've found an extremum.
3. Otherwise,

Gradient descent

1. Pick random point $\boldsymbol{x}_0$
2. If $\nabla f(\boldsymbol{x}_0) = \boldsymbol{0}$, then we've found an extremum.
3. Otherwise, make a small step downhill:

$$\boldsymbol{x}_1 \leftarrow \boldsymbol{x}_0 - \mu_0 \nabla f(\boldsymbol{x}_0)$$

Gradient descent

1. Pick random point $\boldsymbol{x}_0$
2. If $\nabla f(\boldsymbol{x}_0) = \mathbf{0}$, then we've found an extremum.
3. Otherwise, make a small step downhill:
$$\boldsymbol{x}_1 \leftarrow \boldsymbol{x}_0 - \mu_0 \nabla f(\boldsymbol{x}_0)$$
4. … and then another step
$$\boldsymbol{x}_2 \leftarrow \boldsymbol{x}_1 - \mu_1 \nabla f(\boldsymbol{x}_1)$$
5. … and so on until

Gradient descent

1. Pick random point $\boldsymbol{x}_0$
2. If $\nabla f(\boldsymbol{x}_0) = \mathbf{0}$, then we've found an extremum.
3. Otherwise, make a small step downhill:
$$\boldsymbol{x}_1 \leftarrow \boldsymbol{x}_0 - \mu_0 \nabla f(\boldsymbol{x}_0)$$
4. … and then another step
$$\boldsymbol{x}_2 \leftarrow \boldsymbol{x}_1 - \mu_1 \nabla f(\boldsymbol{x}_1)$$
5. … and so on until $\nabla f(\boldsymbol{x}_n) \approx \mathbf{0}$ or we're tired.

With a smart choice of μ_i we'll converge to a minimum

Gradient descent

1.
2.
3.

$$\boldsymbol{x}_1 \leftarrow \boldsymbol{x}_0 - \mu_0 \nabla f(\boldsymbol{x}_0)$$

4.

$$\boldsymbol{x}_2 \leftarrow \boldsymbol{x}_1 - \mu_1 \nabla f(\boldsymbol{x}_1)$$

Gradient descent

$$\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i - \mu_i \nabla f(\boldsymbol{x}_i)$$

