Lecture 2

15.09.2025

Today's plan and announcements

- Review: last week's summary
- Loss function for linear regression
- Optimizing the loss function
- Overfitting/underfitting

- Admin matters
  - Use exercise hour to ask your course related questions
  - Videos are unrecorded. Slides are on Moodle.

Review from last week

- Artificial intelligence: the field of building systems that perceive, reason and act in ways that replicate or augment human intelligence. But how do we define intelligence?

Your professor's answer: Let's think of what kind of world we like to live in (global peace, equity, sustainability) and think of what intelligence we need for humans and machines to get there..

- Machine learning: a data-driven approach to artificial intelligence

- Supervised versus unsupervised learning: In supersvised learning, input data has labels: in unsupervised learning it doesn't

Supervised learning - linear predictors

- Training data $\{x^i, y^i\}$, $x^i \in \mathbb{R}^d$

- Linear predictor $f(x) = b + w_1 x_1 + \ldots + w_d x_d$
  - Parameters to be determined: $(b, w_1, w_2, \ldots, w_d)$
  - Prediction:

$$x^{test} \in \mathbb{R}^d, \quad f(x^{test}) = b + w_1 x_1^{test} + \cdots + w_d x_d^{test}$$

- Training or tuning or fitting parameter: finding the parameters based on training data

Important notation for the remainder of course

- Consider the parameter vector $(b, w_1, w_2, \ldots, w_d) \in \mathbb{R}^{d+1}$
- These are the $d + 1$ parameters we need to find for determining the linear predictor
- We will from now on define $w := (b, w_1, w_2, \ldots, w_d) \in \mathbb{R}^{d+1}$
- By further letting $x := (1, x_1, x_2, \ldots, x_d) \in \mathbb{R}^{d+1}$

We can write our linear predictor

$$f(x) = b + w_1 x_1 + \ldots + w_d x_d = (b, w_1, \ldots, w_d)^T (1, x_1, \ldots, x_d)$$

equivalently as

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

Loss function for linear regression

- Prediction error for a single data point x^i : $f_w(x^i) = b + w_1 x_1^i + \ldots w_d x_d^i = w^T x^i$

$w^T x^i - y^i$ — true label

- Squared prediction error

$(w^T x^i - y^i)^2$

- Loss function is the mean squared error of the predictor on the training set
  - Note: loss function also referred to as cost function or objective function

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} (w^T x^i - y^i)^2$$

Optimization - brief review

Optimizing an unconstrained function - review

- Gradient of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$

$$\nabla f(x) \in \mathbb{R}^n$$

- Consider the unconstrained optimization: $\min_{x \in \mathbb{R}^n} f(x)$ (1)
  - Necessary condition for optimality of a given point

$x^* \in \arg\min f(x)$ (in other words its optimal for (1))

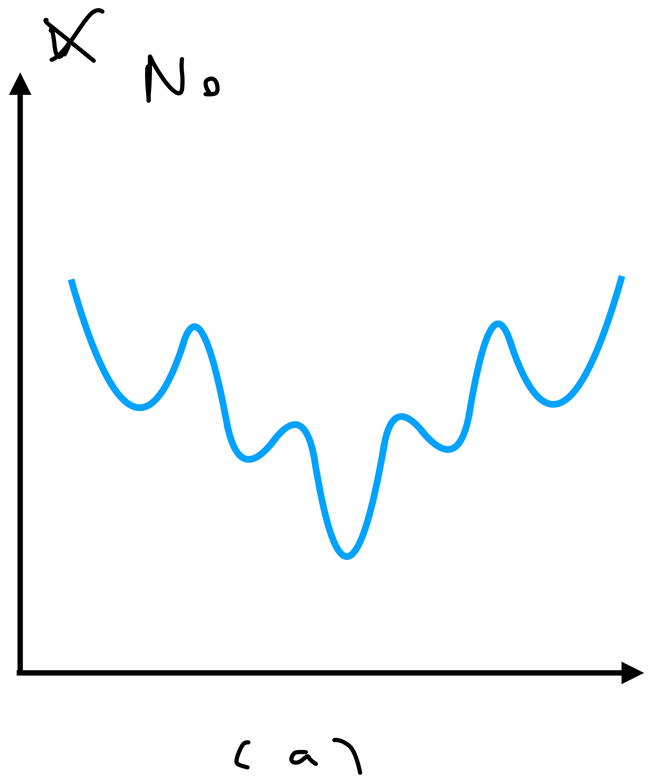
$\Rightarrow \nabla f(x^*) = 0$.

