

# Data Science and Scientific Computation Track Core Course

Christoph Lampert



*Institute of Science and Technology*

Spring Semester 2015/16  
Segment 2, Lecture 1

Date		no.	Topic
Apr 11	Mon	1	predictive models, least squares regression, model selection, regularization
Apr 13	Wed	2	other regularizers, nonlinear regression missing data
Apr 18	Mon	3	robust regression, classification
Apr 20	Wed	4	handling large amounts of data
Apr 25	Mon	5	project Q&A
Apr 27	Wed	6	projects

**predictive models = model that makes predictions from data**

**predictive models = model that makes predictions from data**

**predictive models  $\neq$  model that provides insight into the data**

predictive models = model that makes predictions from data

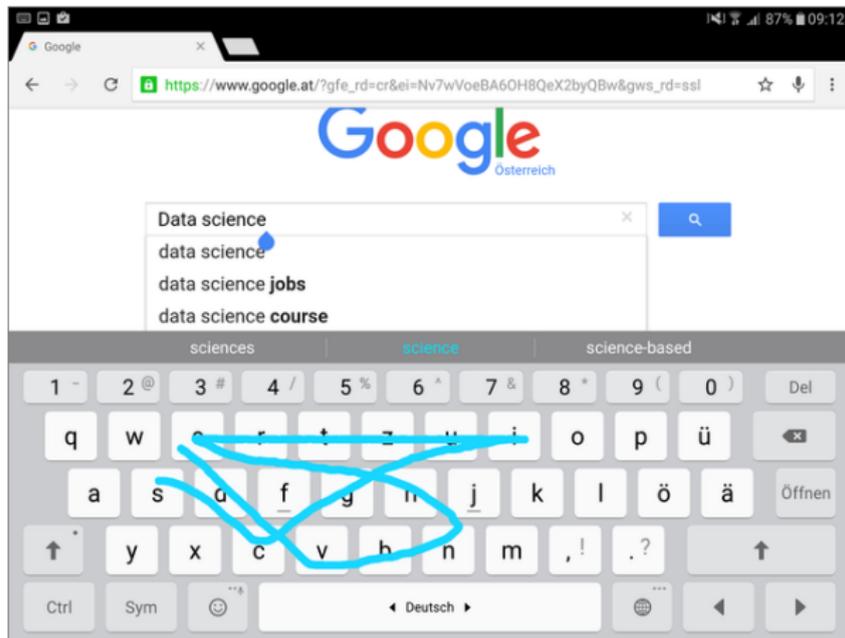
predictive models  $\neq$  model that provides insight into the data

The screenshot shows a web-based "Breast Cancer Risk Calculator" interface. The form includes several input fields: a checkbox for "History of breast cancer" (unchecked), a text input for "Age" (50), a dropdown for "Age at first menstrual period" (12 to 13), a dropdown for "Age at first live birth" (25 to 29), a dropdown for "Immediate family members with breast cancer" (One), a dropdown for "Ethnicity" (White/Caucasian), a dropdown for "Number of biopsies performed" (One), and a checkbox for "History of Atypical Hyperplasia" (checked). Below the form, a shaded box displays the results: "Five Year Risk: 4.3% (Average: 1.3%)" and "Lifetime Risk: 34.3% (Average: 11.2%)". There are "Add to Chart" and "Calculate" buttons.

Category	Value	Average
Five Year Risk	4.3%	1.3%
Lifetime Risk	34.3%	11.2%

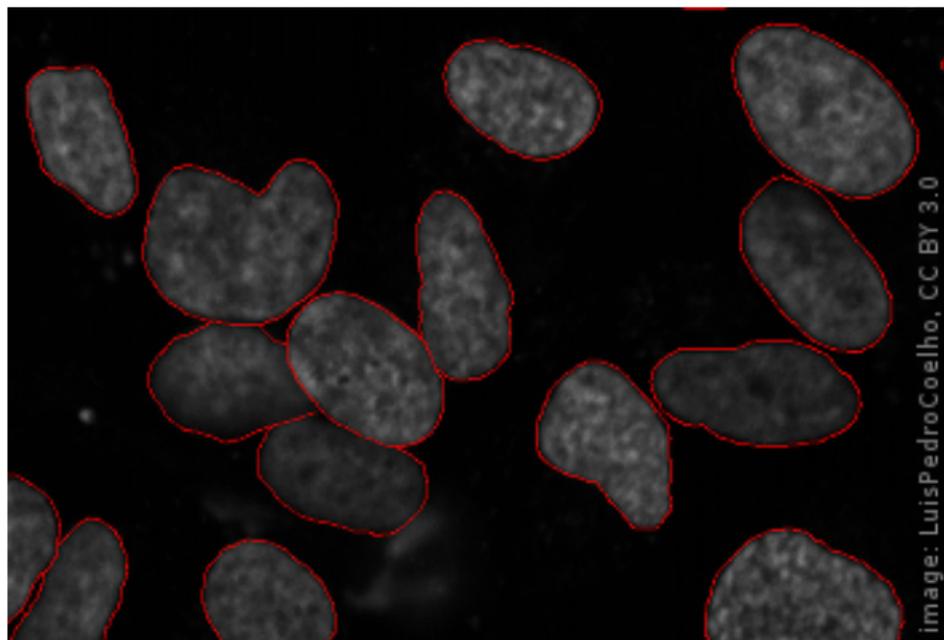
**predictive models = model that makes predictions from data**

**predictive models  $\neq$  model that provides insight into the data**



**predictive models = model that makes predictions from data**

**predictive models  $\neq$  model that provides insight into the data**



## Regression

- Data: anything (number, vector, image, natural text, web document, ...)
- Predicted quantity: a number, e.g. 5.3

## Classification

- Data: anything
- Predicted quantity: a decision, e.g. "yes"

## More complex/structured prediction tasks

- Data: anything
- Predicted quantity: a complex object, e.g. a natural language sentence, a segmentation mask of an image, a strategy for world domination, ...

# Regression

- Linear Regression
- Regularization
- Model Evaluation and Model Selection
- Nonlinear Regression
- Robust Regression

Given this data:

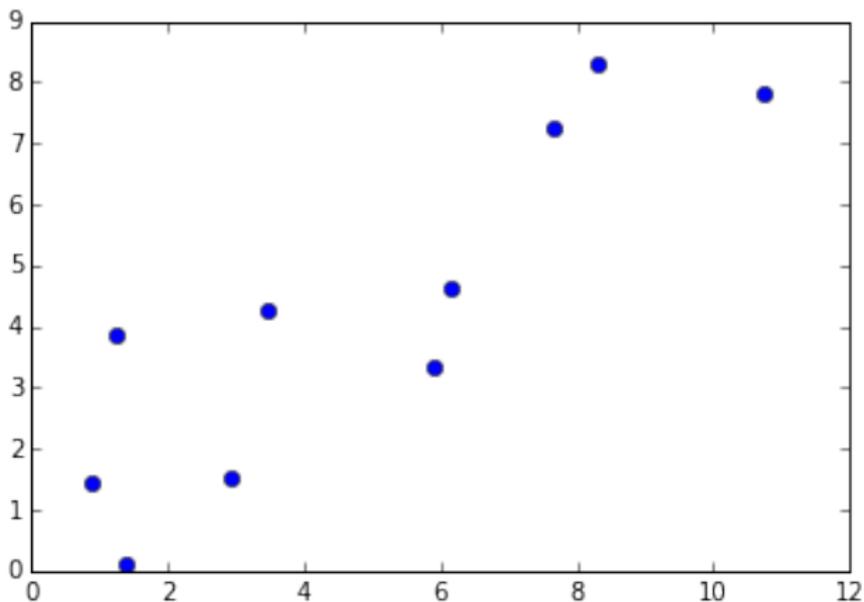
<b>value</b>	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
<b>outcome</b>	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10

How to fit a model that can predict future outcomes?

Given this data:

<b>value</b>	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
<b>outcome</b>	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10

How to fit a model that can predict future outcomes?



Looks more or less straight...

Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10

Let's fit a **linear model**:  $f(x) = ax$  for unknown  $a \in \mathbb{R}$ .

Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10

Let's fit a **linear model**:  $f(x) = ax$  for unknown  $a \in \mathbb{R}$ .

How to choose  $a$ ? **Least squares** criterion:

$$\min_{a \in \mathbb{R}} \sum_{i=1}^n (ax_i - y_i)^2$$

Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10

Let's fit a **linear model**:  $f(x) = ax$  for unknown  $a \in \mathbb{R}$ .