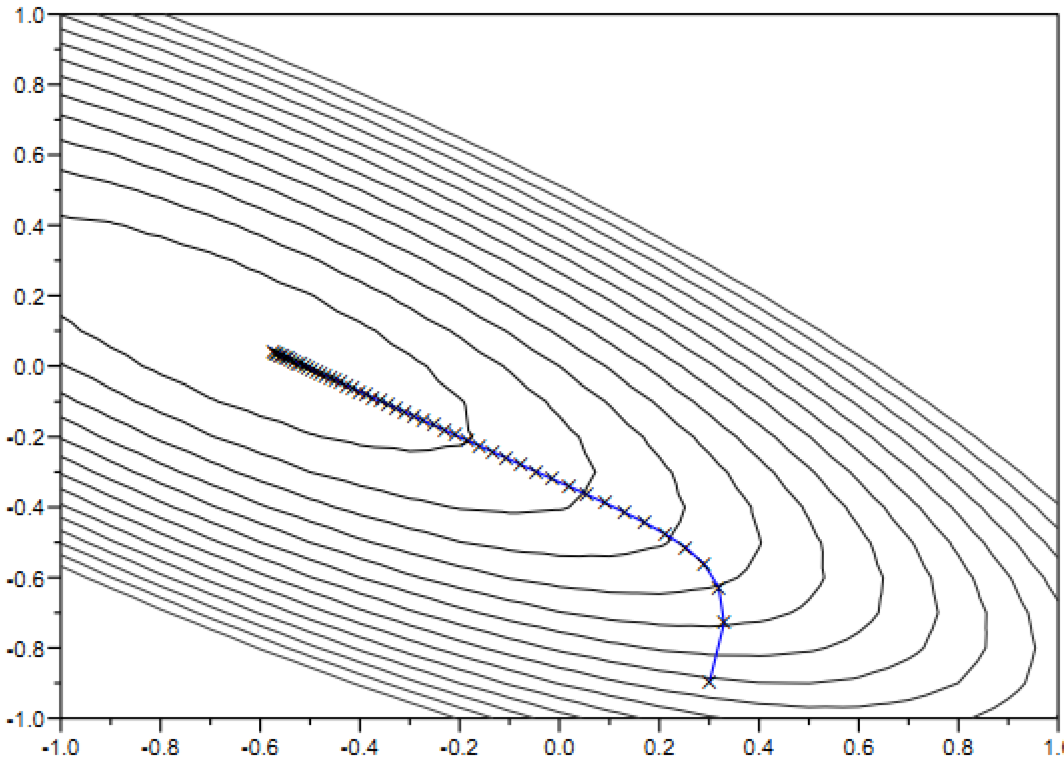
Gradient descent

$$\Delta \boldsymbol{x}_i = -\mu_i \nabla f(\boldsymbol{x}_i)$$

Gradient descent

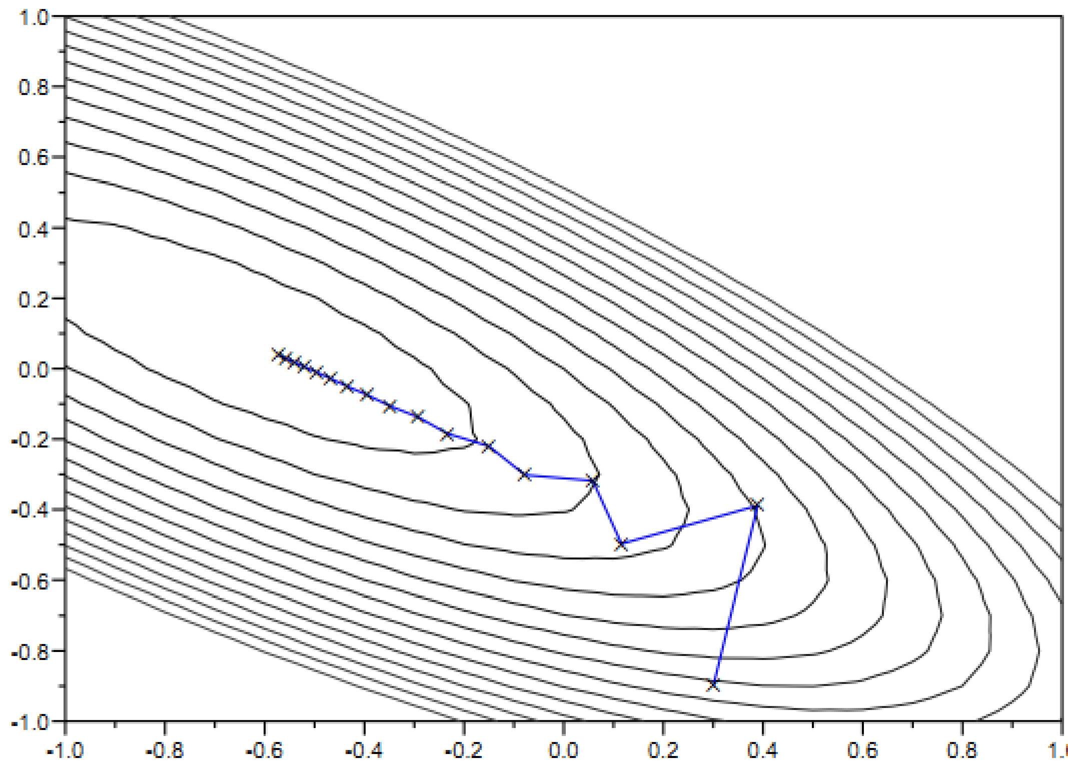
$$\Delta \boldsymbol{x}_i = -\mu \boldsymbol{c}$$

Gradient descent (fixed step)



$$\Delta \boldsymbol{x}_i = -\mu \nabla f(\boldsymbol{x}_i)$$

Gradient descent (fixed step)



$$\Delta \boldsymbol{x}_i = -\mu \nabla f(\boldsymbol{x}_i)$$

What do you need to know about optimization?

1.
2.

What do you need to know about optimization?

1. Optimization is **important**
2. Optimization is **possible***

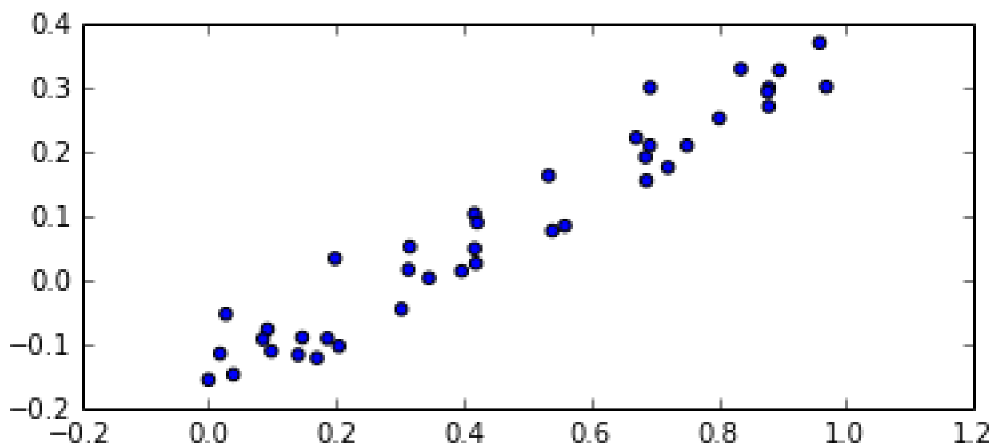
* Basic **techniques**

- Constrained / Unconstrained
- Analytic / Iterative
- Continuous / Discrete

Example: Linear Regression

- Suppose we are given a set of points

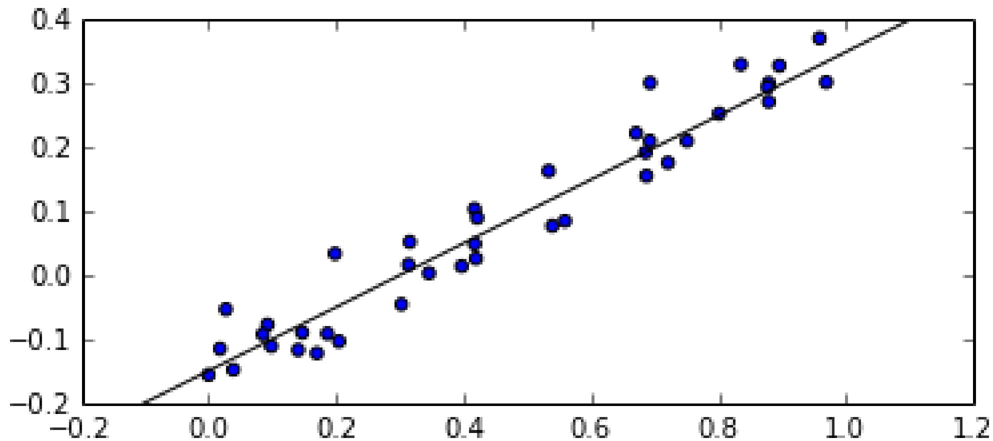
$$D = \{(x_1, y_1), (x_2, y_2), \ldots\}$$