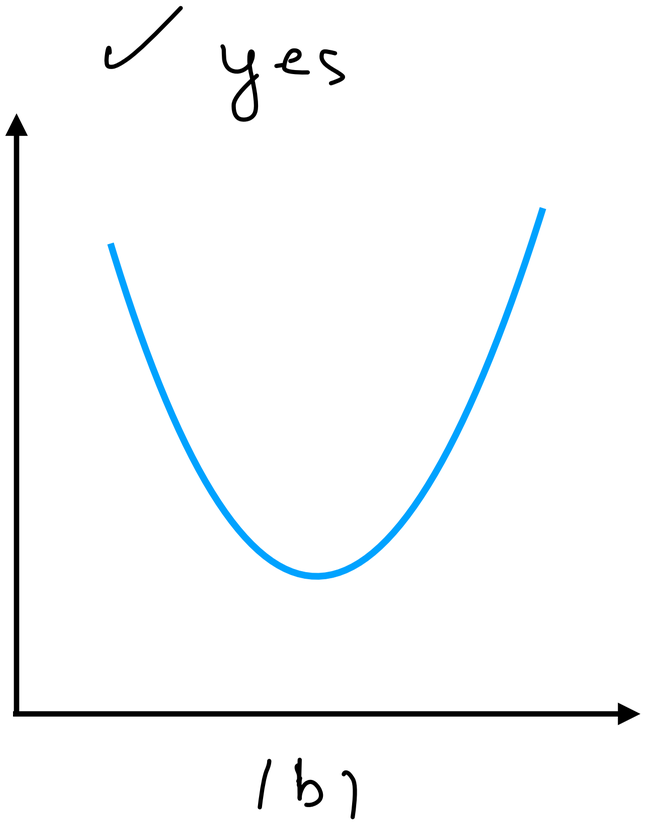
x^* optimal $\Rightarrow \nabla f(x^*) = 0_{n \times 1} \in \mathbb{R}^n$

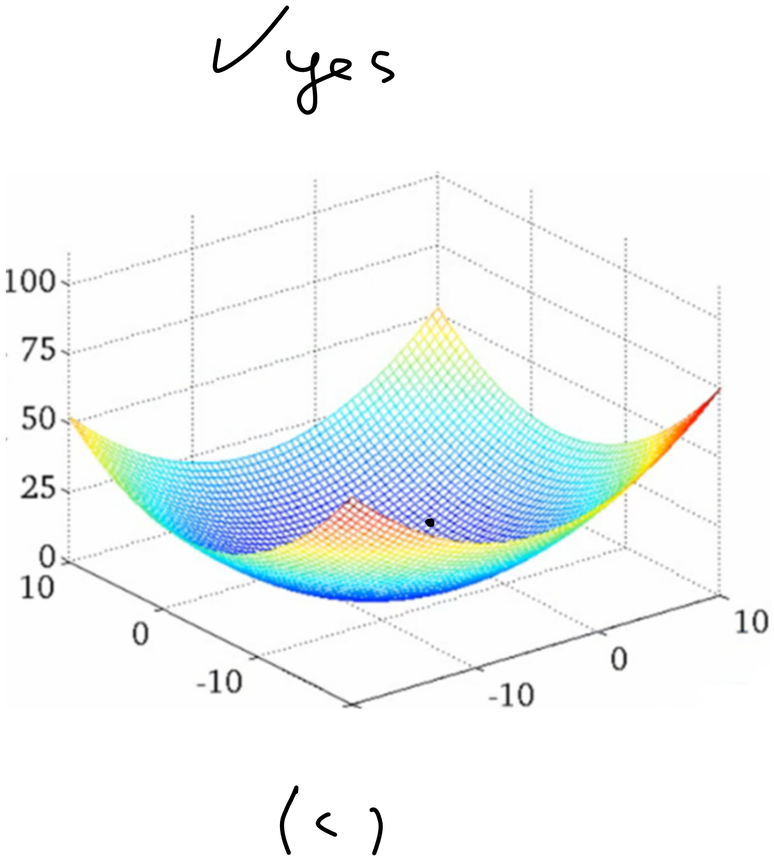
- In which cases the above condition is also sufficient for optimality?

Optimizing an unconstrained function

Consider x^* such that $\nabla f(x^*) = 0$. For which of these functions x^* is an optimizer?





