

CS 480/680

Introduction to Machine Learning

Lecture 2

Linear Regression and Loss Function Design

Kathryn Simone

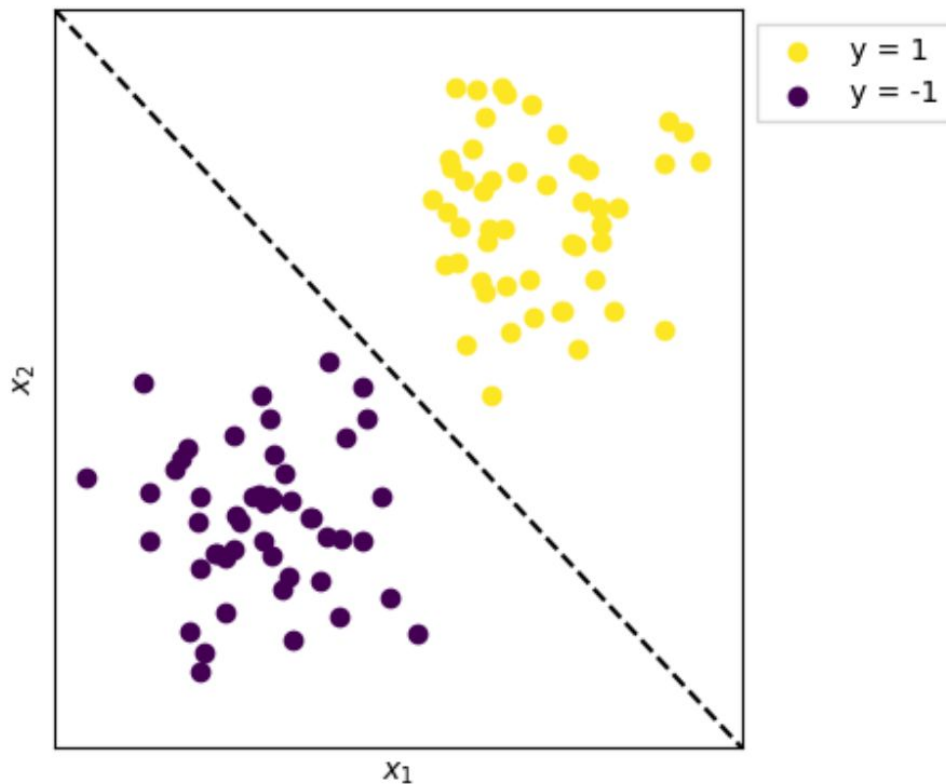
12 September 2024



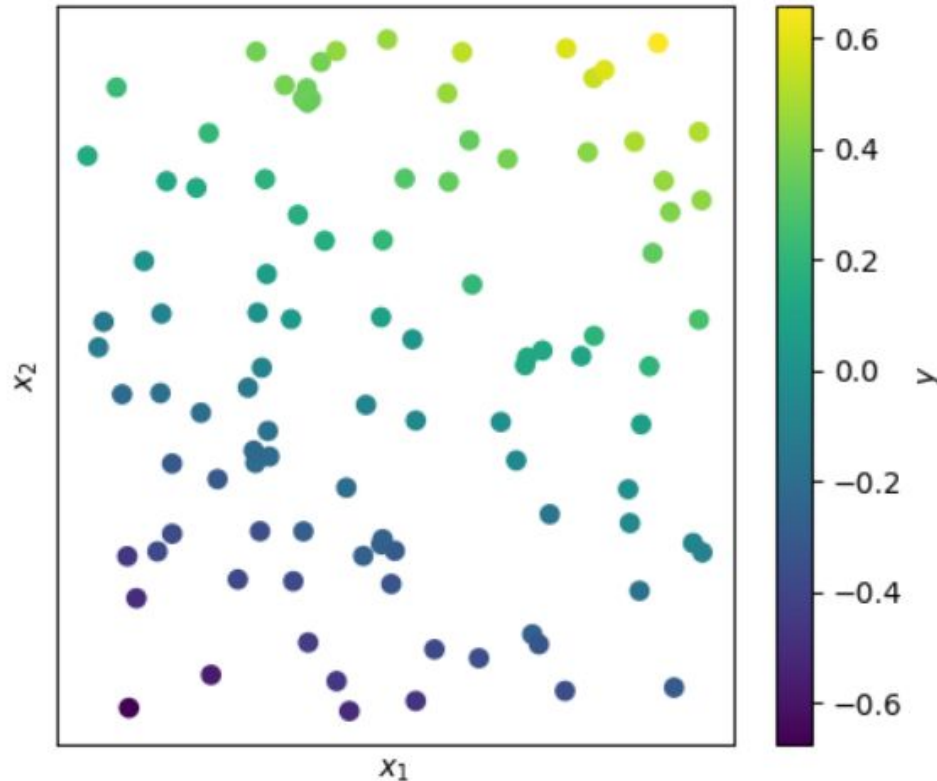
UNIVERSITY OF
WATERLOO

FACULTY OF
MATHEMATICS

Last lecture: The perceptron algorithm learns a hyperplane to classify linearly separable data



In regression, the goal is to predict continuous values



How can we learn in this setting?

1. Expand on our idea of “mistake” to deviation from ideal behavior
2. Select or design a loss *function*
3. Find the parameters that minimize the loss function



Lecture Aims

At the end of the lecture, we should be able to:

- ★ Write code to solve a simple regression problem numerically, given a dataset.
- ★ Characterize and design loss functions using correct terminology and sound mathematical principles.
- ★ Adhere to best practices for model evaluation and iterative improvement.

Lecture Outline

I. What's the basic process for solving regression?

Models, loss functions, and empirical risk minimization

II. What should one consider in loss function design?

Designing for optimization, stability, and generalization

III. How do you evaluate model performance iteratively?

Overfitting, data splits, and cross validation

IV. Summary + Housekeeping



Lecture Outline

I. What's the basic process for solving regression?

Models, loss functions, and empirical risk minimization

II. What should one consider in loss function design?

Designing for optimization, stability and generalization

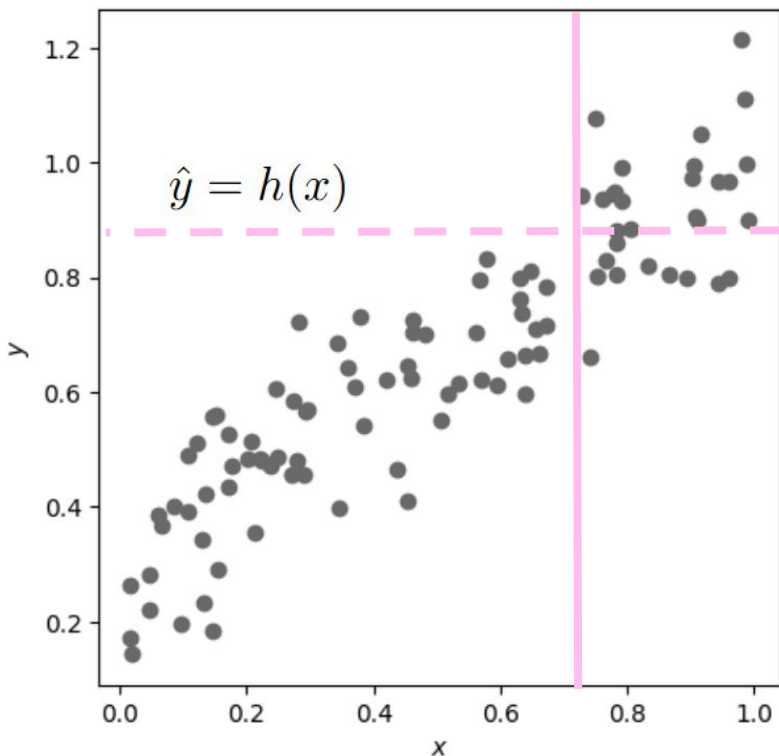
III. How do you evaluate model performance iteratively?