Example: Linear Regression

- Let us find a way to predict y_i from x_i, using the following model:

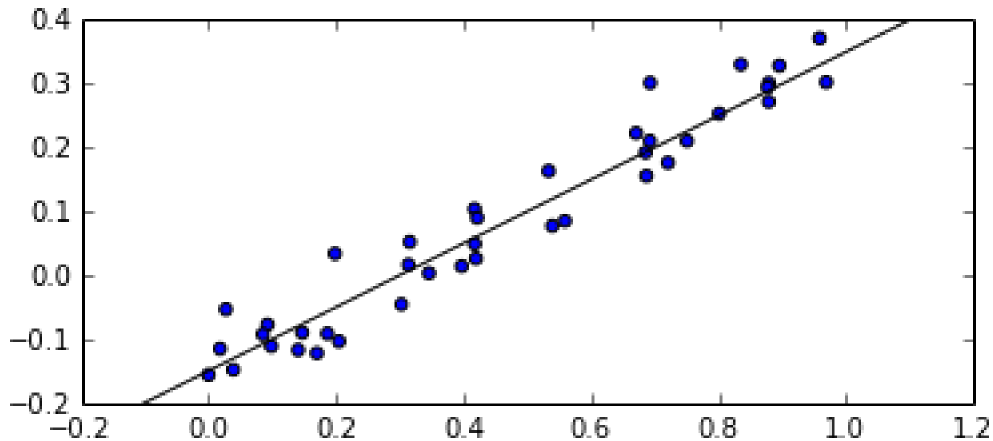
$$\hat{y}_i = w_0 + w_1 x_i$$



Example: Linear Regression

- Define *prediction error* of the model for point i:

$$e_i \coloneqq (\hat{y}_i - y_i)^2$$



Example: Linear Regression

- The error over all training samples is therefore:

$$E := \sum_i (\hat{y}_i - y_i)^2$$

Example: Linear Regression

- The error over all training samples is therefore:

$$E(w_0, w_1) \coloneqq \sum_i (\hat{y}_i - y_i)^2$$

Example: Linear Regression

- The error over all training samples is therefore:

$$E(w_0, w_1) \coloneqq \sum_i (\hat{y}_i - y_i)^2$$

$$\hat{y}_i = w_0 + w_1 x_i$$

Example: Linear Regression

- The error over all training samples is therefore:

$$E(w_0, w_1) := \sum_i (w_0 + w_1 x_i - y_i)^2$$

Example: Linear Regression

- The error over all training samples is therefore:

$$E(w_0, w_1) \coloneqq \sum_i (w_0 + w_1 x_i - y_i)^2$$

- Let us find parameter values w_0, w_1 by minimizing this error function.

Example: Linear Regression

- The error over all training samples is therefore:

$$E(w_0, w_1) := \sum_i (w_0 + w_1 x_i - y_i)^2$$

- Let us find parameter values w_0, w_1 by minimizing this error function.

NB: The error function is simply $-\log P[Data|Model]$, I.e. we are using **maximum likelihood estimation** here.

Example: Linear Regression

- The error over all training samples is therefore:

$$E(w_0, w_1) := \sum_i (w_0 + w_1 x_i - y_i)^2$$

- We shall derive a gradient descent based optimization algorithm.

Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - $\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - & & ? \\ w_1 - & & ? \end{matrix}$
- Until convergence

Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - $$\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - \mu \nabla_{w_0} E\left(w_0, w_1\right) \\ w_1 - \mu \nabla_{w_1} E(w_0, w_1) \end{matrix}$$
- Until convergence

Example: Linear Regression

- $\nabla_{w_0} E\,(w_0, w_1) = \nabla_{w_0}\left(\sum_i (w_0 + w_1 x_i - y_i)^2\right)$

Example: Linear Regression

- $\nabla_{w_0} E\,(w_0, w_1) = \nabla_{w_0}(\sum_i (w_0 + w_1 x_i - y_i)^2)$

$$= \sum_i 2(w_0 + w_1 x_i - y_i) = 2 \sum_i e_i$$

Example: Linear Regression

- $\nabla_{w_0} E\,(w_0, w_1) = \nabla_{w_0}\left(\sum_i (w_0 + w_1 x_i - y_i)^2\right)$
$$= \sum_i 2(w_0 + w_1 x_i - y_i) = 2\sum_i e_i$$
- $\nabla_{w_1} E\,(w_0, w_1) = \nabla_{w_1}\left(\sum_i (w_0 + w_1 x_i - y_i)^2\right)$