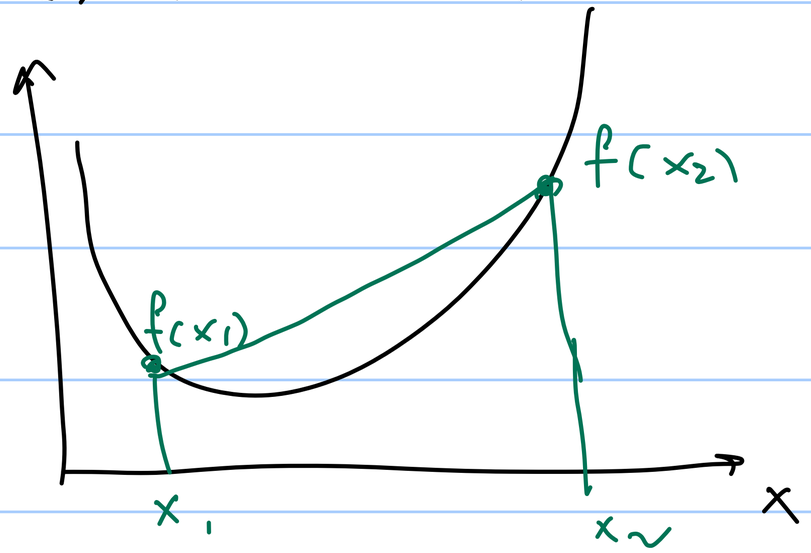


Optimization - convex functions

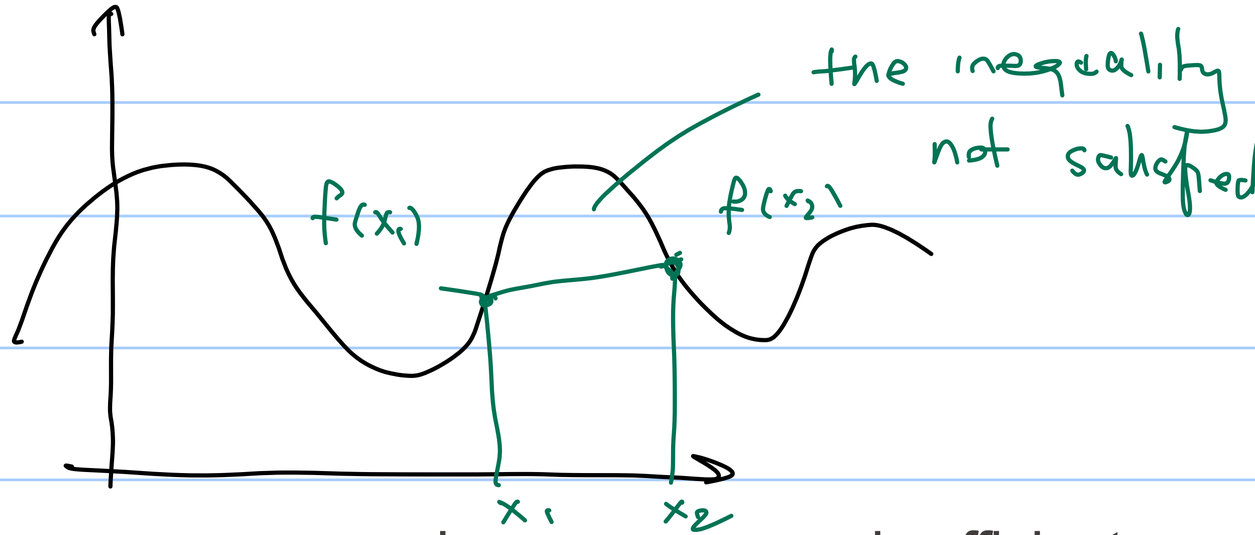
- A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $\forall \lambda \in [0,1]$, $\forall x_1, x_2 \in \mathbb{R}^n$

$$f(\lambda x_1 + (1-\lambda) x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

ex 1 ✓ convex



ex 2 non. convex



- For convex functions, the optimality condition is necessary and sufficient.

$$\nabla f(x^*) = 0 \iff x^* \text{ is optimizer (optimal point)}$$

Review - gradient vector and Hessian matrix

- Consider $f : \mathbb{R}^n \to \mathbb{R}$

- Gradient: vector of first-order derivative of a scalar function

$$\nabla f(x) \in \mathbb{R}^n$$

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)^T \in \mathbb{R}^n$$

- Hessian: matrix of second-order derivative of a scalar function

$$\nabla^2 f(x) \in \mathbb{R}^{n \times n}$$

$$[\nabla^2 f(x)]_{ij} = \frac{\partial^2 f}{\partial x^i \partial x^j}$$

- Example 1 : $f(x) = a^T x$

$$\nabla f(x) = a \in \mathbb{R}^n$$

$$\nabla^2 f(x) = 0_{n \times n}$$

- Example 2 : $f(x) = x^T A x$

$$A \in \mathbb{R}^{n \times n}$$

$$\nabla f(x) = (A + A^T) x$$

$$\nabla^2 f(x) = A + A^T$$

Test for convexity

- A twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if its Hessian is positive semi-definite.
  - Don't confuse the notation for Hessian here $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$ with the Laplacian operator

- Recall: a matrix $A \in \mathbb{R}^{n \times n}$ is positive semi-definite if all eigenvalues of $A + A^T$ have non-negative real parts. It is called positive definite if they have positive real-part.
  - What if the matrix is symmetric? $A^T = A$, sufficient to check eigenvalues of A.

Review: positive (semi)definite matrices, eigenvalues and eigenvectors

- Given $A \in \mathbb{R}^{n\times n}$, $\lambda \in \mathbb{C}$ is eigenvalue of A if $\exists\, v \in \mathbb{R}^n$ such that $Av = \lambda v$

$v \neq 0_{n\times 1}$

called eigenvector of A.

- Examples: $A = \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$, verify eigenvalues are $\lambda_1 = 2$, $\lambda_2 = 1$

- Show that for a matrix $A = CC^T$, with $C \in \mathbb{R}^{n\times l}$, all eigenvalues are non-negative and real.

Let λ be an eigenvalue of A $\Rightarrow$

$\exists\, v \in \mathbb{R}^n$ s.t.

$$Av = \lambda v \quad \text{by } A = CC^T$$

$$CC^T v = \lambda v \quad \text{by premultiply with } v^T$$

$$\|C^T v\|_2^2 = v^T CC^T v = \lambda v^T v = \lambda \|v\|_2^2, \quad \lambda = \frac{\|C^T v\|_2^2}{\|v\|_2^2} \geq 0$$

Back to original question: is the linear regression loss function convex and thus, *easy* to optimize?