How to choose  $a$ ? **Least squares** criterion: 
$$\min_{a \in \mathbb{R}} \sum_{i=1}^n (ax_i - y_i)^2$$

To find minimum, compute derivative:

$$\frac{d}{da} \sum_{i=1}^n (ax_i - y_i)^2 = 2 \sum_i x_i (ax_i - y_i) = 2a \sum_i (x_i)^2 - 2 \sum_i x_i y_i$$

Set derivative to 0 and solve for  $a$ :

$$a = \frac{\sum_i x_i y_i}{\sum_i (x_i)^2} = \frac{281.56}{339.06} = 0.83 \quad \rightarrow \quad f(x) = 0.83x$$

Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10

What else could we have done?

Since we want  $f(x) \approx y$  for  $f(x) = ax$ . Since  $a = \frac{f(x)}{x}$ , how about

$$\min_a \sum_{i=1}^n \left( a - \frac{y_i}{x_i} \right)^2$$

Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10
$y_i/x_i$	0.57	1.00	0.52	1.64	0.73	0.75	0.95	1.23	3.12	0.07

What else could we have done?

Since we want  $f(x) \approx y$  for  $f(x) = ax$ . Since  $a = \frac{f(x)}{x}$ , how about

$$\min_a \sum_{i=1}^n \left(a - \frac{y_i}{x_i}\right)^2 \quad \rightarrow \quad a = \frac{1}{n} \sum_{i=1}^n \frac{y_i}{x_i} = 1.06$$

Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10
$y_i/x_i$	0.57	1.00	0.52	1.64	0.73	0.75	0.95	1.23	3.12	0.07

What else could we have done?

Since we want  $f(x) \approx y$  for  $f(x) = ax$ . Since  $a = \frac{f(x)}{x}$ , how about

$$\min_a \sum_{i=1}^n \left(a - \frac{y_i}{x_i}\right)^2 \quad \rightarrow \quad a = \frac{1}{n} \sum_{i=1}^n \frac{y_i}{x_i} = 1.06$$

Arithmetic mean of ratios? Maybe rather geometric mean:

$$a = \sqrt[n]{\prod_i \frac{y_i}{x_i}} = 0.77$$

Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10
$y_i/x_i$	0.57	1.00	0.52	1.64	0.73	0.75	0.95	1.23	3.12	0.07

What else could we have done?

Since we want  $f(x) \approx y$  for  $f(x) = ax$ . Since  $a = \frac{f(x)}{x}$ , how about

$$\min_a \sum_{i=1}^n \left(a - \frac{y_i}{x_i}\right)^2 \quad \rightarrow \quad a = \frac{1}{n} \sum_{i=1}^n \frac{y_i}{x_i} = 1.06$$

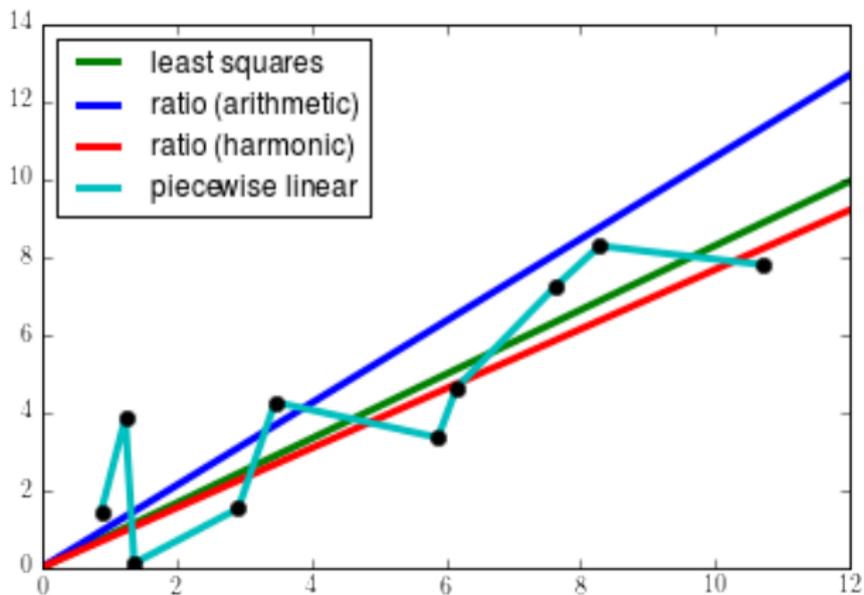
Arithmetic mean of ratios? Maybe rather geometric mean:

$$a = \sqrt[n]{\prod_i \frac{y_i}{x_i}} = 0.77$$

Something completely different: piecewise linear?

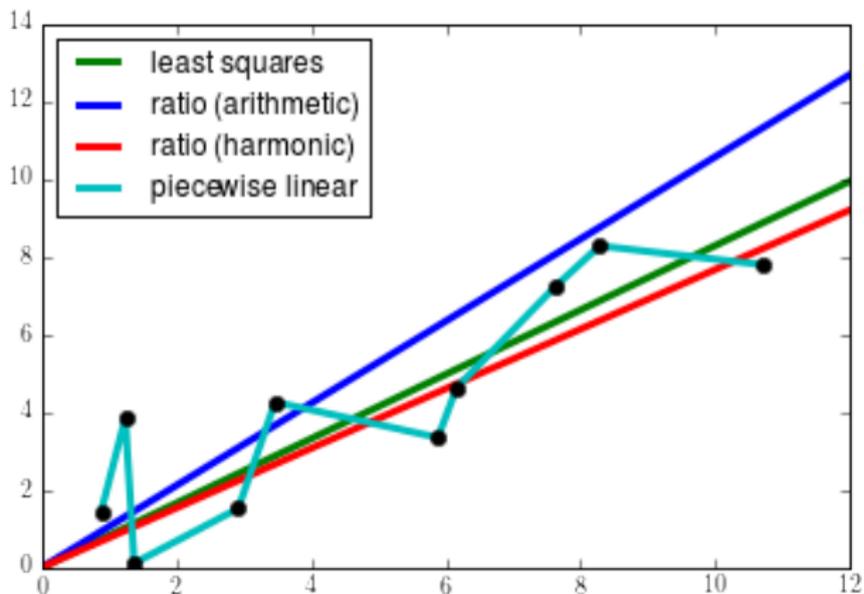
Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10



Data:

$x_i$	5.88	8.28	2.91	0.87	10.72	6.16	7.64	3.46	1.23	1.36
$y_i$	3.35	8.30	1.52	1.43	7.81	4.64	7.27	4.26	3.85	0.10



Visual inspection: least squares (green) might be best...



Given  $(x_1, y_1), \dots, (x_n, y_n)$  with  $x_i = (x_i^1, \dots, x_i^d) \in \mathbb{R}^d$  and  $y_i \in \mathbb{R}$ .

Linear model:  $f(x) = w^\top x$  for  $w^\top x = \sum_{j=1}^d w^j x^j$

Given  $(x_1, y_1), \dots, (x_n, y_n)$  with  $x_i = (x_i^1, \dots, x_i^d) \in \mathbb{R}^d$  and  $y_i \in \mathbb{R}$ .

Linear model:  $f(x) = w^\top x$  for  $w^\top x = \sum_{j=1}^d w^j x^j$

Least squares:  $\min_{w \in \mathbb{R}^d} \mathcal{L}(w)$  with  $\mathcal{L}(w) = \sum_i (w^\top x_i - y_i)^2$

Given  $(x_1, y_1), \dots, (x_n, y_n)$  with  $x_i = (x_i^1, \dots, x_i^d) \in \mathbb{R}^d$  and  $y_i \in \mathbb{R}$ .

Linear model:  $f(x) = w^\top x$  for  $w^\top x = \sum_{j=1}^d w^j x^j$

Least squares:  $\min_{w \in \mathbb{R}^d} \mathcal{L}(w)$  with  $\mathcal{L}(w) = \sum_i (w^\top x_i - y_i)^2$

$$\nabla_w \mathcal{L}(w) = 2 \sum_i x_i (x_i^\top w - y_i) = 2 \sum_i x_i x_i^\top w - 2 \sum_i x_i y_i$$

Setting the gradient to zero:

$$\underbrace{\sum_i x_i x_i^\top}_{\in \mathbb{R}^{d \times d}} w = \sum_i x_i y_i$$

We can solve for  $w$  if  $\sum_i x_i x_i^\top$  is full rank (at least:  $n \geq d$ ),

$$w = \left( \sum_i x_i x_i^\top \right)^{-1} \sum_i x_i y_i$$

In matrix notation:  $X = (x_1 | x_2 | \dots | x_n) \in \mathbb{R}^{d \times n}$ ,  $Y \in \mathbb{R}^n$ ,  $w \in \mathbb{R}^d$ .