Example: Linear Regression

- $\nabla_{w_0} E\,(w_0, w_1) = \nabla_{w_0}(\sum_i (w_0 + w_1 x_i - y_i)^2)$
$$= \sum_i 2(w_0 + w_1 x_i - y_i) = 2\sum_i e_i$$
- $\nabla_{w_1} E\,(w_0, w_1) = \nabla_{w_1}(\sum_i (w_0 + w_1 x_i - y_i)^2)$
$$= \sum_i 2(w_0 + w_1 x_i - y_i)x_i = 2\sum_i e_i x_i$$

Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - $$\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - \mu \nabla_{w_0} E\left(w_0, w_1\right) \\ w_1 - \mu \nabla_{w_1} E(w_0, w_1) \end{matrix}$$
- Until convergence

Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - $$\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - \mu 2 \sum_i e_i \\ w_1 - \mu 2 \sum_i e_i x_i \end{matrix}$$
- Until convergence

Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - $\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - \mu \sum_i e_i \\ w_1 - \mu \sum_i e_i x_i \end{matrix}$
- Until convergence

Stochastic gradient descent

- Whenever the function to be minimized is a sum over samples coming from some distribution

$$\boldsymbol{f}(\boldsymbol{w}) = \sum \boldsymbol{g}(\boldsymbol{w}, \boldsymbol{x}_{\boldsymbol{k}})$$

the gradient is also a sum:

$$\nabla \boldsymbol{f}(\boldsymbol{w}) = \sum \nabla \boldsymbol{g}(\boldsymbol{w}, \boldsymbol{x}_{\boldsymbol{k}})$$

Stochastic gradient descent

- The step of the gradient descent algorithm is then:

$$\Delta \boldsymbol{w_i} = -\boldsymbol{\mu} \sum \nabla \boldsymbol{g}(\boldsymbol{w_i}, \boldsymbol{x_k})$$

- It is referred to as the "batch" update. It turns out, the minimization can also be performed by sampling a single random element from the sum on each step (the "on-line" update).

$$\Delta \boldsymbol{w_i} = -\boldsymbol{\mu} \nabla \boldsymbol{g}(\boldsymbol{w_i}, \boldsymbol{x_{random}})$$

Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - $$\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - \mu \sum_i e_i \\ w_1 - \mu \sum_i e_i x_i \end{matrix}$$
- Until convergence

Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - Pick a random training sample i
  - $\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - \mu e_i \\ w_1 - \mu e_i x_i \end{matrix}$
- Until convergence

Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - Pick a random training sample i
  - $\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - \mu e_i \\ w_1 - \mu e_i x_i \end{matrix}$
- Until convergence

Widrow & Hoff, "ADALINE"