Gradient and Hessian of linear regression loss function

- Loss function $J(w) = \frac{1}{N}\sum_{i=1}^{N}(w^T x^i - y^i)^2 = \frac{1}{N}\sum_{i=1}^{N}(w^T x^i - y^i)(w^T x^i - y^i)$

- Gradient of the loss function $\nabla_w J(w) = \nabla_w \frac{1}{N}\sum_{i=1}^{N}(w^T x^i - y^i)^2$

$= \frac{1}{N}\sum_{i=1}^{N}\nabla_w (w^T x^i - y^i)^2 = \frac{2}{N}\sum_{i=1}^{N}\underbrace{(w^T x^i - y^i)}_{\text{scalar}}\underbrace{x^i}_{\text{vector in } \mathbb{R}^{d+1}} \in \mathbb{R}^{d+1}$

Recall $x^i = (1, x^i_1, \dots, x^i_d)$

- Hessian of the loss function

$\nabla^2_w J(w) = \frac{2}{N}\sum_{i=1}^{N}(x^i)(x^i)^T$

, $x^i \in \mathbb{R}^{d+1} \Rightarrow x^i (x^i)^T \in \mathbb{R}^{(d+1)\times(d+1)}$

Verifying convexity of linear regression loss

- We just showed that

$$\nabla_w J(w) = \left(\frac{\partial J(w)}{\partial b}, \frac{\partial J(w)}{\partial w_1}, \frac{\partial J(w)}{\partial w_2}, \ldots, \frac{\partial J(w)}{\partial w_d} \right)^T = \frac{2}{N} \sum_{i=1}^{N} x^i (w^T x^i - y^i), \quad \text{where } w \in R^{d+1} = \begin{pmatrix} b \\ w_1 \\ \cdots \\ w_d \end{pmatrix}$$

$$\frac{\partial^2 J}{\partial^2 w} = \frac{2}{N} \sum_{i=1}^{N} x^i (x^i)^T \in \mathbb{R}^{(d+1)\times(d+1)}$$

- As the Hessian above is positive semi-definite, the loss function is convex

(Recall $A = CC^T$, $C \in \mathbb{R}^{n \times 1}$ is positive semidefinite)

Exercise: gradient and Hessian using data matrix

Define data matrix: $X \in R^{N\times(d+1)} = \begin{pmatrix} 1 & x_1^1 & x_2^1 & \dots & x_d^1 \\ 1 & x_1^2 & x_2^2 & \dots & x_d^2 \\ \dots & & & & \\ 1 & x_1^N & x_2^N & \dots & x_d^N \end{pmatrix}$ **Recall:** $w \in R^{d+1} = \begin{pmatrix} b \\ w_1 \\ \dots \\ w_d \end{pmatrix}$

$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

Verify that: $J(w) = \frac{1}{N}\sum_{i=1}^{N}(w^T x^i - y^i)^2$ $\in \mathbb{R}$ **can be equivalently written as :** $J(w) = \frac{1}{N}(Xw - y)^T(Xw - y) \in \mathbb{R}$

Conclude: $\nabla J(w) = \frac{2}{N}X^T(Xw - y), \quad \nabla^2 J(w) = \frac{2}{N}X^T X$

Hessian is $\frac{2}{N}$ CC^T where

$C = X^T$, Hence, positive semi defnd

Finding the optimal parameters of linear regression

- Since the loss function is convex, we can find the optimizer and the corresponding optimal value by setting the derivative of the loss to zero:

$$\frac{\partial J(w)}{\partial w} = \frac{2}{N} X^T (Xw - y) = 0 \iff w = (X^T X)^{-1} X^T y$$

X, y are data

$X^T \in \mathbb{R}^{(d+1) \times N}$, $y \in \mathbb{R}^N$

- The optimizer $w^* = (\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\,y$
  - Note: this assumes invertibility of the matrix. Under which conditions it would hold?

$X^T X$

- $N \geq d$, and also, linearly independent measurement.

- The loss evaluated at the optimal parameter $J(w^*) = \frac{1}{N}\sum_{i=1}^{N} (w^{*T} x^i - y^i)^2 = \dots$

Exercise: plug in w^* &

Gradient descent for optimization

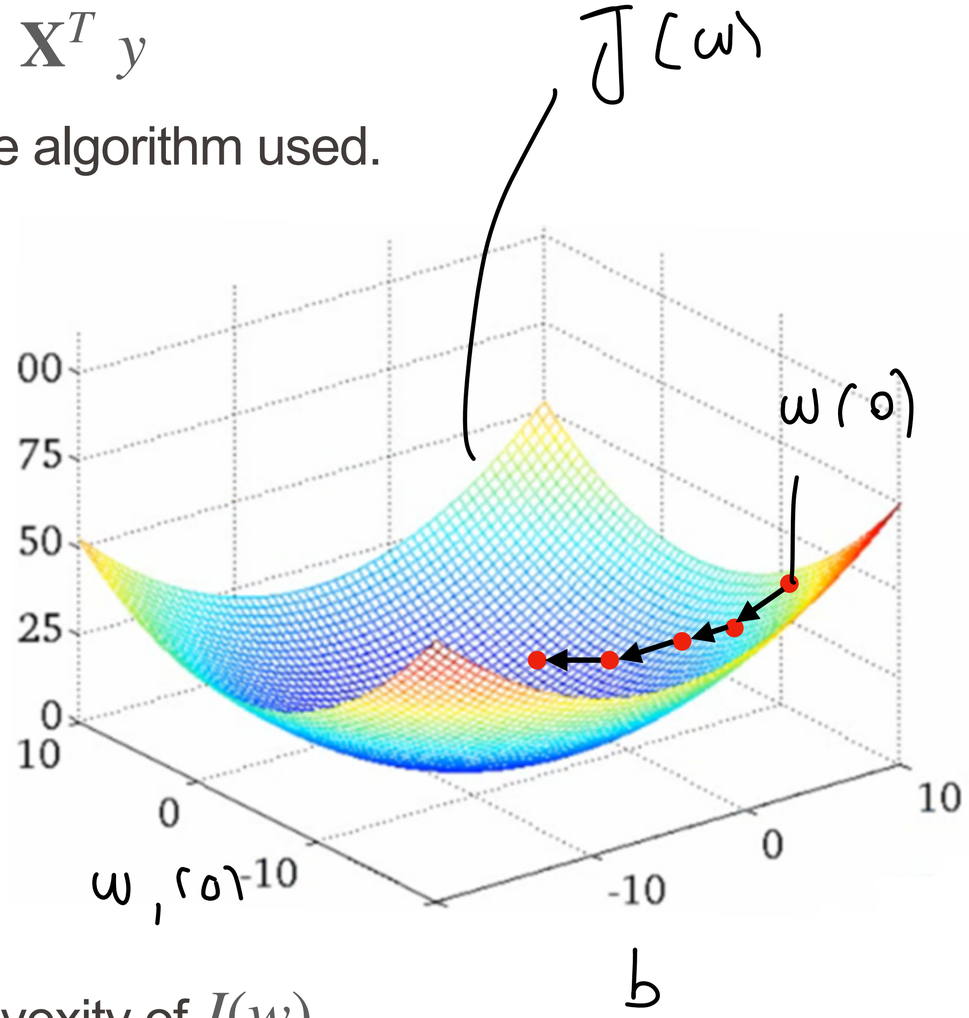
Need inverse of $\mathbf{X}^T\mathbf{X}$ to compute parameters $w^* = (\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\,y$