Least squares:  $\min_{w \in \mathbb{R}^d} \mathcal{L}(w)$  with  $\mathcal{L}(w) = \|X^\top w - Y\|^2$

In matrix notation:  $X = (x_1|x_2|\dots|x_n) \in \mathbb{R}^{d \times n}$ ,  $Y \in \mathbb{R}^n$ ,  $w \in \mathbb{R}^d$ .

Least squares:  $\min_{w \in \mathbb{R}^d} \mathcal{L}(w)$  with  $\mathcal{L}(w) = \|X^\top w - Y\|^2$

$$\nabla_w \mathcal{L}(w) = \nabla_w (w^\top X X^\top w - Y^\top X^\top w - w^\top X Y + Y^\top Y) = 2X X^\top w - 2X Y$$

---

Very useful to memorize: matrix calculus

$$\nabla_x a = 0 \quad \nabla_x c^\top x = c \quad \nabla_x x^\top c = c \quad \nabla_x x^\top A x = (A^\top + A)x$$

In matrix notation:  $X = (x_1 | x_2 | \dots | x_n) \in \mathbb{R}^{d \times n}$ ,  $Y \in \mathbb{R}^n$ ,  $w \in \mathbb{R}^d$ .

Least squares:  $\min_{w \in \mathbb{R}^d} \mathcal{L}(w)$  with  $\mathcal{L}(w) = \|X^\top w - Y\|^2$

$$\nabla_w \mathcal{L}(w) = \nabla_w (w^\top X X^\top w - Y^\top X^\top w - w^\top X Y + Y^\top Y) = 2X X^\top w - 2X Y$$

Setting the gradient to zero:

$$X X^\top w = X Y$$

If  $X X^\top$  is full rank, we can solve for  $w$ :

$$w = (X X^\top)^{-1} X Y \quad \text{or} \quad \begin{array}{l} \text{lhs} = \text{dot}(X, X.T) \\ \text{rhs} = \text{dot}(X, Y) \\ w = \text{solve}(\text{lhs}, \text{rhs}) \end{array}$$

---

Very useful to memorize: matrix calculus

$$\nabla_x a = 0 \quad \nabla_x c^\top x = c \quad \nabla_x x^\top c = c \quad \nabla_x x^\top A x = (A^\top + A)x$$





## Linear model with bias term (= "intercept"):

$$f(x) = w^\top x + b \text{ for } w \in \mathbb{R}^d, b \in \mathbb{R}$$

Least squares:  $\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \mathcal{L}(w, b)$  with  $\mathcal{L}(w, b) = \sum_i (w^\top x_i + b - y_i)^2$

## Linear model with bias term (= "intercept"):

$$f(x) = w^\top x + b \text{ for } w \in \mathbb{R}^d, b \in \mathbb{R}$$

Least squares:  $\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \mathcal{L}(w, b)$  with  $\mathcal{L}(w, b) = \sum_i (w^\top x_i + b - y_i)^2$

$$0 = \nabla_b \mathcal{L}(w, b) = 2 \sum_i (x_i^\top w + b - y_i) \rightarrow b^{\text{opt}} = \bar{y} - \bar{x}^\top w$$

for  $\bar{x} = \frac{1}{n} \sum_i x_i$  and  $\bar{y} = \frac{1}{n} \sum_i y_i$ .

## Linear model with bias term (= "intercept"):

$$f(x) = w^\top x + b \text{ for } w \in \mathbb{R}^d, b \in \mathbb{R}$$

Least squares:  $\min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \mathcal{L}(w, b)$  with  $\mathcal{L}(w, b) = \sum_i (w^\top x_i + b - y_i)^2$

$$0 = \nabla_b \mathcal{L}(w, b) = 2 \sum_i (x_i^\top w + b - y_i) \rightarrow b^{\text{opt}} = \bar{y} - \bar{x}^\top w$$

for  $\bar{x} = \frac{1}{n} \sum_i x_i$  and  $\bar{y} = \frac{1}{n} \sum_i y_i$ .

$$\bar{\mathcal{L}}(w) = \bar{\mathcal{L}}(w, b^{\text{opt}}) = \sum_i \left( (x_i - \bar{x})^\top w - (y_i - \bar{y}) \right)^2$$

$$0 = \nabla_w \bar{\mathcal{L}}(w) = 2 \sum_i \left( (x_i - \bar{x})(x_i - \bar{x})^\top w - (y_i - \bar{y}) \right)$$

Solve for  $w$  (if possible):

$$\begin{aligned} w &= \underbrace{\left( \sum_i (x_i - \bar{x})(x_i - \bar{x})^\top \right)^{-1}}_{=n\text{Cov}(x_1, \dots, x_n)} \underbrace{\sum_i (x_i - \bar{x})(y_i - \bar{y})}_{=n\text{Cov}(x_1, \dots, x_n; y_1, \dots, y_n)} \\ &= \text{Cov}(X)^{-1} \text{Cov}(X, Y) \end{aligned}$$

### Observation: centered data

If the training data is centered, i.e.  $\frac{1}{n} \sum_i x_i = 0$ ,  $\frac{1}{n} \sum_i y_i = 0$ , we don't need a bias term.

### Alternative trick: feature augmentation

Adding a constant feature allows us to avoid models with explicit bias term:

- instead of  $x = (x^1, \dots, x^d) \in \mathbb{R}^d$ , use  $\tilde{x} = (x^1, \dots, x^d, 1) \in \mathbb{R}^{d+1}$
- for any  $\tilde{w} \in \mathbb{R}^{d+1}$ , think  $\tilde{w} = (w, b)$  with  $w \in \mathbb{R}^d$  and  $b \in \mathbb{R}$

Linear model in  $\mathbb{R}^{d+1}$ :

$$f(\tilde{x}) = \tilde{w}^\top \tilde{x} = \sum_{i=1}^{d+1} \tilde{w}_i \tilde{x}_i = \sum_{i=1}^d \tilde{w}_i \tilde{x}_i + \tilde{w}_{d+1} \tilde{x}_{d+1} = w^\top x + b$$

Linear model with bias term in  $\mathbb{R}^d \quad \equiv \quad$  linear model with no bias term in  $\mathbb{R}^{d+1}$

## Boston housing dataset

$x \in \mathbb{R}^{13}$													$y$
0.01	18.00	2.31	0.00	0.54	6.58	65.20	4.09	1.00	296.00	15.30	396.90	4.98	24.00
0.03	0.00	7.07	0.00	0.47	6.42	78.90	4.97	2.00	242.00	17.80	396.90	9.14	21.60
0.03	0.00	7.07	0.00	0.47	7.18	61.10	4.97	2.00	242.00	17.80	392.83	4.03	34.70
0.03	0.00	2.18	0.00	0.46	7.00	45.80	6.06	3.00	222.00	18.70	394.63	2.94	33.40
0.07	0.00	2.18	0.00	0.46	7.15	54.20	6.06	3.00	222.00	18.70	396.90	5.33	36.20
0.03	0.00	2.18	0.00	0.46	6.43	58.70	6.06	3.00	222.00	18.70	394.12	5.21	28.70
0.09	12.50	7.87	0.00	0.52	6.01	66.60	5.56	5.00	311.00	15.20	395.60	12.43	22.90
0.14	12.50	7.87	0.00	0.52	6.17	96.10	5.95	5.00	311.00	15.20	396.90	19.15	27.10
0.21	12.50	7.87	0.00	0.52	5.63	100.00	6.08	5.00	311.00	15.20	386.63	29.93	16.50
0.17	12.50	7.87	0.00	0.52	6.00	85.90	6.59	5.00	311.00	15.20	386.71	17.10	18.90
0.22	12.50	7.87	0.00	0.52	6.38	94.30	6.35	5.00	311.00	15.20	392.52	20.45	15.00
0.12	12.50	7.87	0.00	0.52	6.01	82.90	6.23	5.00	311.00	15.20	396.90	13.27	18.90
0.09	12.50	7.87	0.00	0.52	5.89	39.00	5.45	5.00	311.00	15.20	390.50	15.71	21.70
0.63	0.00	8.14	0.00	0.54	5.95	61.80	4.71	4.00	307.00	21.00	396.90	8.26	20.40
0.64	0.00	8.14	0.00	0.54	6.10	84.50	4.46	4.00	307.00	21.00	380.02	10.26	18.20
0.63	0.00	8.14	0.00	0.54	5.83	56.50	4.50	4.00	307.00	21.00	395.62	8.47	19.90
							⋮						
0.08	0.00	5.96	0.00	0.50	5.85	41.50	3.93	5.00	279.00	19.20	396.90	8.77	11.90