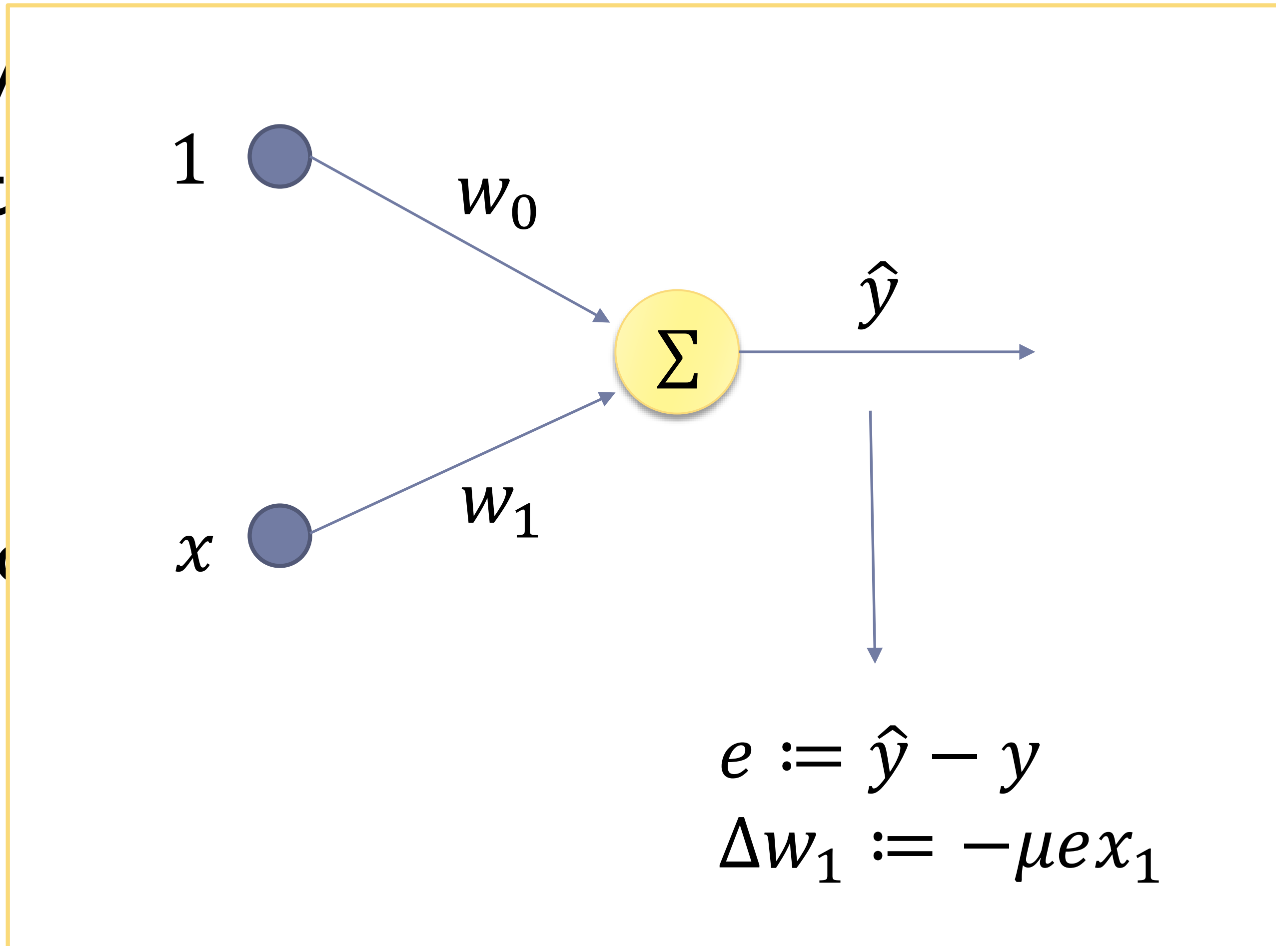
Example: Linear Regression

- Start with $w_0 = 0, w_1 = 0$
- Repeat:
  - Pick a random training sample i
  - $$\begin{matrix} w_0 \\ w_1 \end{matrix} := \begin{matrix} w_0 - \mu e_i \\ w_1 - \mu e_i x_i \end{matrix}$$
- Until convergence

Widrow & Hoff, "ADALINE", 1960

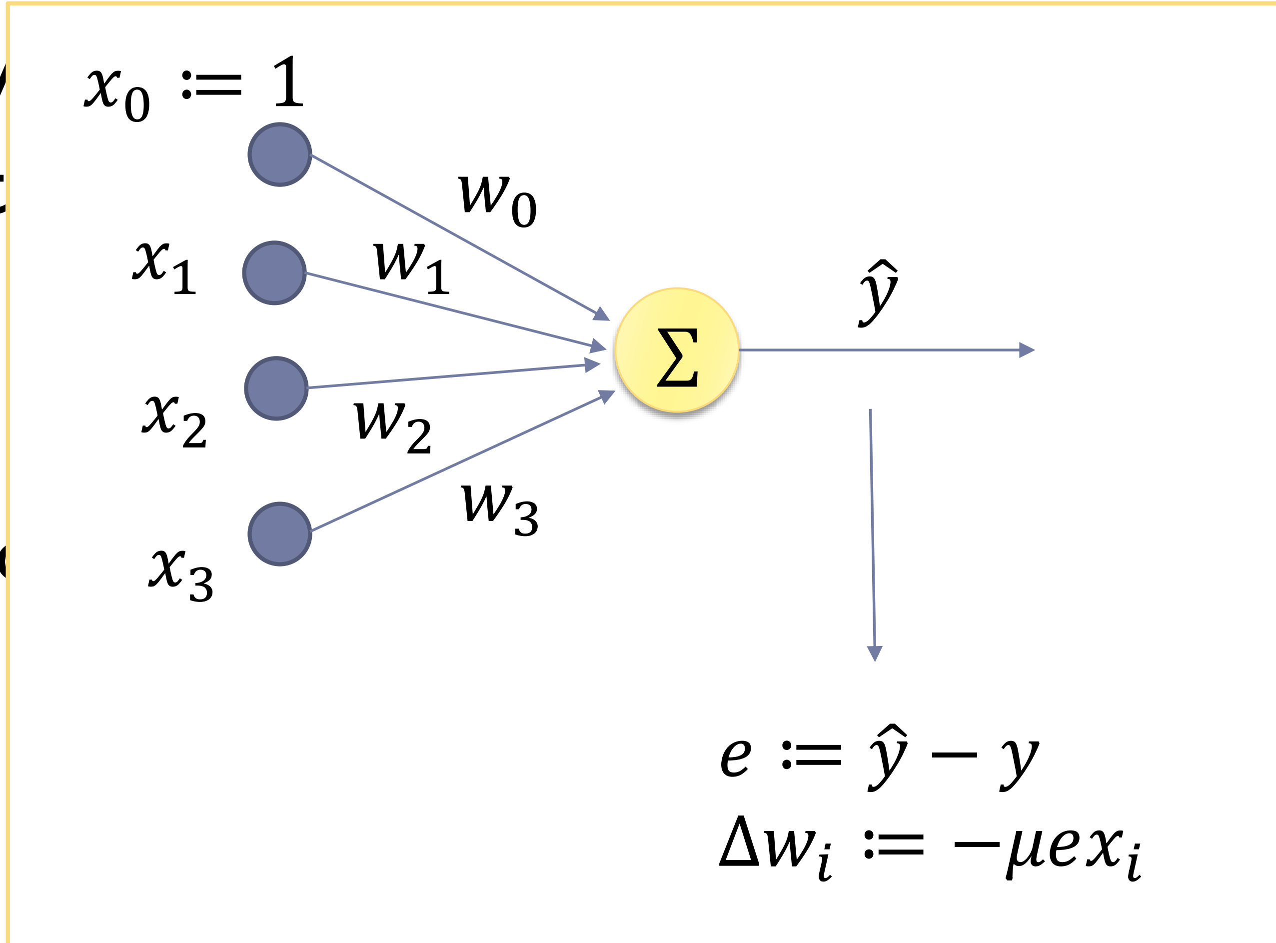
Example: Linear Regression

- Start w
- Repeat
  - Pick a
  - w_0 w_1 :=
- Until c



Example: Linear Regression

- Start w
- Repeat
  - Pick a
  - w_0 w_1 :=
- Until c



SKLearn & SGD Regression

```
from sklearn.linear_model import SGDRegressor

model = SGDRegressor(alpha=0, n_iter=30)
model.fit(X, y)

w0 = model.intercept_
w = model.coef_

model.predict(X_new)
```