Computational complexity is $O(d^{2.4})$ to $O(d^3)$ depending on the algorithm used.

Gradient descent: Alternative approach to compute: $\arg\min_w J(w)$

- Calculate the gradient of $J(w)$ at the starting point.
- Iteratively update w by taking steps in the direction of the negative gradient

$$w(t+1) = w(t) - \alpha \nabla J(w(t))$$



Can find step size $\alpha > 0$ to ensure convergence to w^* due to convexity of $J(w)$

Linear regression with nonlinear features

Feature expansion of data

Feature expansion: transforming input and output data using a map

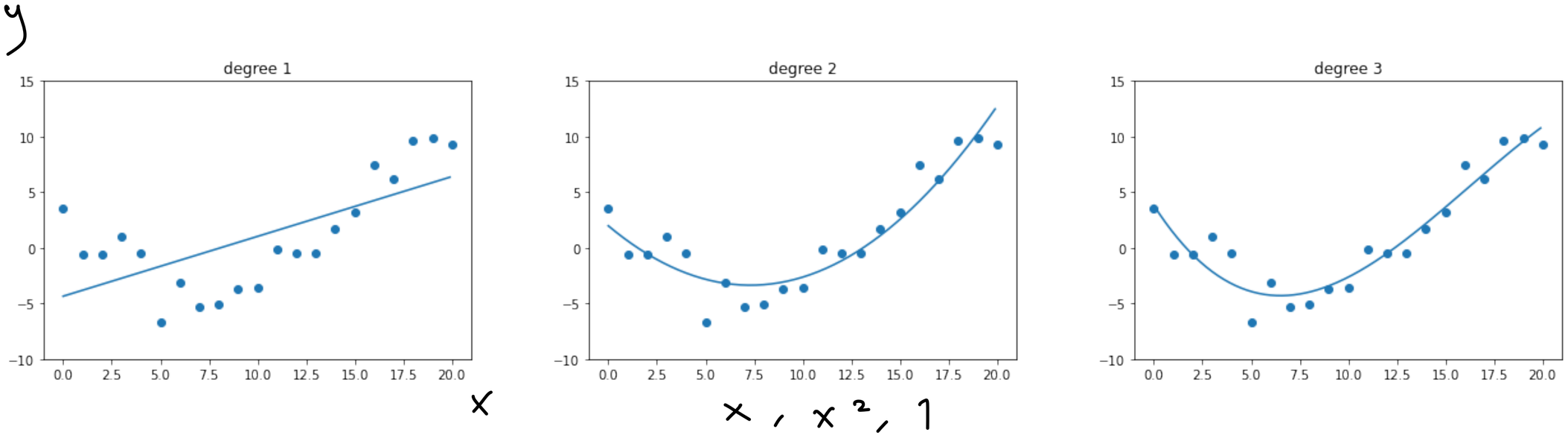
- Why? can significantly improve performance
- How? often requires insight about the problem

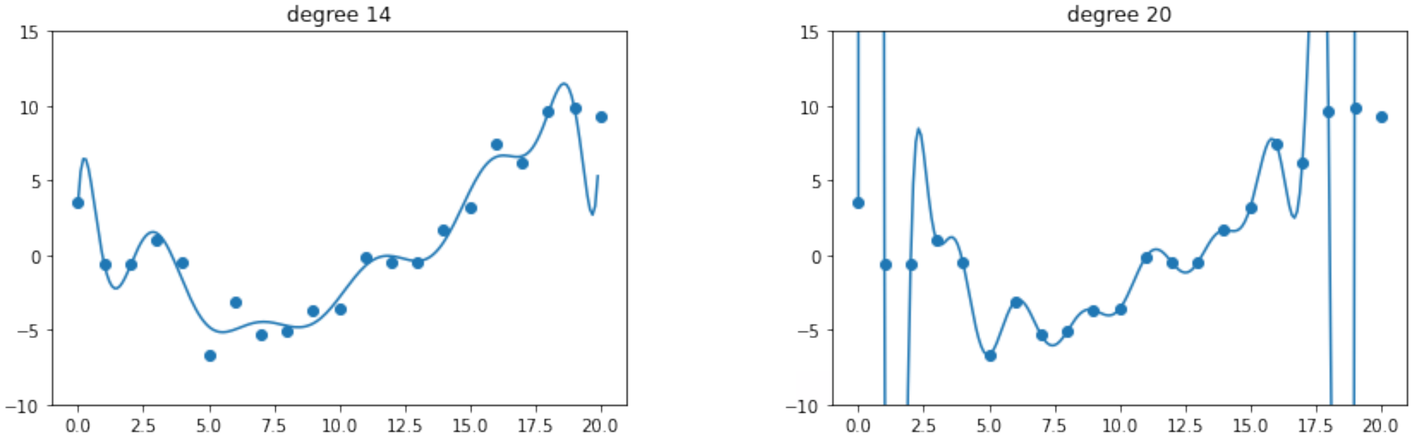
Popular feature expansion techniques:

- Applying a non-linear function to each feature (*e.g.,* sin, log, polynomials)
- Example: polynomial feature expansion

$$\boldsymbol{x} = (x_1, x_2, x_3) \xrightarrow{\text{Second order polynomial feature expansion}} (1,\ x_1,\ x_2,\ x_3,\ x_1^2,\ x_1x_2,\ x_1x_3,\ x_2^2,\ x_2x_3,\ x_3^2)$$

Example of polynomial feature expansion





Linear regression on transformed data

- Consider a feature map $\phi : \mathbb{R}^n \to \mathbb{R}^p$

example $x \in \mathbb{R}^2$, $\phi_1(x) = x_1, x_2$, $\phi_2(x) = \sin x_1$

$\phi : \mathbb{R}^2 \to \mathbb{R}^2$, $\phi(x) = (x_1, x_2, \sin x_1)$

- Predictor is linear in the transformed features

$$f_w(x) = b + w_1 \phi_1(x) + w_2 \phi_2(x) + \cdots + w_p \phi_p(x)$$

- Exercise: Given the transformed features, derive the loss function, its gradient and Hessian. Derive the optimal parameters.

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} \left(b + w_1 \phi_1(x^i) + \cdots + w_p \phi_p(x^i) - y^i \right)^2$$

Overfitting and underfitting

Linear regression in supervised learning

- You have seen linear regression in your past courses. So, why do we do it in this course again?
  - It helps understand more advanced methods: logistic regression, neural nets, support vector machines, etc.
- What are the differences between how we do linear regression in machine learning versus in past courses such as in physics?
  - In physics/engineering the emphasis is on *inference: the relationship between input and output is known to be linear/well-approximated by a linear function and we want to find the true parameters of the linear map*
  - In machine learning, the emphasis now is on *prediction: does model perform well on new data points,* rather than if the model is truly correct

Train and test

all N data points $\{x^i, y^i\}_{i=1}^N$

To evaluate the predictive power of model (how well the model does on unseen input data), divide the data into **train** and **test**

Train the model (setting the parameters) based on the **Train** set. Evaluate the model based on the **Test** set

Note: Later, we will split the data into 3 sets (train, validate and test sets) in order to find best *hyper-parameters* of the model

Performance on the train and test set

Goal of supervised ML models: Predict well on new data based on the patterns learned from known data. This is also referred to as generalization

Performance metric: $J(w) = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} (w^T x^i - y^i)^2$

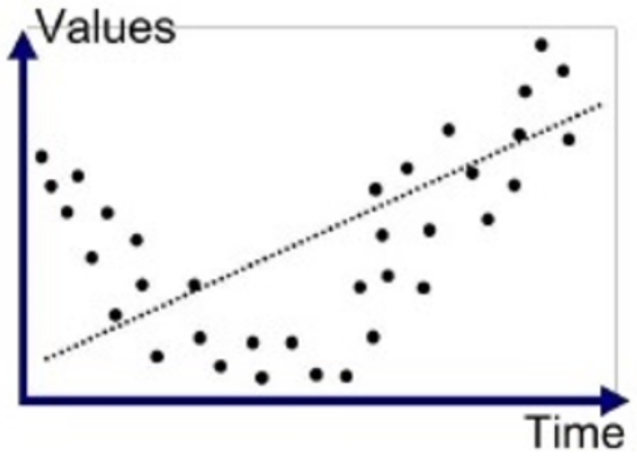
N_{train} : # of data points saved for training

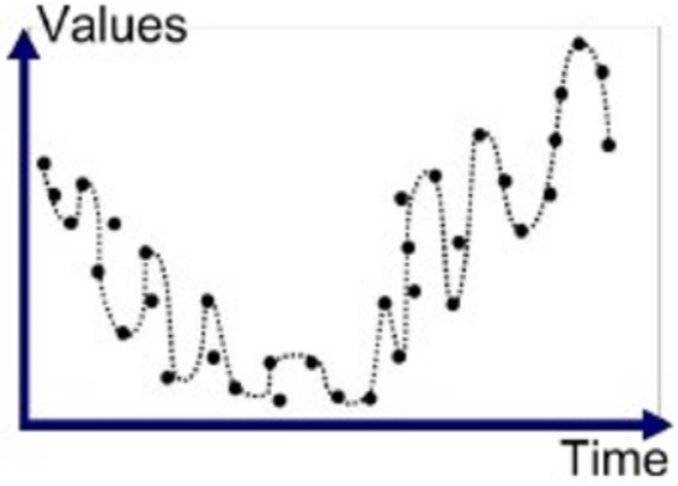
Training versus Test performance: x^{test} which wasn't in the training set ... what is the error $(w^T x^{test} - y^{test})^2$?