Learned linear least-square model:  $f(x) = w^\top x + b$

for  $w = (0.10, 0.01, 0.01, -1.02, 5.20, -4.41, 0.09, 0.15, 0.05, 0, -0.12, 0)$

$b = 30.15$



**So you've trained a model. How good it is?**

**You've trained multiple models. Which one to choose?**

**Is the model 'good enough'? What would be the best possible model?**

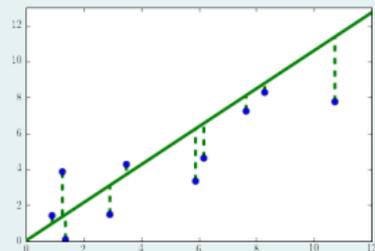
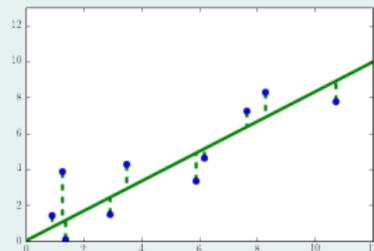
So you've trained a model. How good it is?

You've trained multiple models. Which one to choose?

Is the model 'good enough'? What would be the best possible model?

### Wrong approach to evaluate predictive models: explained variance

Evaluate model error by checking how well it fits the training data.



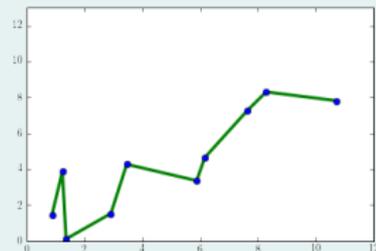
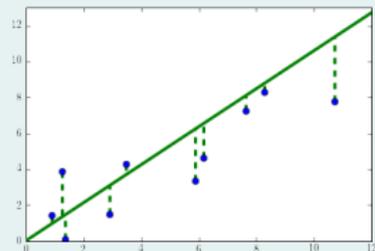
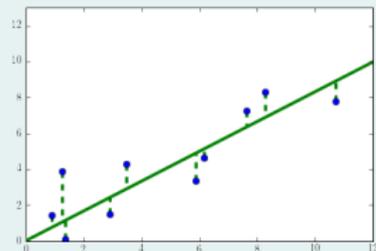
So you've trained a model. How good it is?

You've trained multiple models. Which one to choose?

Is the model 'good enough'? What would be the best possible model?

## Wrong approach to evaluate predictive models: explained variance

Evaluate model error by checking how well it fits the training data.



Explained variance prefers complex models over simple ones (always!)

## Right approach to model evaluation

The model is *not for you*, it's send out to be used by a *user*.

### You: train the model on your data

**input** training data  $\mathcal{D}_{trn}$

$f \leftarrow$  some procedure using  $\mathcal{D}_{trn}$

**output** predictive model  $f : \mathcal{X} \rightarrow \mathbb{R}$

### The user: make predictions on his/her own data

**input** trained predictive model  $f : \mathcal{X} \rightarrow \mathbb{R}$

**input** new data  $\mathcal{D}_{tst}$

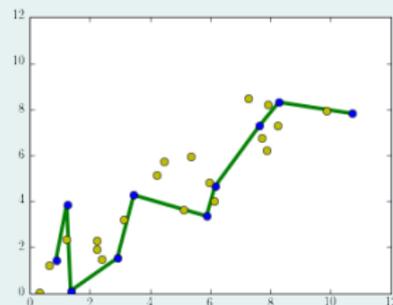
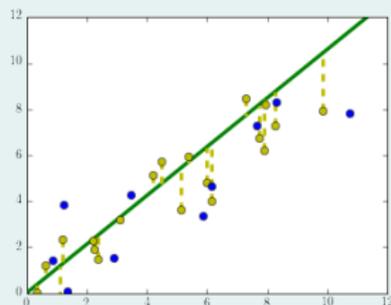
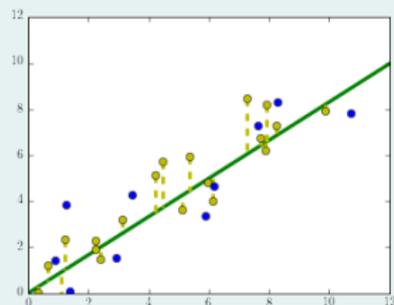
use  $f$  to make predictions on  $\mathcal{D}_{tst}$

**output** happy face or lawsuit

What matters isn't how good the model is on the training set, but on **new data**.

## Evaluate predictive models: generalization performance

Evaluate model error on data that has not been used for training.



For trained model  $f$  and new data  $\mathcal{D}_{tst} = \{(x'_1, y'_1), \dots, (x'_m, y'_m)\}$ ,

$$E(f) = \frac{1}{m} \sum_{i=1}^m (f(x'_i) - y'_i)^2$$

**Observation:** A model can be perfect on training data (blue), but still not do great on new data (yellow).

In practice, we don't get "new" data. We'll have to use the available data both for training and for evaluation.

### Model Training and Evaluation

**input** data  $\mathcal{D}$

**input** learning method  $A$

randomly split  $\mathcal{D} = \mathcal{D}_{trn} \dot{\cup} \mathcal{D}_{tst}$  disjointly

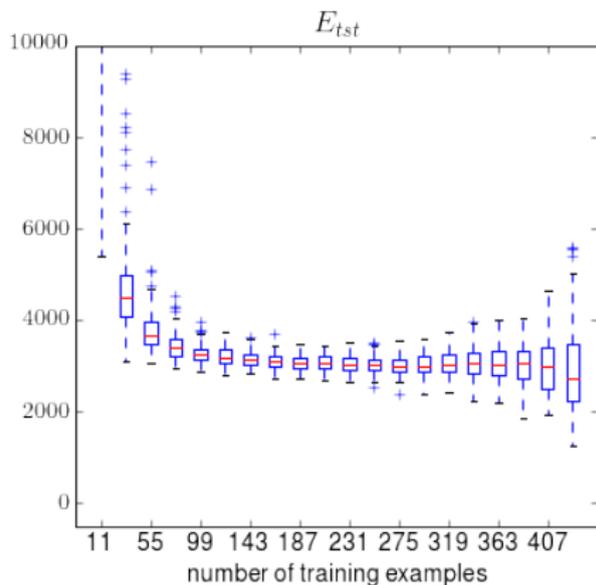
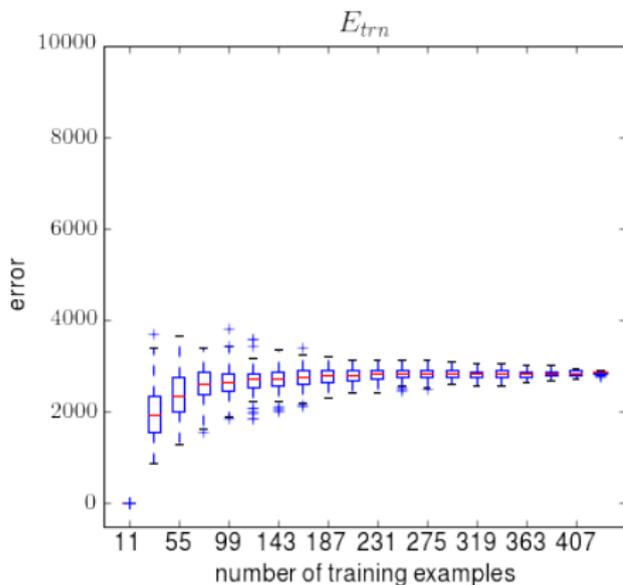
set aside  $\mathcal{D}_{tst}$  to a safe place // do not look at it!