Linear regression analytically

```
from sklearn.linear_model import
                                   LinearRegression

model = LinearRegression()
model.fit(X, y)
```

Polynomial Regression

Say we'd like to fit a model:

$$f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2^2 + w_3 x_1 x_2$$

Polynomial Regression

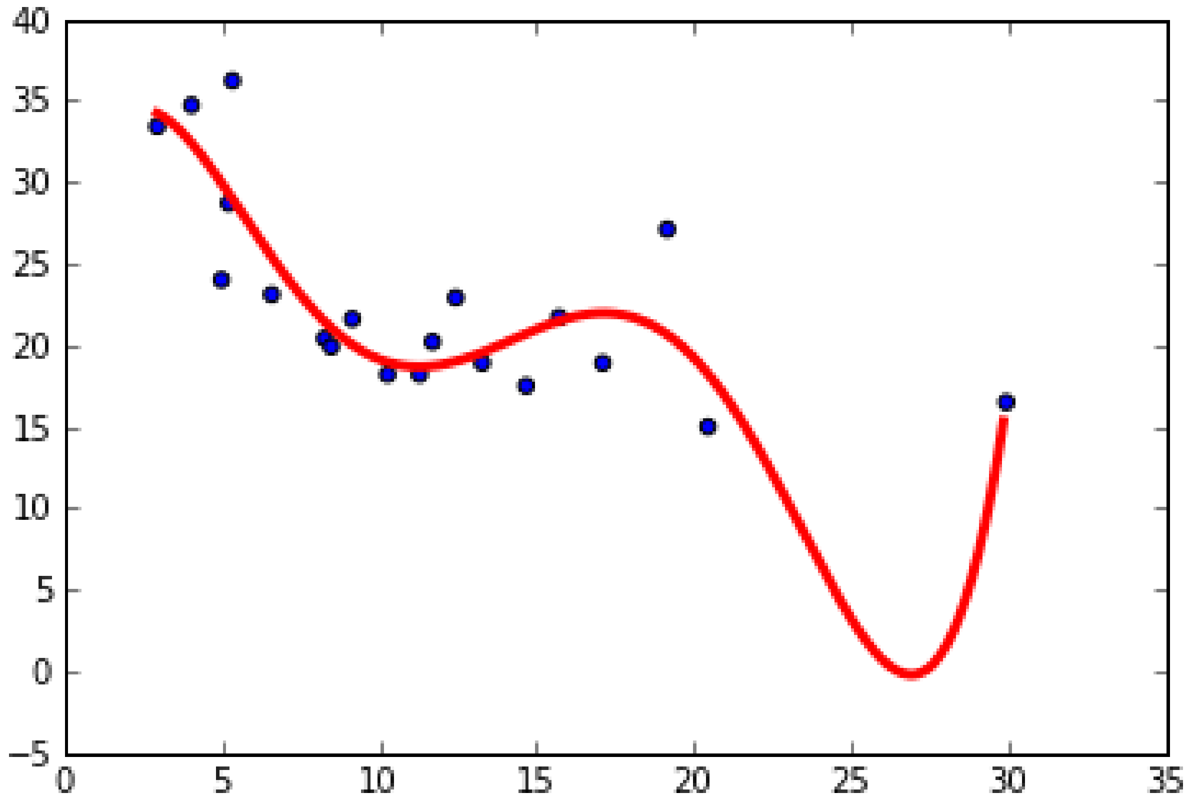
Say we'd like to fit a model:

$$f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2^2 + w_3 x_1 x_2$$

Simply transform the features and proceed as normal:

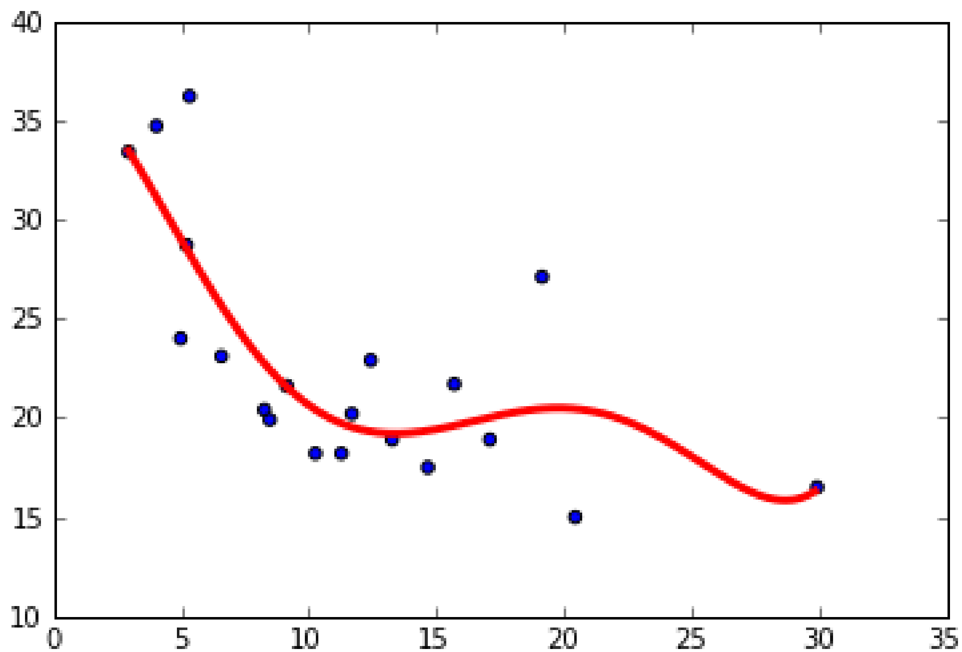
$$(x_1, x_2) \to (x_1, x_2^2, x_1 x_2)$$