What kind of problems can occur in terms of generalization?

Underfitting & overfitting

- Goal of supervised ML models: generalise well on new data based on the patterns learned from known data
- Two situations where it fails:
  - Underfitting: the model doesn't fit well on the training data.
  - Overfitting: Fits well on training data, but doesn't generalise well to unknown data.





Approaches to overfitting/underfitting

Possible approaches to address underfitting

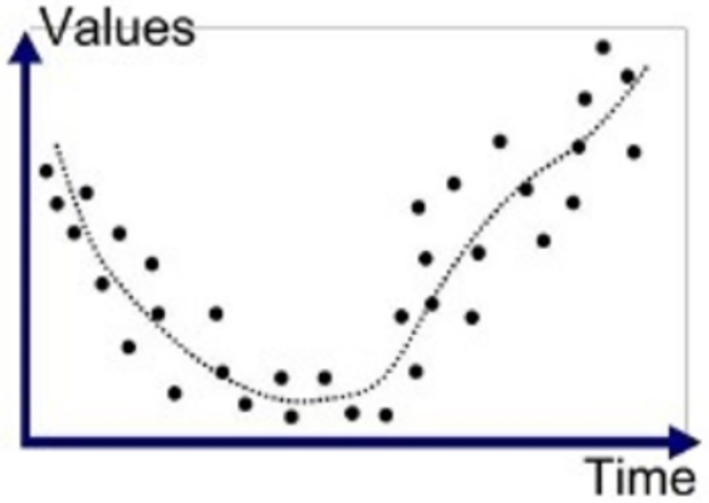
- More complex models, e.g. nonlinear feature expansion

Possible approaches to address overfitting

- Simpler model → fit the data and not the noises and errors
- More training data (→ less effect of noise)
- Add a regularisation term (common solution)

let focus on this

Overall aim: find the equilibrium between *fitting the training data* and *keeping the model simple enough to ensure it will generalise well on new data.*



Regularization - motivation based on Lipschitz constant

- Sensitivity of a predictor: *similar* data should lead to *similar* outcome
  - Less sensitive models tend to not overfit
- Let's define similar data $x, x' \in \mathbb{R}^n$ are similar if $\| x - x' \|_2$ is small.

Recall: $\| x \|_2 = (x^T x)^{1/2}$ Euclidean 2-norm

- Let's define similar outcome $f : \mathbb{R}^n \to \mathbb{R}$, $|f(x) - f(x')|$ should be small

Sensitivity of a linear predictor and regularization

Let $f_w(x) = b + w_1 x_1 + \cdots + w_d x_d$

$$|f_w(x) - f_w(x')| = |b + w_1 x_1 + \cdots + w_d x_d - (b + w_1 x_1' + \cdots + w_d x_d')|$$

$$= |w_1(x_1 - x_1') + w_2(x_2 - x_2') + \cdots + w_d(x_d - x_d')|$$

$$= |(w_1, \cdots, w_d)^T \begin{bmatrix} x_1 - x_1' \\ \vdots \\ x_d - x_d' \end{bmatrix}| \leq \|w\|_2 \|x - x'\|_2$$

↳ from Cauchy-Schwarz inequality

- Regularizing the weights

So, for f_w to be not too sensitive,

we want $\|w = (w_1, \cdots, w_d)\|_2$ to be small

Exercise - optimize a regularized loss function

- Consider the regularized loss function $J(w) = \frac{1}{N}\sum_{i=1}^{N}(w^T x^i - y^i)^2 + \lambda(w_1^2 + w_2^2 + \ldots + w_d^2)$

Regularized

original

$\|(w_1, \ldots, w_d)\|_2^2$

How do we set λ?

- Find the gradient of the regularized loss function with respect to the model parameters

- How can we find the optimal parameters corresponding to the regularized loss?

λ in this case is the hyper-parameter of the model.

- Next - how do we set the regularization parameter?

hyper-parameter tuning

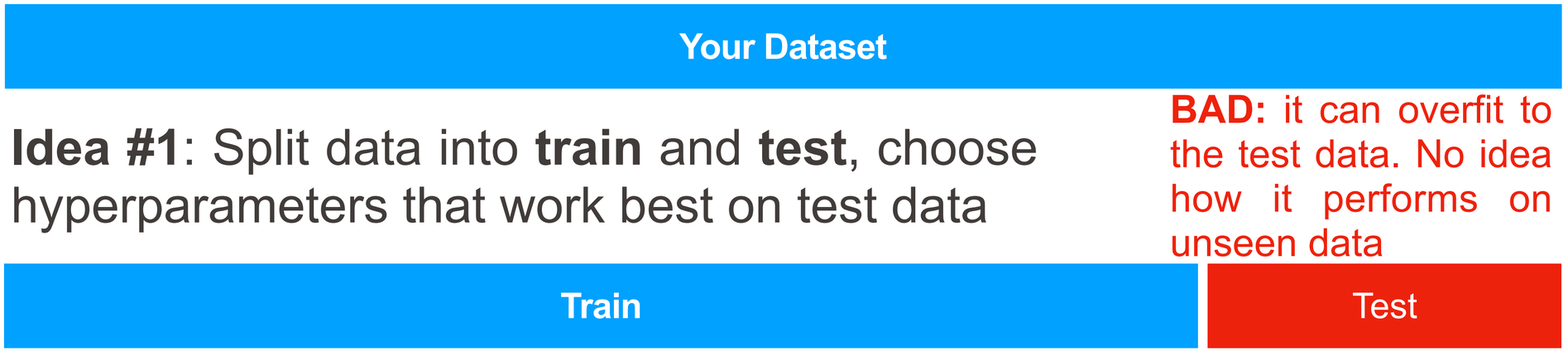
Train, validate, test in ML

Setting hyperparameters: example, the regulariser

Train, validate, test in ML

Setting hyperparameters

1000 data points



Idea #1: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: it can overfit to the test data. No idea how it performs on unseen data