Overfitting, data splits, and cross validation

IV. Summary + Housekeeping



The Regression Problem



Given: $(\vec{x}_1, y_1), \dots, (\vec{x}_k, y_k), \vec{x}_i \in \mathbb{R}^d, y \in \mathbb{R}$
Goal: Learn $h : \mathbb{R}^d \rightarrow \mathbb{R}$ that best approximates relationship between variables.



From last lecture: Statistical (Batch) vs. Online Learning

Online Learning:

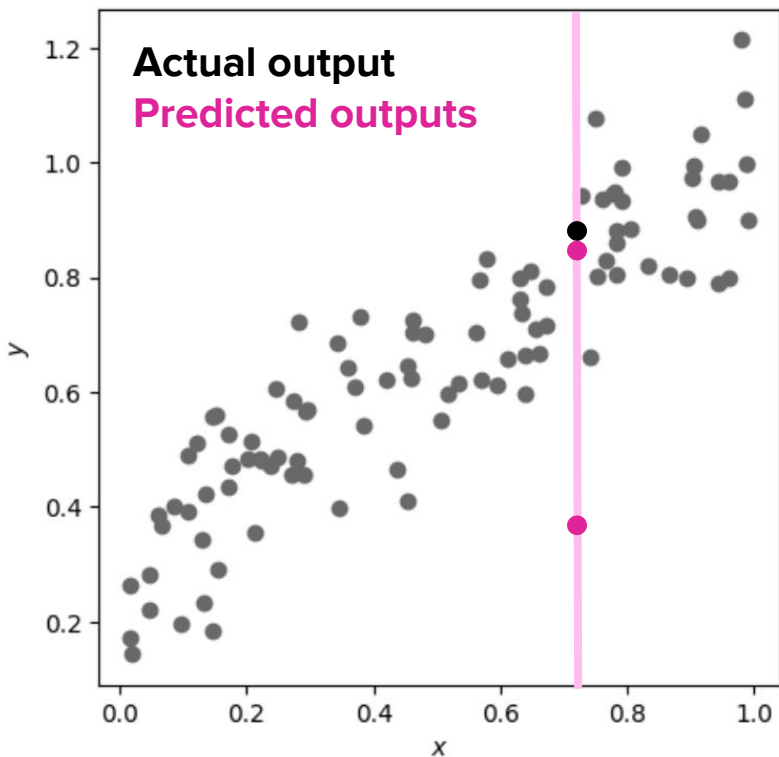
- Learner has access to a data *stream*
- Prediction is made before knowing its true value
- Interested in minimizing the number of errors

Batch Learning:

- Given a training set $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_k, y_k) \sim_{i.i.d} P$, where
 - *i.i.d*: independently and identically distributed
 - P : some unknown distribution
- Goal: learn $h : \mathbb{R}^d \rightarrow \{\pm 1\}$ such that $\Pr_{(x,y) \sim P}[h(x) = y]$ to be large.



Restating the regression problem



Batch Learning:

- Given a training set $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_k, y_k) \sim_{i.i.d} P$, where
 - *i.i.d.*: independently and identically distributed
 - P : some unknown distribution
- Goal: learn $h : \mathbb{R}^d \rightarrow \{\pm 1\}$ such that $\Pr_{(x,y) \sim P}[h(x) = y]$ to be large.

- ~~(Classification) learn $h : \mathbb{R}^d \rightarrow \{\pm 1\}$~~
- (Regression) learn $h : \mathbb{R}^d \rightarrow \mathbb{R}$

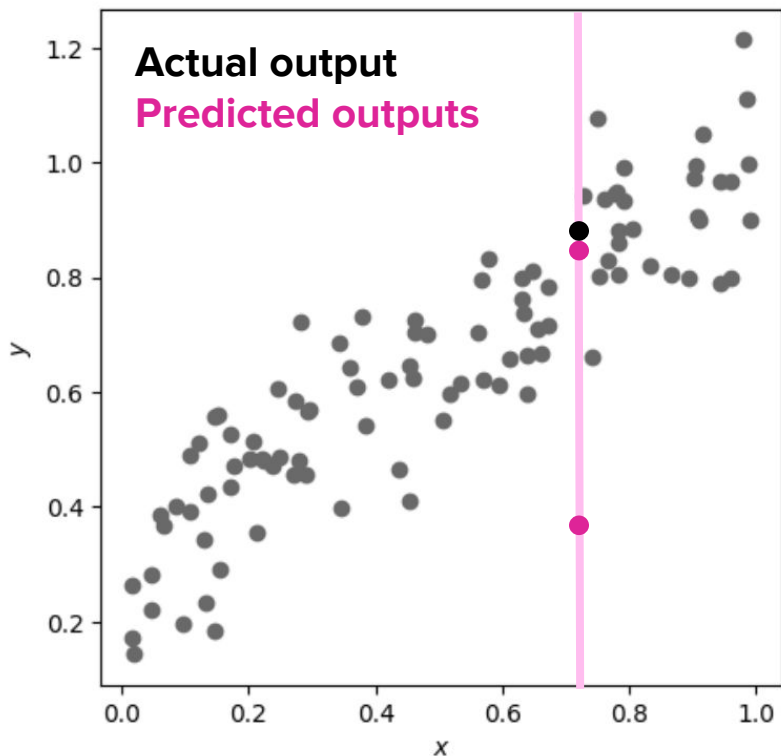
$E_{(x,y) \sim P}[l_w(x, y)]$ is small

Empirical Risk Minimization:

$$\operatorname{argmin}_w \frac{1}{n} \sum_{i=1}^n l_w(\vec{x}_i, y_i)$$

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n l_w(\vec{x}_i, y_i) = \operatorname{argmin}_w E_{(x,y) \sim P}[l_w(\mathbf{X}, \mathbf{y})]$$

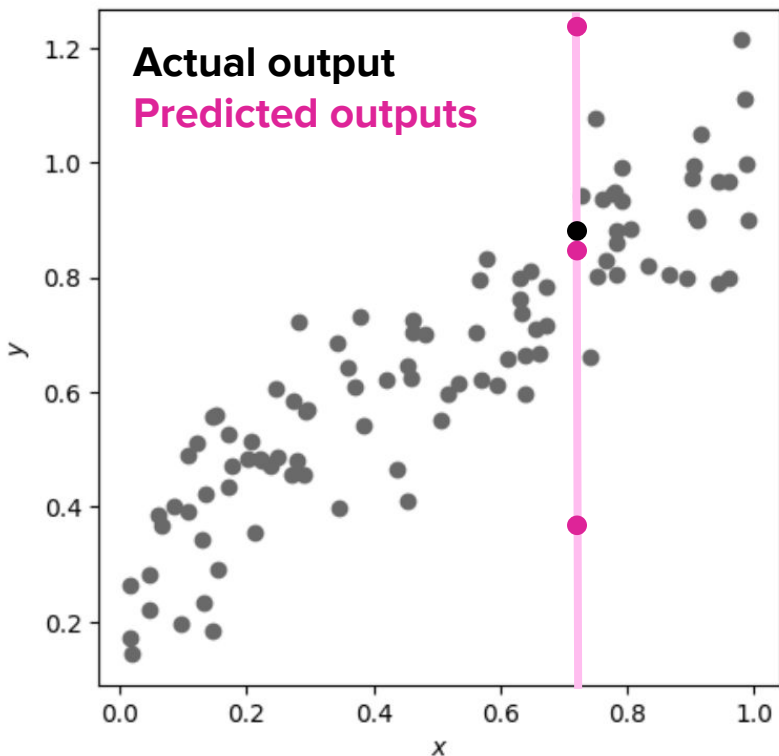
Restating the regression problem



Batch Learning, Restated:

- Given: A training set $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_k, y_k) \sim_{i.i.d} P$, and loss function $l_w(x, y)$
- Goal: $\operatorname{argmin}_w \frac{1}{n} \sum_{i=1}^n l_w(\vec{x}_i, y_i)$

The loss function defines your performance objective



Select loss function of

$$l_w(x_i, y_i) = (h(x_i) - y_i)^2$$

Where:

$h(x_i)$: output predicted by the model given the feature vector x_i , and
 y_i : is the true output

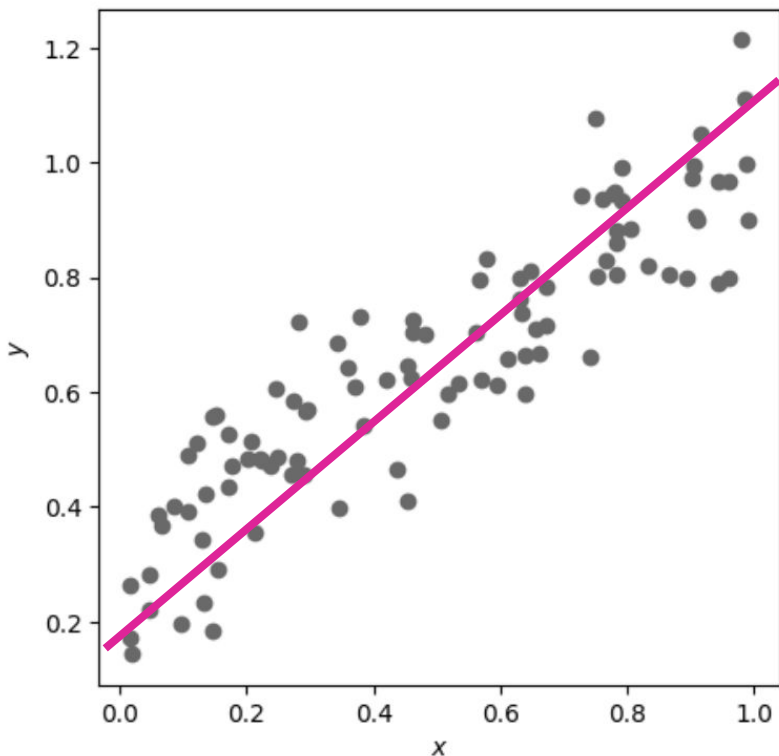
Then the expected loss is given by:

$$E[l_w] = \frac{1}{n} \sum_{i=1}^n \gamma_i^2$$

Where $\gamma_i = h(x_i) - y_i$ is the residual for sample (x_i, y_i)



The linear regression predictor hypothesis class



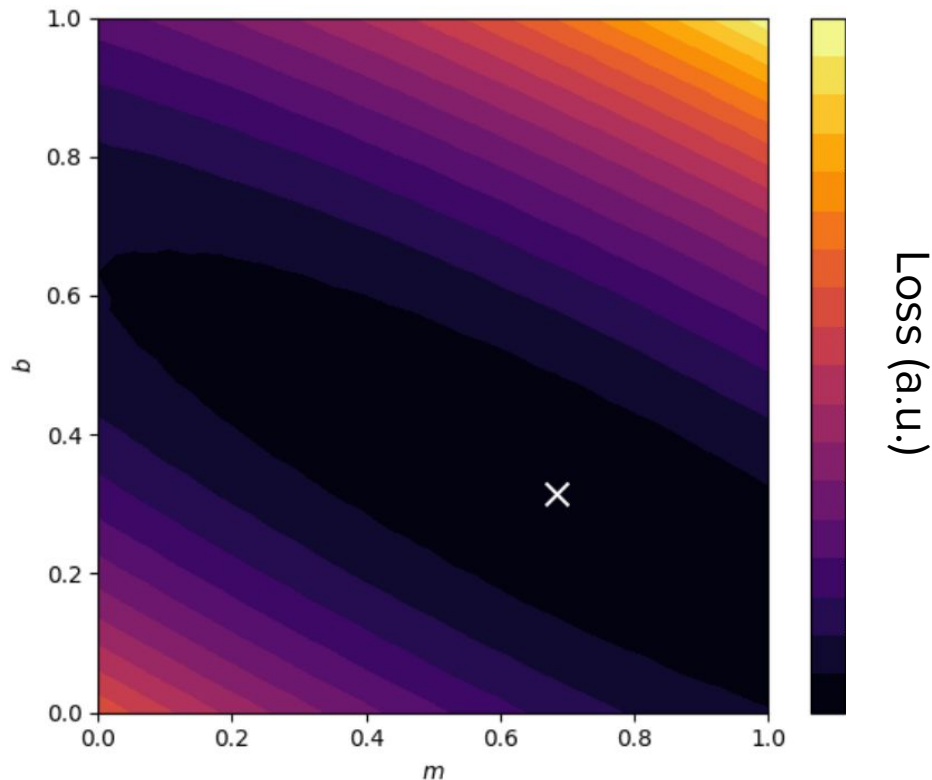
If we assume a model of the form $y = mx + b$, where m, b are parameters. Then

$$\begin{aligned} E[l_w] &= \sum_{i=1}^n \gamma_i^2 \\ &= \sum_{i=1}^n (h(x_i) - y_i)^2 \\ &= \sum_{i=1}^n (\langle (m, b), (x_i, 1) \rangle - y_i)^2 \\ &= \sum_{i=1}^n (\langle w, x'_i \rangle - y_i)^2 \end{aligned}$$