Overfitting



Regularization

$$E(\boldsymbol{w}) := \frac{1}{2}\sum_i e_i^2 + \lambda \sum_{j=1}^{m} w_j^2$$



Regularization

$$E(\boldsymbol{w}) := \frac{1}{2}\sum_i e_i^2 + \lambda \sum_{j=1}^{m} w_j^2$$

ℓ_2-loss

ℓ_2-penalty

Regularization

$$E(\boldsymbol{w}) := \frac{1}{2}\sum_{i} e_i^2 + \lambda \sum_{j=1}^{m} w_j^2$$

ℓ_2-loss

ℓ_2-penalty

"Ridge Regression"

Regularization

$$E(\boldsymbol{w}) := \frac{1}{2}\sum_i e_i^2 + \lambda \sum_{j=1}^{m} |w_j|$$

ℓ_2-loss

ℓ_1-penalty

Regularization

$$E(\boldsymbol{w}) := \frac{1}{2}\sum_{i} |e_i| + \lambda \sum_{j=1}^{m} |w_j|$$

ℓ_1-loss

ℓ_1-penalty

Regularization

$$E(\boldsymbol{w}) := \frac{1}{2}\sum_i |e_i| + \lambda \sum_{j=1}^{m} |w_j|$$

ℓ_1-loss

ℓ_1-penalty

Data likelihood

Model prior

Regularization

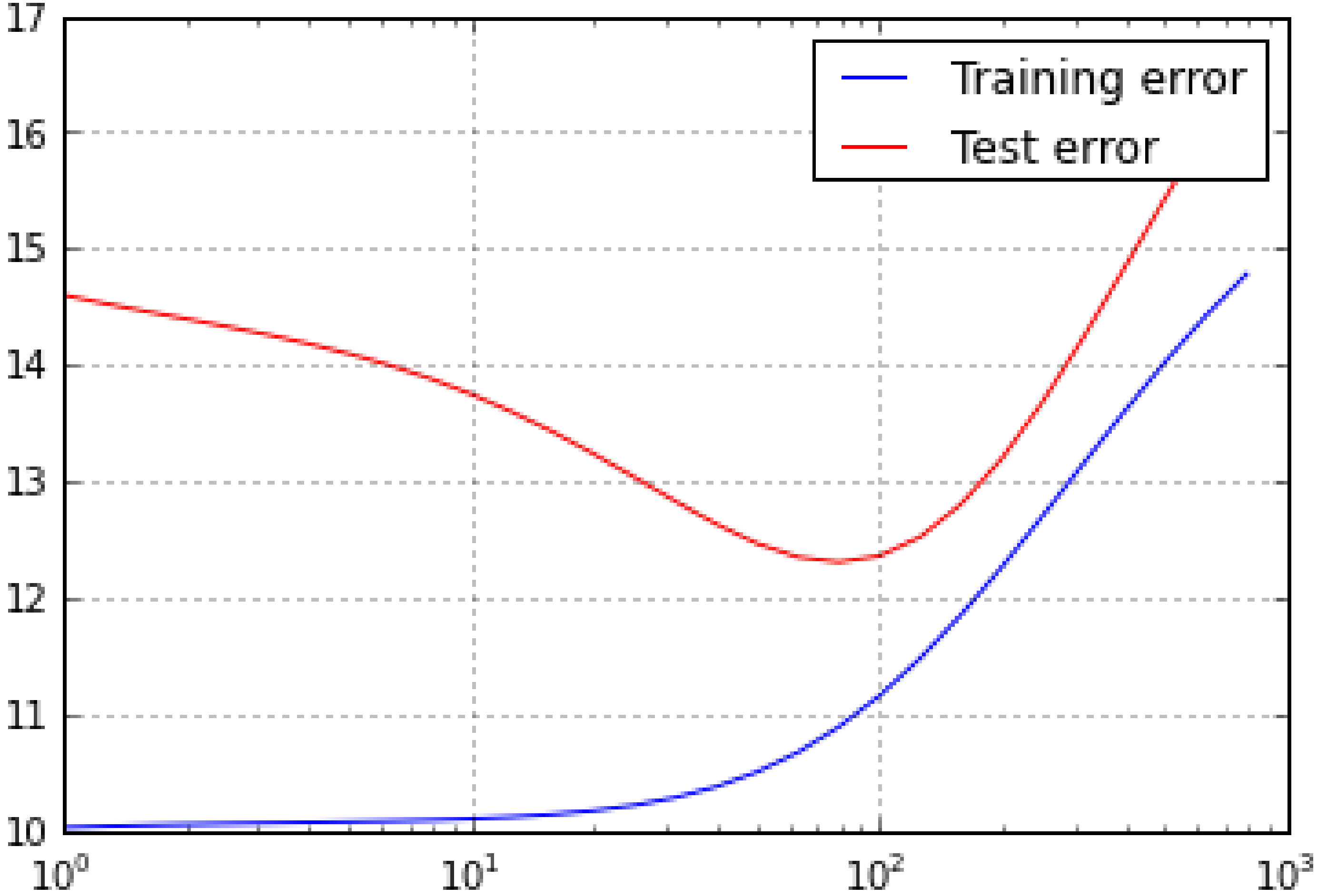
$$E(\boldsymbol{w}) := \frac{1}{2}\sum_i |e_i| + \lambda \sum_{j=1}^{m} |w_j|$$

```
>>> SGDRegressor?
Parameters
----------
loss : str, 'squared_loss' or 'huber' ...
...
penalty : str, 'l2' or 'l1' or 'elasticnet'
...
```