$f \leftarrow A[\mathcal{D}_{trn}]$  // i.e. train model on training set

$E(f) \leftarrow$  performance of  $f$  on  $\mathcal{D}_{tst}$

**output** trained model  $f$ , performance estimate  $E(f)$

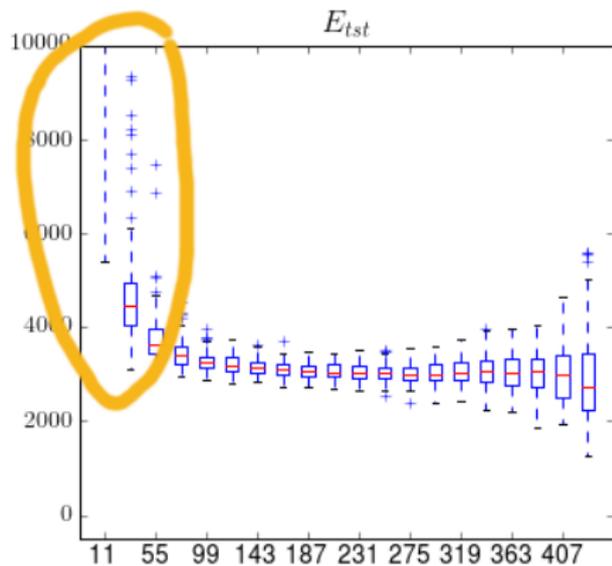
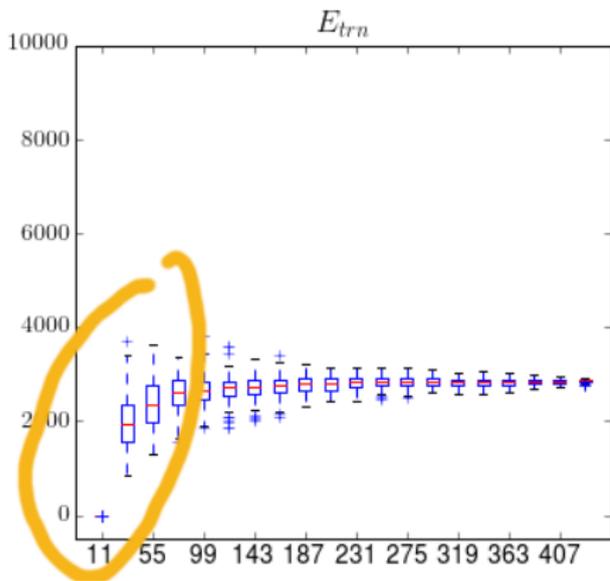
- Do not use  $\mathcal{D}_{tst}$  for anything except the very last step.
- Do not look at  $\mathcal{D}_{tst}$ ! Even if the learning algorithm doesn't see it, you looking at it can and will influence your model design or parameter selection (human overfitting).



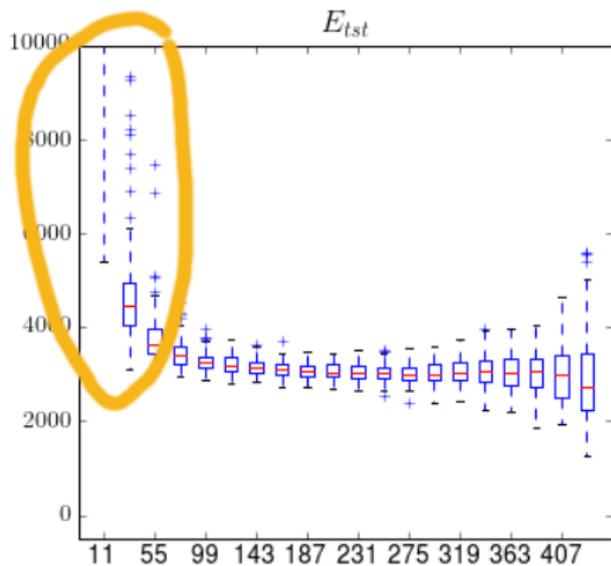
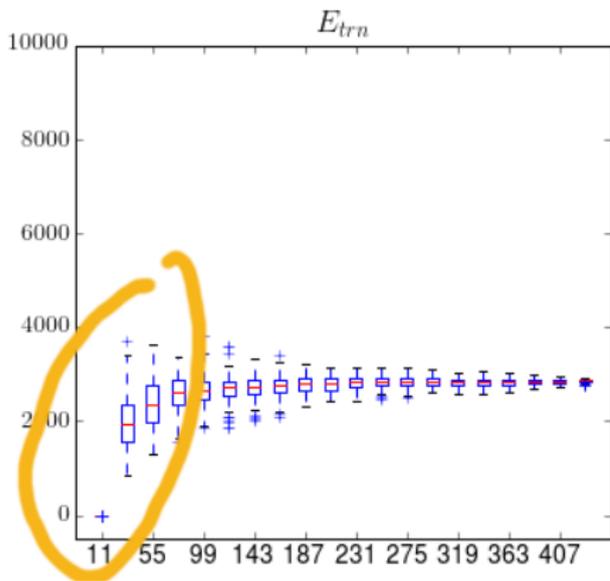
**Fact 1:** Predictive models tend to get better when trained on more data.

**Fact 2:** A too small test set makes the quality estimate unreliable.

**Guideline:**  $\mathcal{D}_{trn}$  should be as big as possible, but  $\mathcal{D}_{tst}$  must be large enough to be convincing.



**Fact 3:** With very little training data, the training error is very small, but the test error is very large.



**Fact 3:** With very little training data, the training error is very small, but the test error is very large.

**OVERFITTING!**

The model has learned to reproduce idiosyncracies/noise. On new data, this causes a large error.

**Observation:** Least squares regression overfits if the number of example is not much bigger than the number of data dimensions.

**How to avoid overfitting?**

**Observation:** Least squares regression overfits if the number of example is not much bigger than the number of data dimensions.

**How to avoid overfitting?**

### Feature Selection

**Idea:** reduce the dimensionality of the data by dropping some dimensions

**Problems:** 1) which ones? 2) simply throwing away data is rarely a good idea

**Observation:** Least squares regression overfits if the number of example is not much bigger than the number of data dimensions.

## How to avoid overfitting?

### Feature Selection

**Idea:** reduce the dimensionality of the data by dropping some dimensions

**Problems:** 1) which ones? 2) simply throwing away data is rarely a good idea

### Dimensionality Reduction

**Idea:** reduce data dimensionality differently, e.g. Principal Component Analysis.

**Problems:** 1) can destroy structure, 2) few dimensions might not be enough

**Observation:** Least squares regression overfits if the number of example is not much bigger than the number of data dimensions.

## How to avoid overfitting?

### Feature Selection

**Idea:** reduce the dimensionality of the data by dropping some dimensions

**Problems:** 1) which ones? 2) simply throwing away data is rarely a good idea

### Dimensionality Reduction

**Idea:** reduce data dimensionality differently, e.g. Principal Component Analysis.

**Problems:** 1) can destroy structure, 2) few dimensions might not be enough

### Regularization

**Idea:** prevent the model from overfitting by making it **robust**

**Problems:** almost none. This is actually a good idea.

## What do we mean by robustness?

### Robustness of model parameters

How robust are the linear least squares model parameters

$$w = (XX^T)^{-1}XY$$

against small changes in the training data  $X$ ?

### Robustness of predictions

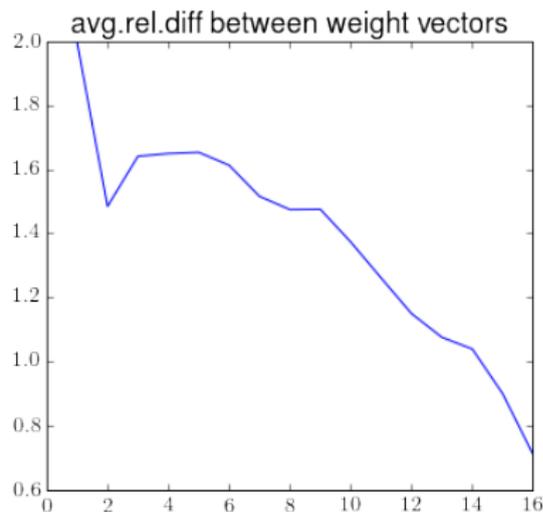
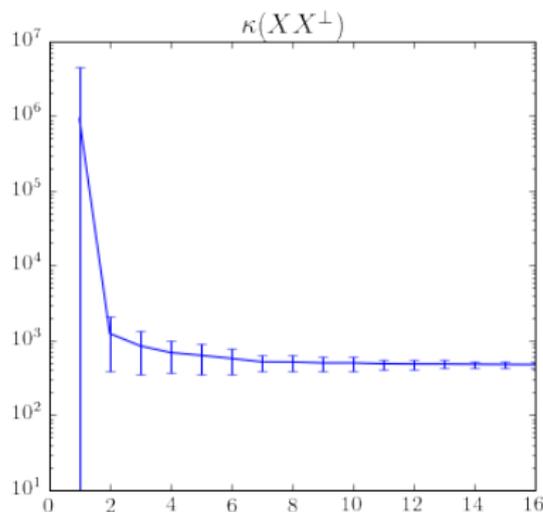
How robust are the predictions

$$f(x) = w^T x$$

against small changes of  $x$  ?