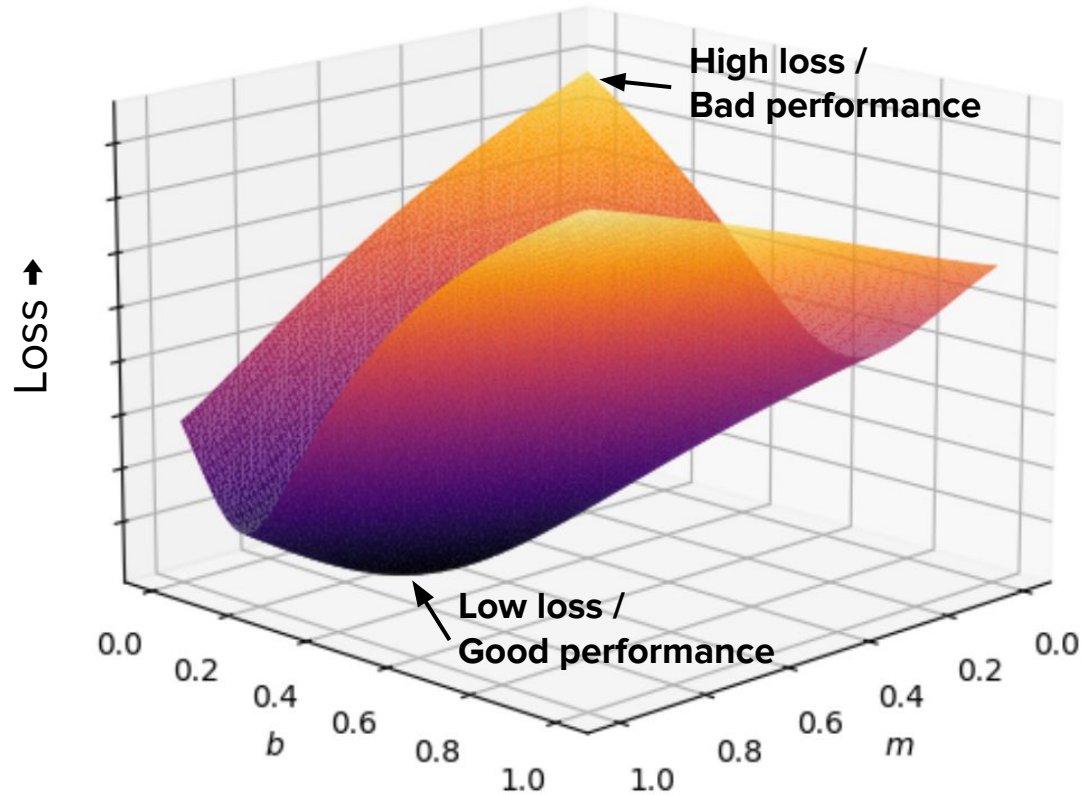


How to minimize the loss?

Brute force search suggests a unique set of parameters



The loss function surface and the gradient



Interlude: Calculus Review

Derivative:

Let $f(x) : \mathbb{R} \rightarrow \mathbb{R}$ be a scalar-valued function of one variable. Then

$f'(x) = \frac{df}{dx} : \mathbb{R} \rightarrow \mathbb{R}$, is the derivative of $f(x)$

Example: $f(x) = x^2 + 3x + 5$, then $f'(x) = 2x + 3$

Gradient:

Let $f(\vec{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ be a scalar-valued function of a d -vector. Then

$\nabla f(\vec{x}) = (\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_d}) : \mathbb{R}^d \rightarrow \mathbb{R}^d$, is the gradient of $f(\vec{x})$

Example: $f(\vec{x}) = 2x_1 + 3x_2 + 5x_3$, then $\nabla f(\vec{x}) = (2, 3, 5)$

Interlude: Calculus Review (Continued)

Hessian:

Let $f(\vec{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ be a scalar-valued function of a d -vector. Then

$\nabla^2 f(\vec{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$, is the Hessian of $f(\vec{x})$

$$\nabla^2 f(\vec{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix}$$

Example: $f(x) = 3x_1^2 + 2x_1x_2 + 5x_2^2$, then $\nabla^2 f(\vec{x}) = \begin{pmatrix} 6 & 2 \\ 2 & 10 \end{pmatrix}$

Equivalent notation of loss to leverage the gradient

Let $A, \mathbb{R}^{n \times (d+1)}$, be a matrix of the padded feature vectors in the training dataset,

$$A = \begin{bmatrix} \text{---} & x'_1 & \text{---} \\ \text{---} & x'_2 & \text{---} \\ & \vdots & \\ \text{---} & x'_n & \text{---} \end{bmatrix}$$

and $z, \mathbb{R}^{n \times 1}$, be a matrix of the outputs of the training dataset,

$$z = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

then we can construct a loss matrix, $L, \mathbb{R}^{n \times 1}$ as

$$L = \|Aw - z\|_2^2$$

How to find the solution?

Leveraging the gradient of the loss function

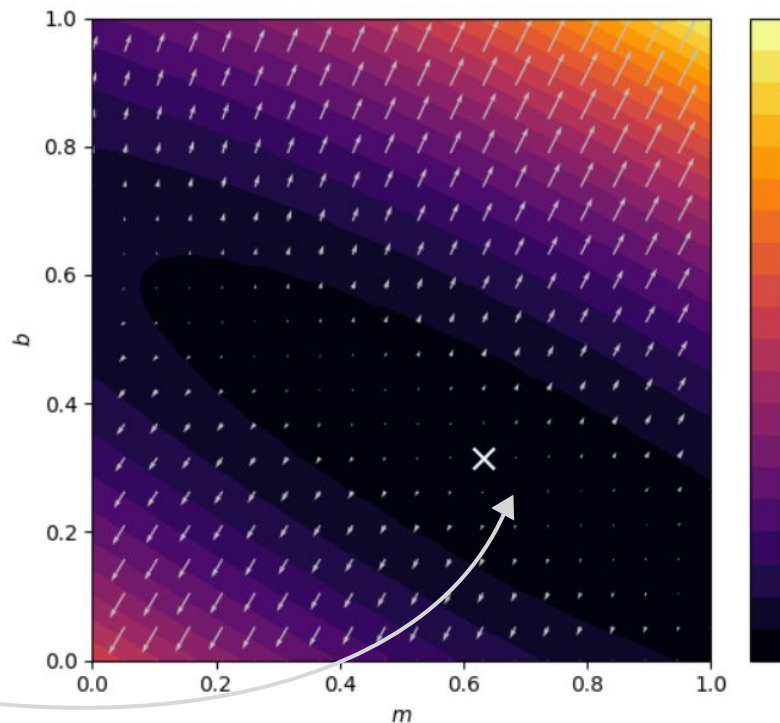
We are interested in the gradient of the loss with respect to the parameters w , that is $\nabla_w L$

$$\begin{aligned}\nabla_w L &= \nabla_w \|Aw - z\|_2^2 \\ &= \nabla_w (Aw - z)^T (Aw - z) \\ &= \nabla_w [w^T A^T Aw - 2z^T Aw + z^T z] \\ &= 2A^T Aw - 2A^T z\end{aligned}$$

The expression for the loss can be rearranged to solve for the parameters where the gradient is zero:

$$\begin{aligned}2A^T Aw - 2A^T z &= 0 \\ 2A^T Aw &= 2A^T z\end{aligned}$$

$$w = (A^T A)^{-1} A^T z$$



Practical issues with minimizing the loss function

- $A^T A$ might not be invertible
- $A^T A$ can be computationally intensive
- Could be imprecise if ill-conditioned
- Could also solve system of linear equations with gaussian elimination



Summary of our process to solve a regression problem

Define Performance

Specified a loss function in the statistical learning setting using empirical risk minimization.

Select a Model

Linear:

$$y = mx + b$$

Rewrite as:

$$y = \langle (m, b), (x, 1) \rangle$$

$$y = \langle w, x' \rangle$$

Estimate Parameters



Used brute force to find a high-performing solution (low loss)



Computed gradient and determined parameters where it is zero.