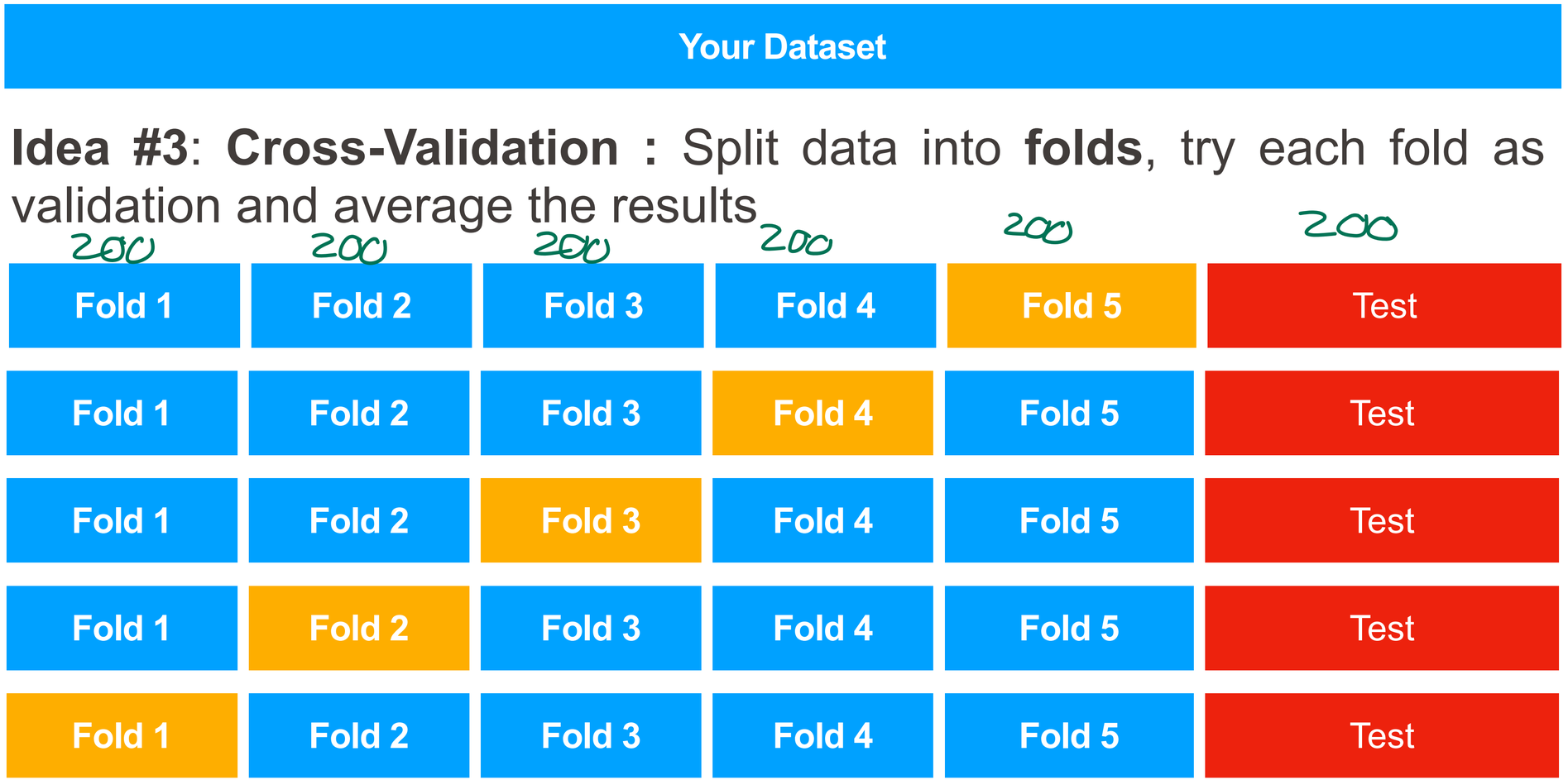
Idea #2: Split data into **train, validation** and **test**, choose hyperparameters on validation and evaluate performance on test

800 100 100

Potential challenge: 1) for small data sets, the train/validation and test sets become small; 2) we don't want performance too dependent on the possible data points in each set

Cross-fold validation

What if we don't have enough data



See Problem set 1 for more details

Summary

- Empirical loss function/cost function for linear regression
- Optimizing the empirical loss function
  - Gradients, Hessians
  - Convex functions, convex quadratic functions
  - Unconstrained optimization

- Overfitting/underfitting
  - Sensitivity of a predictor
  - Regularization
  - train, validat, test

Your tasks this week

- Go through lecture 2 slides
- Review background on gradient, Hessian and optimization (see notes posted)
- Do the exercises in lecture slides
- Do the python exercises posted on Moodle
- Bring your questions to exercise hour

- Next Monday 22.09: no class
- Next Wednesday 24.09: we do have exercise hour