Robustness of the parameter vector  $w = (XX^T)^{-1}XY$  is determined by the **condition number** of the matrix  $XX^T$ :

$$\kappa = \frac{\sigma_{\max}(XX^T)}{\sigma_{\min}(XX^T)}.$$



- for real data,  $\kappa$  is often large unless  $n$  is much larger than  $d$   
→ randomness in  $X$  can have large impact on  $w$

For any linear model, the robustness of the predictions  $f(x) = w^\top x$  is determined by the **norm** of the weight vector  $\|w\|$ :

$$\frac{f(x + \epsilon) - f(x)}{\|\epsilon\|} = \frac{\langle w, x + \epsilon \rangle - \langle w, x \rangle}{\|\epsilon\|} = \frac{\langle w, \epsilon \rangle}{\|\epsilon\|} \leq \frac{\|w\| \|\epsilon\|}{\|\epsilon\|} = \|w\|$$

### Insight:

- if many different  $w$  work well on the training data, prefer the one with small  $\|w\|$
- maybe even: allows for higher  $E_{\text{trn}}$  to avoid models with very large  $\|w\|$

So far:

- learn model parameters by minimizing error on training set

Now:

- take *robustness* into account as well when learning parameters

### Regularized Least Squared Regression (= Ridge Regression)

For some  $\lambda \geq 0$  (=regularization parameter), solve

$$\min_{w \in \mathbb{R}^d} \sum_i (w^\top x_i - y_i)^2 + \lambda \|w\|^2$$

So far:

- learn model parameters by minimizing error on training set

Now:

- take *robustness* into account as well when learning parameters

## Regularized Least Squared Regression (= Ridge Regression)

For some  $\lambda \geq 0$  (=regularization parameter), solve

$$\min_{w \in \mathbb{R}^d} \underbrace{\sum_i (w^\top x_i - y_i)^2}_{\text{training error}} + \lambda \underbrace{\|w\|^2}_{\text{"regularizer"}}$$

# Regularization

So far:

- learn model parameters by minimizing error on training set

Now:

- take *robustness* into account as well when learning parameters

## Regularized Least Squared Regression (= Ridge Regression)

For some  $\lambda \geq 0$  (=regularization parameter), solve

$$\min_{w \in \mathbb{R}^d} \underbrace{\sum_i (w^\top x_i - y_i)^2}_{\text{training error}} + \lambda \underbrace{\|w\|^2}_{\text{"regularizer"}}$$

Observation:

- the bigger  $\lambda$ , the more emphasis we put on *robustness* versus *training error*
- $\lambda = 0$  → no regularization, original least squares regression
- $\lambda \rightarrow \infty$  → training error ignored ( $w \rightarrow 0$ ), but perfectly robust

Open question: what's the best value for  $\lambda$ ?

## Ridge Regression

Ridge regression is as easy to learn as least squares:

$$\min_{w \in \mathbb{R}^d} \mathcal{L}(w) + \lambda \Omega(w)$$

for  $\mathcal{L}(w) = \sum_i (w^\top x_i - y_i)^2$  and  $\Omega(w) = \|w\|^2$ .

## Ridge Regression

Ridge regression is as easy to learn as least squares:

$$\min_{w \in \mathbb{R}^d} \mathcal{L}(w) + \lambda \Omega(w)$$

for  $\mathcal{L}(w) = \sum_i (w^\top x_i - y_i)^2$  and  $\Omega(w) = \|w\|^2$ .

$$\nabla_w [\mathcal{L}(w) + \lambda \Omega(w)] = 2 \sum_i x_i x_i^\top w - 2 \sum_i x_i y_i + 2\lambda w$$

Ridge regression is as easy to learn as least squares:

$$\min_{w \in \mathbb{R}^d} \mathcal{L}(w) + \lambda \Omega(w)$$

for  $\mathcal{L}(w) = \sum_i (w^\top x_i - y_i)^2$  and  $\Omega(w) = \|w\|^2$ .

$$\nabla_w [\mathcal{L}(w) + \lambda \Omega(w)] = 2 \sum_i x_i x_i^\top w - 2 \sum_i x_i y_i + 2\lambda w$$

Set gradient to zero:

$$\sum_i x_i x_i^\top w + \lambda w = \sum_i x_i y_i$$

Ridge regression is as easy to learn as least squares:

$$\min_{w \in \mathbb{R}^d} \mathcal{L}(w) + \lambda \Omega(w)$$

for  $\mathcal{L}(w) = \sum_i (w^\top x_i - y_i)^2$  and  $\Omega(w) = \|w\|^2$ .

$$\nabla_w [\mathcal{L}(w) + \lambda \Omega(w)] = 2 \sum_i x_i x_i^\top w - 2 \sum_i x_i y_i + 2\lambda w$$

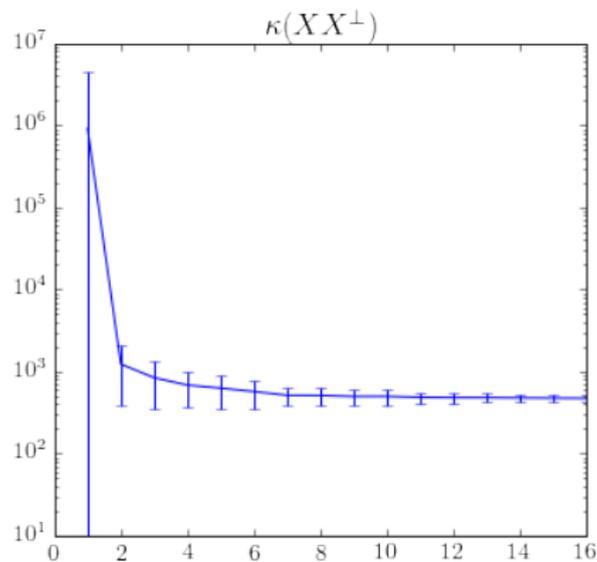
Set gradient to zero:

$$\begin{aligned} \underbrace{\sum_i x_i x_i^\top w + \lambda w}_{=} &= \sum_i x_i y_i \\ &= (\sum_i x_i x_i^\top + \lambda \text{Id}_{n \times n}) w \end{aligned}$$

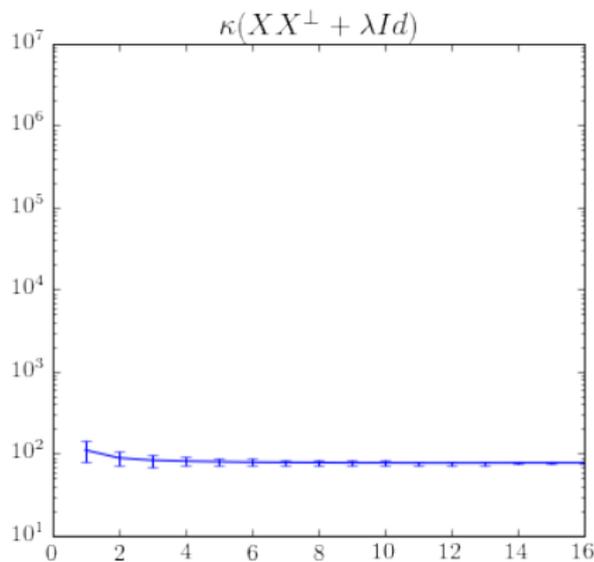
For  $\lambda > 0$ , we can *always* solve for  $w$  (regardless if  $n \geq d$ ),

$$w = \left( \sum_i x_i x_i^\top + \lambda \text{Id} \right)^{-1} \sum_i x_i y_i$$

Already small regularization strongly increases robustness (here:  $\lambda = 0.0001n$ )