Lecture Outline

I. What's the basic process for solving regression?

Models, loss functions, and empirical risk minimization

II. What should one consider in loss function design?

Designing for optimization, stability and generalization

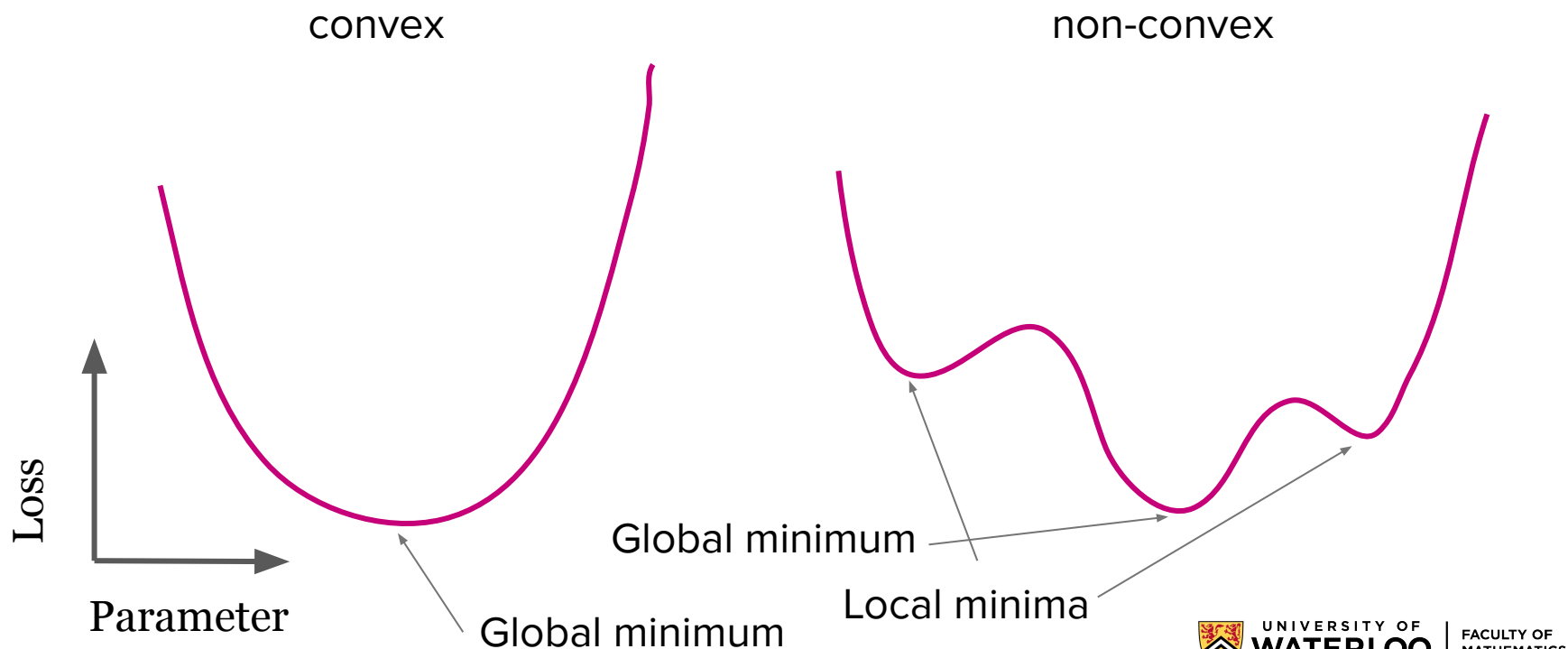
III. How do you evaluate model performance iteratively?

Overfitting, data splits, and cross validation

IV. Summary + Housekeeping



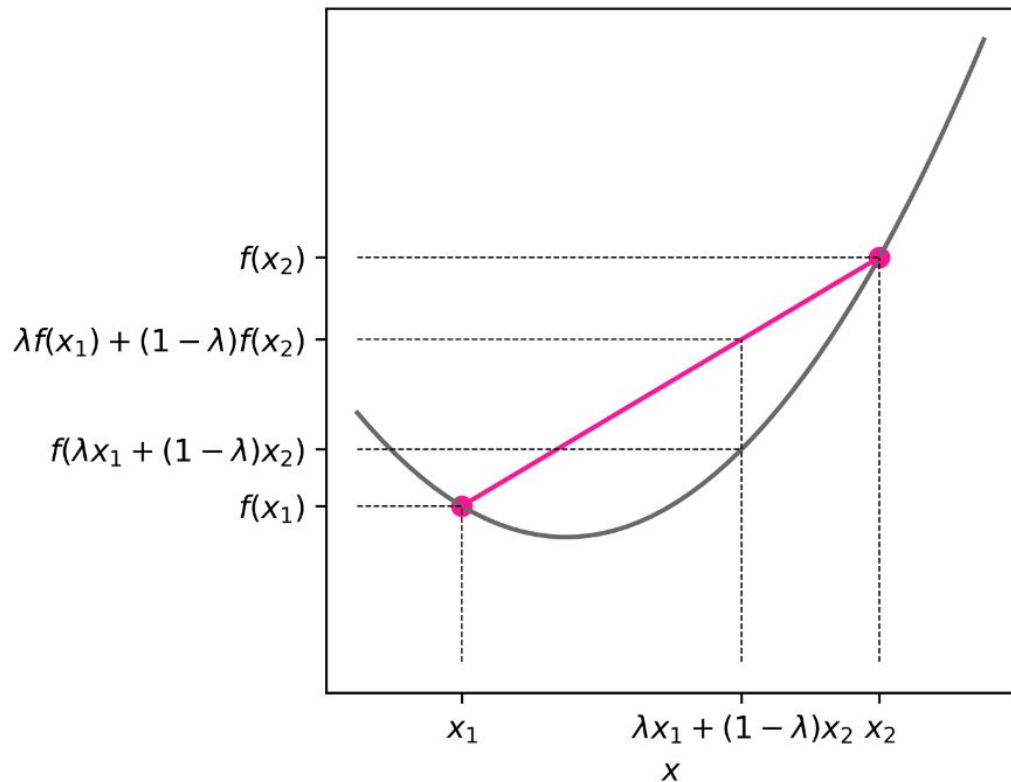
Can we be sure that the minimum is a global minimum?



Definition:
A function is convex iff it satisfies Jensen's Inequality

Jensen's Inequality:

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



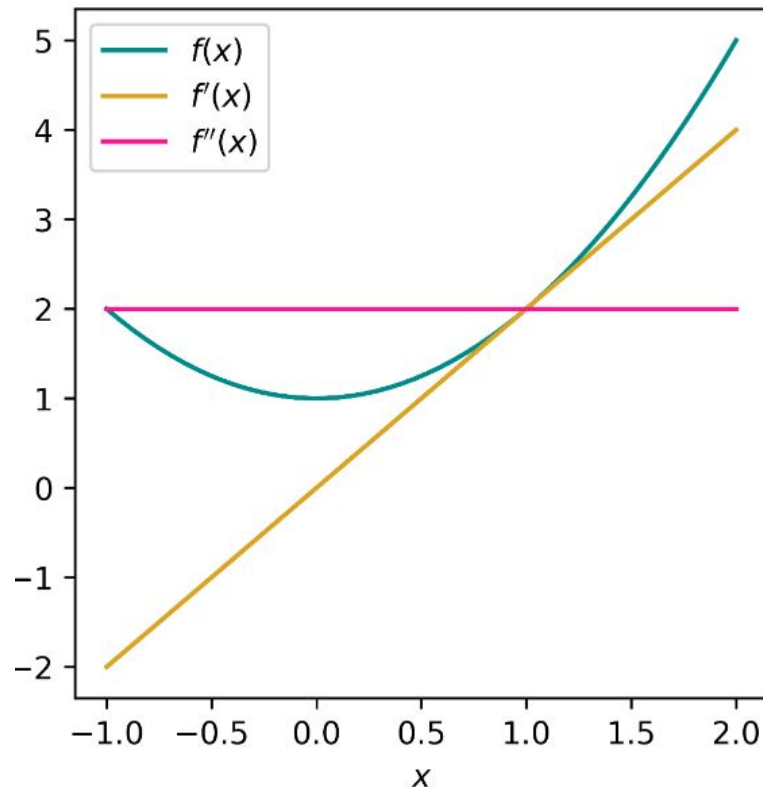
If a function is convex, its second derivative is strictly positive