Exercise

Derive an SGD algorithm for Ridge Regression.

Effects of Regularization



Quiz

- Fermat' theorem says that ________________

- ADALINE update rule:

$$\boldsymbol{\Delta w} = \_\_\_\_\_\_\_$$

Quiz

- Large number of model parameters and/or small data may lead to _______________.

- We address overfitting by ______________.

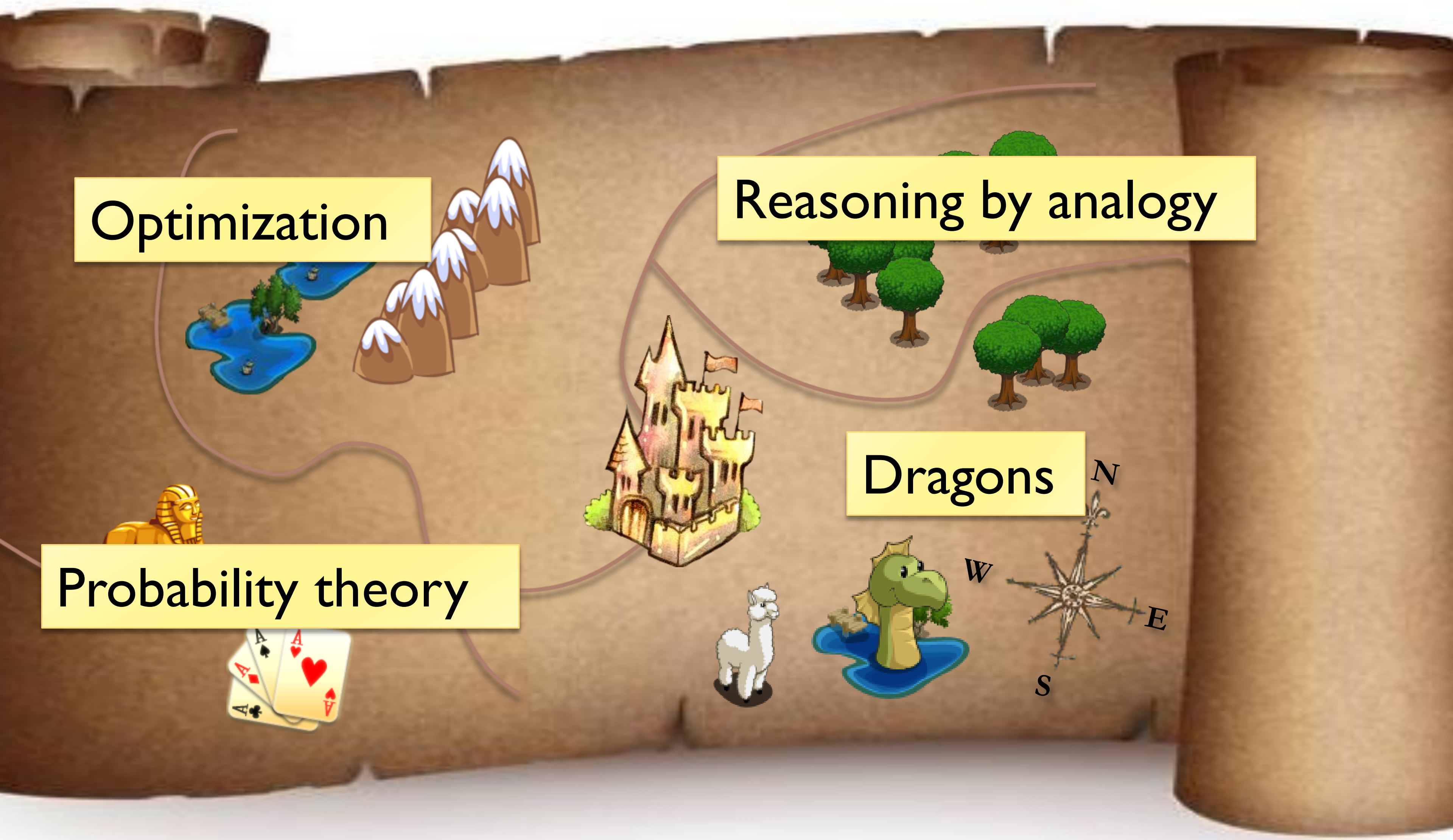
- "Ridge regression" means ___-loss and ____-penalty.

Quiz

- As we increase regularization strength (i.e. increase λ), the training error ____________.

- … and the test error ___________.

The Land of Machine Learning



Questions?