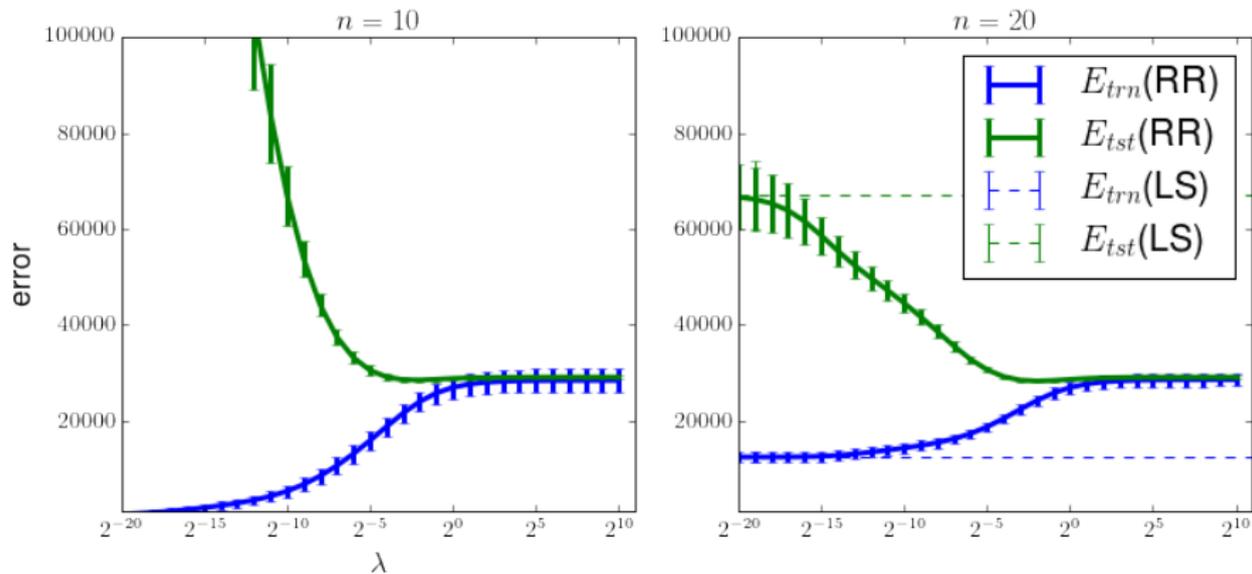


→



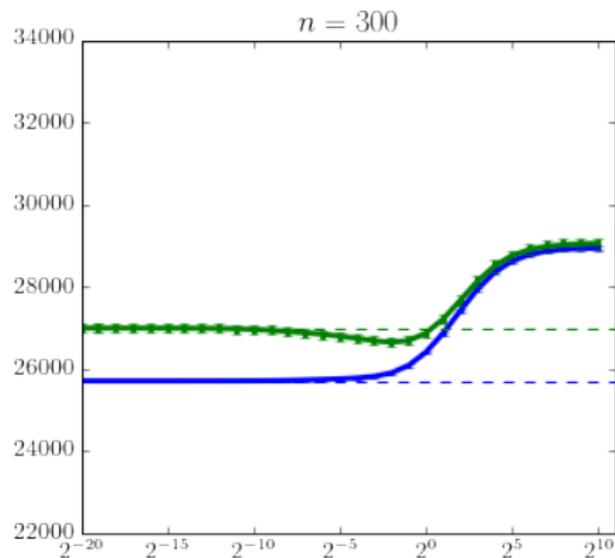
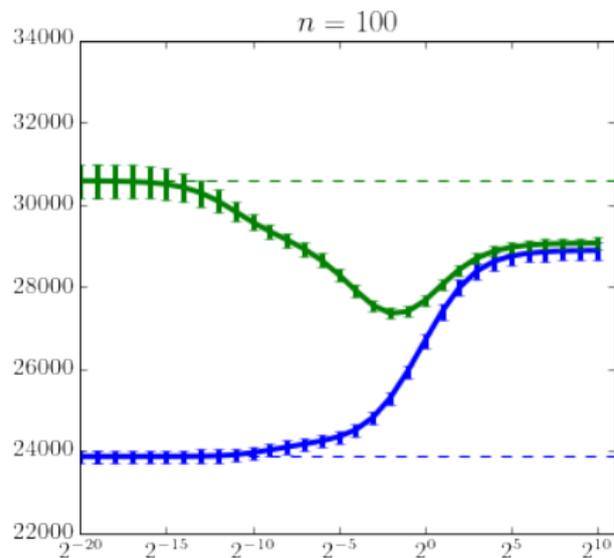
Training and test error for different regularization constants:

diabetes dataset,  $d = 10$



Training and test error for different regularization constants:

diabetes dataset,  $d = 10$



Question:

- How to select one model from many possible ones.
- How to set free parameters of a model (e.g. regularization)?

### Training and Choosing between Multiple Models (suboptimal, don't use)

```
input data  $\mathcal{D}$ , set of method  $\mathcal{A} = \{A_1, \dots, A_K\}$   
randomly split  $\mathcal{D} = \mathcal{D}_{trn} \dot{\cup} \mathcal{D}_{tst}$  disjointly  
for all possible procedures  $A_i \in \mathcal{A}$  do  
   $f_i \leftarrow A_i[\mathcal{D}_{trn}]$   
   $E(f_i) \leftarrow$  performance of  $f_i$  on  $\mathcal{D}_{tst}$   
end for  
output  $f \leftarrow f_i$  for  $i = \operatorname{argmin}_i E(f_i)$  // pick best performing  $f_i$ 
```

Question:

- How to select one model from many possible ones.
- How to set free parameters of a model (e.g. regularization)?

### Training and Choosing between Multiple Models (suboptimal, don't use)

```
input data  $\mathcal{D}$ , set of method  $\mathcal{A} = \{A_1, \dots, A_K\}$   
randomly split  $\mathcal{D} = \mathcal{D}_{trn} \dot{\cup} \mathcal{D}_{tst}$  disjointly  
for all possible procedures  $A_i \in \mathcal{A}$  do  
   $f_i \leftarrow A_i[\mathcal{D}_{trn}]$   
   $E(f_i) \leftarrow$  performance of  $f_i$  on  $\mathcal{D}_{tst}$   
end for  
output  $f \leftarrow f_i$  for  $i = \operatorname{argmin}_i E(f_i)$  // pick best performing  $f_i$ 
```

Problem: How good is the selected model? We don't know!

$\mathcal{D}_{tst}$  was used to select  $f$ , so it became part of the *training* stage.

**Proper model selection:** We simulate the model evaluation step during the training procedure. This requires one additional data split:

### Training and Selecting between Multiple Models

**input** data  $\mathcal{D}$

**input** set of method  $\mathcal{A} = \{A_1, \dots, A_K\}$

randomly split  $\mathcal{D} = \mathcal{D}_{trnval} \dot{\cup} \mathcal{D}_{tst}$  disjointly

set aside  $\mathcal{D}_{tst}$  to a safe place (and do not look at it)

randomly split  $\mathcal{D}_{trnval} = \mathcal{D}_{trn} \dot{\cup} \mathcal{D}_{val}$  disjointly

**for all** possible procedures  $A_i \in \mathcal{A}$  **do**

$f_i \leftarrow A_i[\mathcal{D}_{trn}]$

$E_{val}(f_i) \leftarrow$  performance of  $f_i$  on  $\mathcal{D}_{val}$

**end for**

$f \leftarrow f_i$  for  $i = \mathbf{argmin}_i E_{val}(f_i)$  // pick best performing  $f_i$

(optional)  $f \leftarrow A_i[\mathcal{D}_{trnval}]$  // retrain best method on full data

$E_{tst}(f) \leftarrow$  performance of  $f$  on  $\mathcal{D}_{tst}$

**output** trained model  $f$ , performance estimate  $E_{tst}(f)$

## Discussion.

- Each model is trained on  $\mathcal{D}_{trn}$  and evaluated on disjoint  $\mathcal{D}_{val}$  ✓
- Which model is selected depends on  $\mathcal{D}_{trn}$  and  $\mathcal{D}_{val}$  ✓
- Only then the "new"  $\mathcal{D}_{tst}$  is used to evaluate the single final model ✓