For $x \in \mathbb{R} : f''(x) \geq 0 \forall x$

For $x \in \mathbb{R}^d : \nabla^2 f(x) \geq 0$

$$\implies v^T H v \geq 0$$

(Hessian is positive semidefinite)



Convex functions are straightforward to optimize

Fermat's condition:

If x is a local extremum of f , then $\nabla f(x) = 0$



If f is convex, the converse is also true:
 $\nabla f(x) = 0 \implies$ global extremum



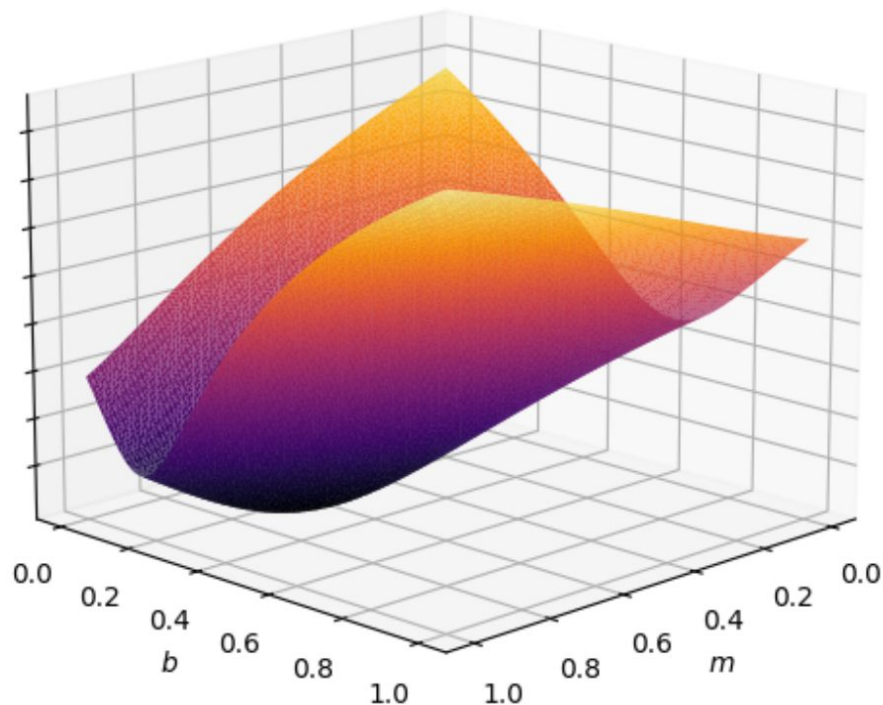
Is the least-squares loss convex?

Recall that the least-squares loss was defined as:

$$\begin{aligned}L &= \|Aw - z\|_2^2 \\ \nabla_w L &= 2A^T Aw - 2A^T z \\ \nabla_w^2 L &= 2A^T A\end{aligned}$$

Is the Hessian positive semidefinite?

$$\begin{aligned}2v^T A^T A v &\geq 0 \forall v \\ 2\|Av\|_2^2 &\geq 0 \forall v\end{aligned}$$



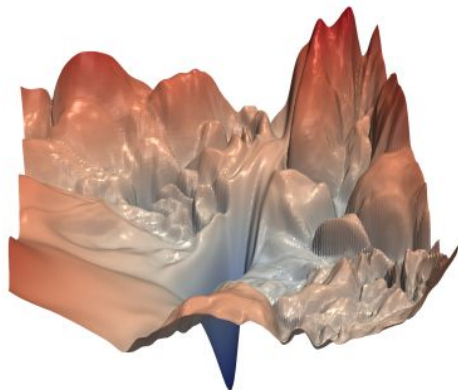
Visualizing the Loss Landscape of Neural Nets

Hao Li¹, Zheng Xu¹, Gavin Taylor², Christoph Studer³, Tom Goldstein¹

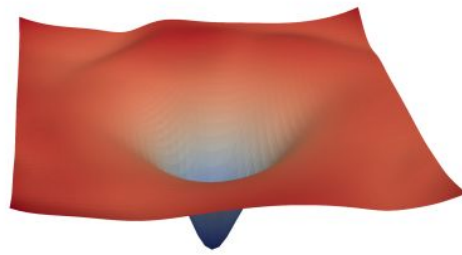
¹University of Maryland, College Park ²United States Naval Academy ³Cornell University
{haoli, xuzh, tomg}@cs.umd.edu, taylor@usna.edu, studer@cornell.edu

Abstract

Neural network training relies on our ability to find minima of non-convex loss functions. It is well-known that different network designs (e.g., skip connections) produce loss functions with different properties for chosen training parameters (batch size, learning rate, etc.) that generalize better. However, the reasons for these differences on the underlying loss landscape, and how they affect generalization, are not well understood. In this paper, we explore the structure of neural loss functions, and how training parameters affect the loss landscape, using a range of visualization methods. In particular, we use a “filter normalization” method that helps us make meaningful side-by-side comparisons between different loss landscapes. As a first step, we explore how network design affects the loss landscape, and how training parameters affect the



(a) without skip connections



(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Lecture Outline

I. What's the basic process for solving regression?

Models, loss functions, and empirical risk minimization

II. What should one consider in loss function design?

Designing for optimization, stability and generalization

III. How do you evaluate model performance iteratively?

Overfitting, data splits, and cross validation

IV. Summary + Housekeeping



Lecture Outline

I. What's the basic process for solving regression?

Models, loss functions, and empirical risk minimization

II. What should one consider in loss function design?

*Designing for optimization, **stability and generalization***

III. How do you evaluate model performance iteratively?

Overfitting, data splits, and cross validation

IV. Summary + Housekeeping



Stabilize the weights with L_2 Regularization

Ridge (Tikhonov) Regression:

$$\operatorname{argmin}_w \underbrace{\|Aw - z\|_2^2}_{\text{error}} + \underbrace{\lambda\|w\|_2^2}_{\text{loss}}$$

- Penalizes large weights
- Stabilizes the weights



Sparsify the weights with L_1 Regularization

Lasso Regression:

$$\operatorname{argmin}_w \|Aw - z\|_2^2 + \lambda \|w\|_1$$

- Penalizes non-zero weights
- Sparsifies the weights (many will be zero)
- Will cause many features to be ignored



Lecture Outline

I. What's the basic process for solving regression?

Models, loss functions, and empirical risk minimization

II. What should one consider in loss function design?

Designing for optimization, stability and generalization

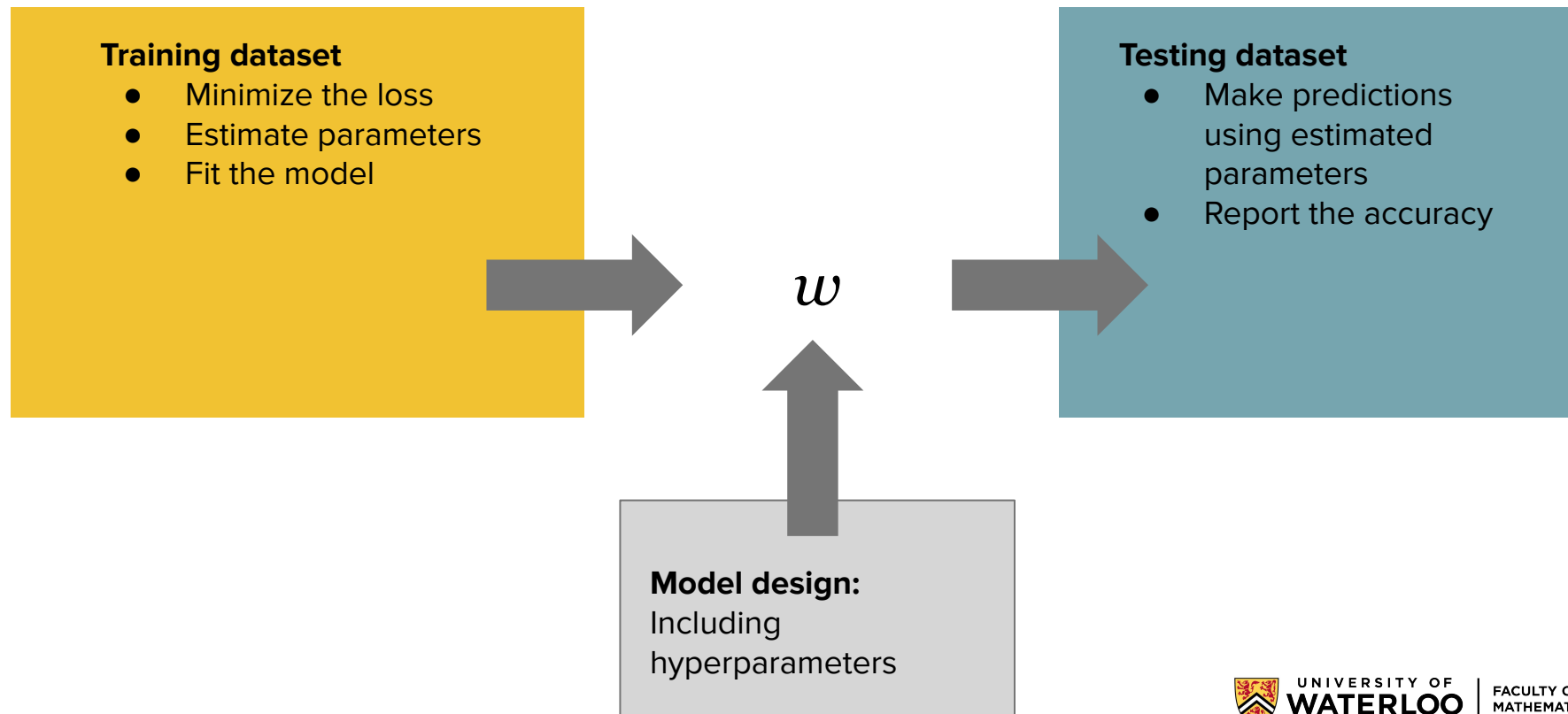
III. How do you evaluate model performance iteratively?

Overfitting, data splits, and cross validation

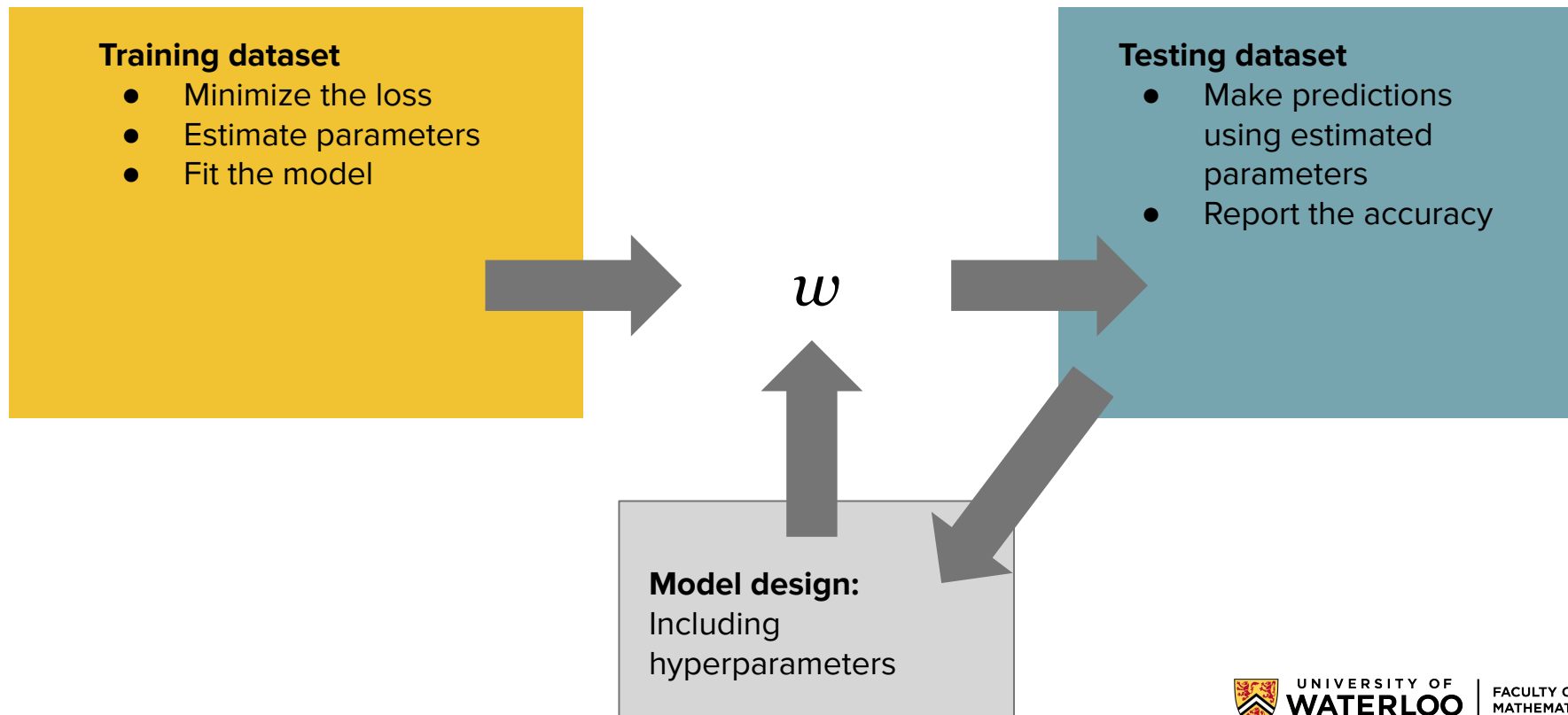
IV. Summary + Housekeeping



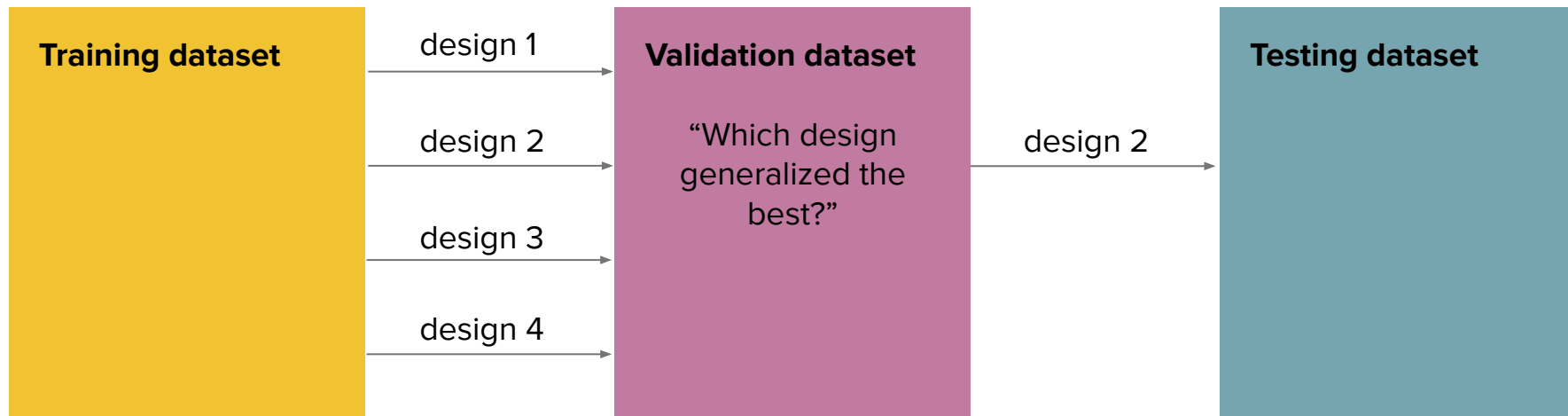
Correct use of training and testing datasets