## Discussion.

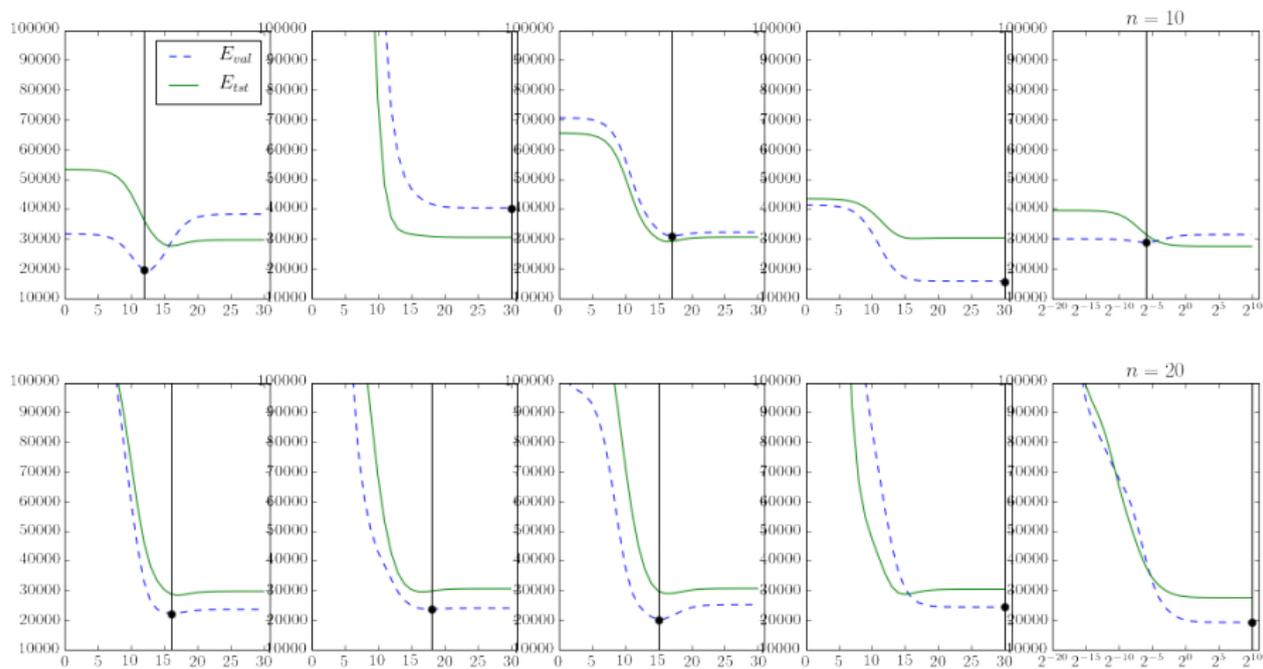
- Each model is trained on  $\mathcal{D}_{trn}$  and evaluated on disjoint  $\mathcal{D}_{val}$  ✓
- Which model is selected depends on  $\mathcal{D}_{trn}$  and  $\mathcal{D}_{val}$  ✓
- Only then the "new"  $\mathcal{D}_{tst}$  is used to evaluate the single final model ✓

## Problems.

- small  $\mathcal{D}_{val}$  is bad:  $E_{val}$  could be bad estimate of  $f_i$ 's true performance, and we might pick a suboptimal method.
- large  $\mathcal{D}_{val}$  is bad:  $\mathcal{D}_{trn}$  is much smaller than  $\mathcal{D}_{trnval}$ , so the classifier learned on  $\mathcal{D}_{trn}$  might be much worse than necessary.
- retraining the best model on  $\mathcal{D}_{trnval}$  might overcome that, but that comes at a risk: just because a model/parameter was the best for  $\mathcal{D}_{trn}$ , does not mean it is also the best for the larger  $\mathcal{D}_{trnval}$ .

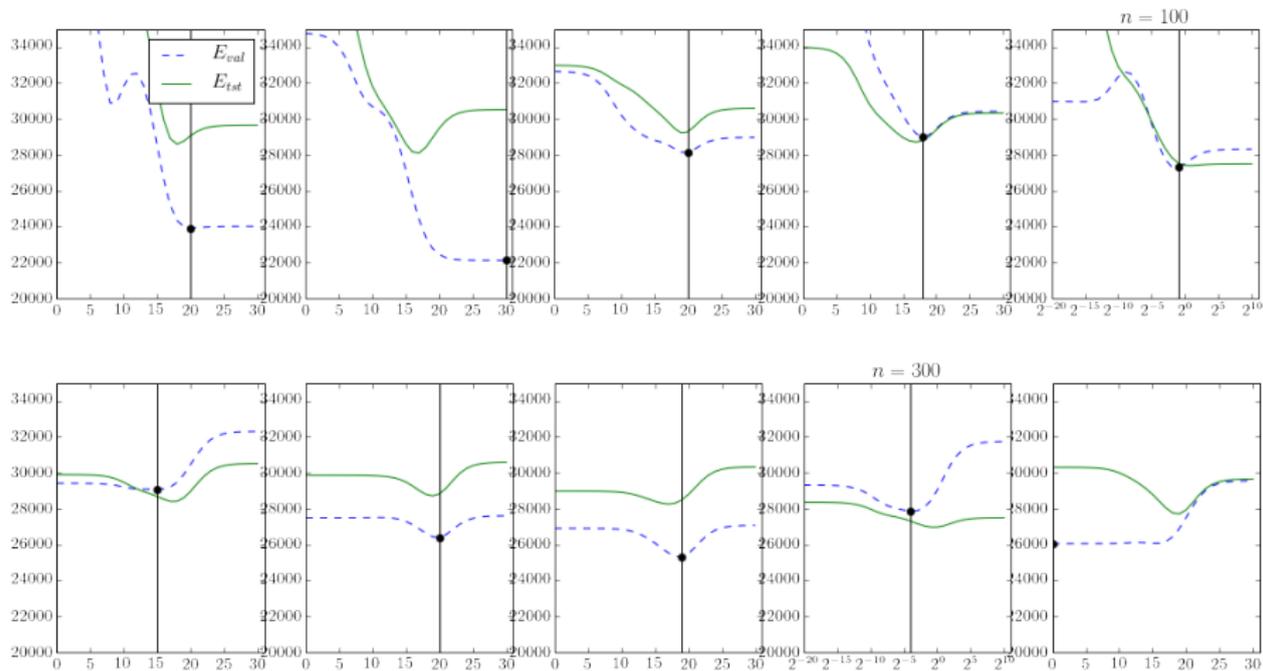
Validation error and test error for different regularization constants:

diabetes dataset,  $d = 10$



Validation error and test error for different regularization constants:

diabetes dataset,  $d = 10$



Can we use *all data* for training as well as model selection?

## Leave-One-Out Evaluation (for a single model/algorithm)

```
input algorithm  $A$ 
input data  $\mathcal{D}$       (trnval part only: tst part was set aside earlier)
  for all  $(x_i, y_i) \in \mathcal{D}$  do
     $f_{-i} \leftarrow A[\mathcal{D} \setminus \{(x_i, y_i)\}]$       //  $\mathcal{D}_{trn}$  is  $\mathcal{D}$  with  $i$ -th example removed
     $r_i \leftarrow$  performance of  $f_i$  on  $(x_i, y_i)$       // i.e.  $\mathcal{D}_{val}$  is  $\{(x_i, y_i)\}$ 
  end for
output  $E_{loo}(f) = \frac{1}{n} \sum_{i=1}^n r_i$       // average leave-one-out risk
```

### Properties.

- Each  $r_i$  is a unbiased (but high variance) estimate of the quality of  $f_{-i}$
- $\mathcal{D} \setminus \{(x_i, y_i)\}$  is almost the same as  $\mathcal{D}$ , so we can hope that each  $f_{-i}$  is almost the same as  $f = A[\mathcal{D}]$ .
- Therefore,  $E_{loo}$  can be expected a good estimate of  $E$  on new data

**Problem:** slow, trains  $n$  times on  $n - 1$  examples instead of once on  $n$

**Problem:** all training sets are almost the same,  $r_i$  are correlated

Compromise: use fixed number of small  $\mathcal{D}_{val}$

## $K$ -fold Cross Validation (CV)

**input** algorithm  $A$ , loss function  $\ell$ , data  $\mathcal{D}$  (train/val part)

split  $\mathcal{D} = \dot{\bigcup}_{k=1}^K \mathcal{D}_k$  into  $K$  equal sized disjoint parts

**for**  $k = 1, \dots, K$  **do**

$f_{-k} \leftarrow A[\mathcal{D} \setminus \mathcal{D}_k]$

$r_k \leftarrow$  performance of  $f_{-k}$  on  $\mathcal{D}_k$

**end for**

**output**  $R_{K-CV} = \frac{1}{K} \sum_{k=1}^K r_k$  ( $K$ -fold cross-validation risk)

### Observation.

- for  $K = |\mathcal{D}|$  same as leave-one-out error.
- approximately  $k$  times increase in runtime.
- most common:  $k = 10$  or  $k = 5$ .

**Remaining problem:** training sets overlap, so the error estimates are not independent, and it's hard to interpret error bars or design statistical tests