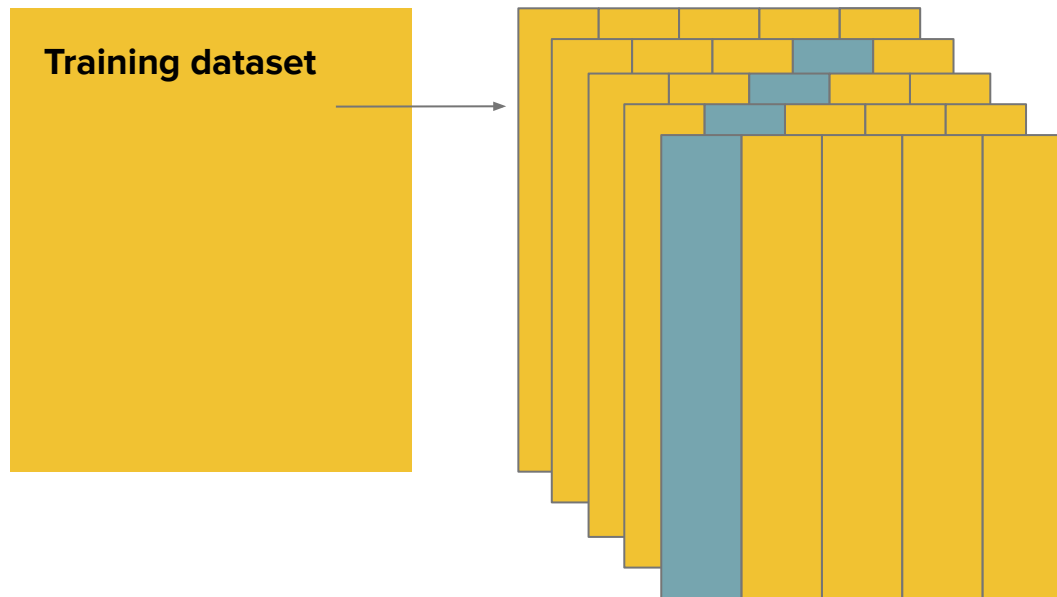
Incorrect use of training and testing datasets



Introducing a validation dataset



Cross-validation for small datasets



Algorithm 1 Cross-Validation.

Input:

- 1: Dataset $D = \{(x_i, y_i) \in \mathbb{R}^{d+1}\}$,
- 2: number of folds k

Output: Optimal hyperparameter λ

- 3: **for** $\lambda = \lambda_1, \lambda_2, \dots$ **do**
 - 4: **for** $i = 1, 2, \dots, k$ **do**
 $w_{\lambda, i} = \text{train}(\cup_{j \neq i}^k D_j)$
 $\text{Perf}_{\lambda, i} = \text{Acc}(D_i)$
 - 5: **end for**
 $\text{Perf}_{\lambda} = \frac{1}{k} \sum_{i=1}^k \text{Perf}_{\lambda, i}$
 - 6: **end for**
 - 7: **return** $\text{argmax}_{\lambda} \text{Perf}_{\lambda}$
-



Lecture Outline

I. What's the basic process for solving regression?

Models, loss functions, and empirical risk minimization

II. What should one consider in loss function design?

Designing for optimization, stability and generalization

III. How do you evaluate model performance iteratively?

Overfitting, data splits, and cross validation

IV. Summary + Housekeeping



Lecture Aims

At the end of the lecture, we should be able to:

- Write code to solve a simple regression problem numerically, given a dataset.
- Characterize and design loss functions using correct terminology and sound mathematical principles.
- Adhere to best practices for model evaluation and iterative improvement.

Questions and Errata from Perceptron Lecture

- Why do we need a bias?
- Why is an affine set defined as $wTx = -b$?
- Relationship of oracle to optimal separating hyperplane.



Errata

- On the slide titled “The Binary Classification Problem” (Slide 12), $\vec{x}_i \in \mathbb{R}^m$. This has been corrected to $\vec{x}_i \in \mathbb{R}^d$, for consistency with the remainder of the lecture.
- On the slide titled “The notation and meaning of inner product” (slide 16), the right-hand side of the graphical depiction has been changed from b to $-b$ as per the definition of an affine set.
- On the slide titled “The padding trick simplifies analysis” (Slide 30) The position of the 1 in the augmented feature vector did not line up with that of b in the augmented parameter vector. This has been fixed.
- On the slide titled “A few more properties” (Slide 49), there appeared a statement “No guarantee to be the oracle associated with the margin of the dataset”. This has been corrected to “No guarantee to be optimal, in terms of maximizing the margin”

Minor Changes

- In the slide titled “Exploring the divination effort” dataset (Slide 13), the color of data points have been changed for accessibility purposes.
- On the slide titled “From hyperplanes to halfspaces” (Slide 17), the text failed to appear on an earlier version of the slides. This has been fixed.
- Two slides were inserted prior to that titled “The Perceptron Algorithm in Action” (Slide 26) to provide necessary context on the video that was shown.
- The slide titled “Linear separability and the margin γ of a dataset” was moved to just after the explanation of padding.

Lecture	Date	Topics
0	05/09/2024	Introduction + Administrative Remarks
1	10/09/2024	Halfspaces the Perceptron Algorithm
2	12/09/2024	Linear Regression and Convexity
3	17/09/2024	Maximum Likelihood Estimation
4	19/09/2024	k-means Clustering
5	24/09/2024	k-NN Classification and Logistic Regression
6	26/09/2024	Hard-margin SVM
7	01/10/2024	Soft-margin SVM
8	03/10/2024	Kernel methods
9	08/10/2024	Decision Trees
10	10/10/2024	Bagging and Boosting
	15/10/2024	NO LECTURE - MIDTERM BREAK
	17/10/2024	NO LECTURE- MIDTERM BREAK
11	22/10/2024	Expectation Maximization Algorithm
12	24/10/2024	MLPs and Fully-Connected NNs
	29/10/2024	NO LECTURE - MIDTERM EXAM
13	31/10/2024	Convolutional Neural Networks
14	05/11/2024	Recurrent Neural Networks
15	07/11/2024	Attention and Transformers
16	12/11/2024	Graph Neural Networks (Time permitting)
17	14/11/2024	VAEs and GANs
18	19/11/2024	Flows
19	21/11/2024	Contrastive Learning (Time permitting)
20	26/11/2024	Robustness
21	28/11/2024	Privacy (Saber Malekmohammadi)
22	03/12/2024	Fairness