Announcements

- Homework 2 due **Tonight at 8pm**

- Midterm 1 is **Wednesday in class**
  - Example exam has been released

Projects

- **Teams**
  - Teams of 3, self-organized
  - We’ll allow teams of 2, but you’ll be graded the same way
- **Project options**
  - **Option 1 (computer vision):** Predict GPS coordinates from campus image
  - **Option 2 (NLP):** Predict source of news headline
- **Timeline**
  - Details released after Spring Break (3/17)
  - Milestone due mid April, final project due early May (end of semester)

Lecture 13: Review

CIS 4190/5190

Spring 2025

Concepts

- **Concepts**
  - Types of learning
  - Loss minimization
  - Bias-variance tradeoff
  - Maximum likelihood estimation
  - Non-parametric models

Algorithms

- **Algorithms**
  - What does the model family look like?
  - What does the loss function measure?
  - How does the optimizer work?
  - What is the effect of each design decision and hyperparameter on bias-variance tradeoff and/or optimization?

Concept: Types of Learning

- **Supervised learning**
  - Predict unknown output given a new input
  - Most common task
- **Unsupervised learning**
  - Infer structure in unlabeled data
  - Automatically learn features, visualize data, etc.
- **Reinforcement learning**
  - Sequential decision-making in unknown environment
  - Robotics, control, etc.

Concept: Loss Minimization View

- **Model family:** What are the candidate models f?
- **Loss function:** How to define “approximating”?
- **Optimizer:** How do we minimize the loss?

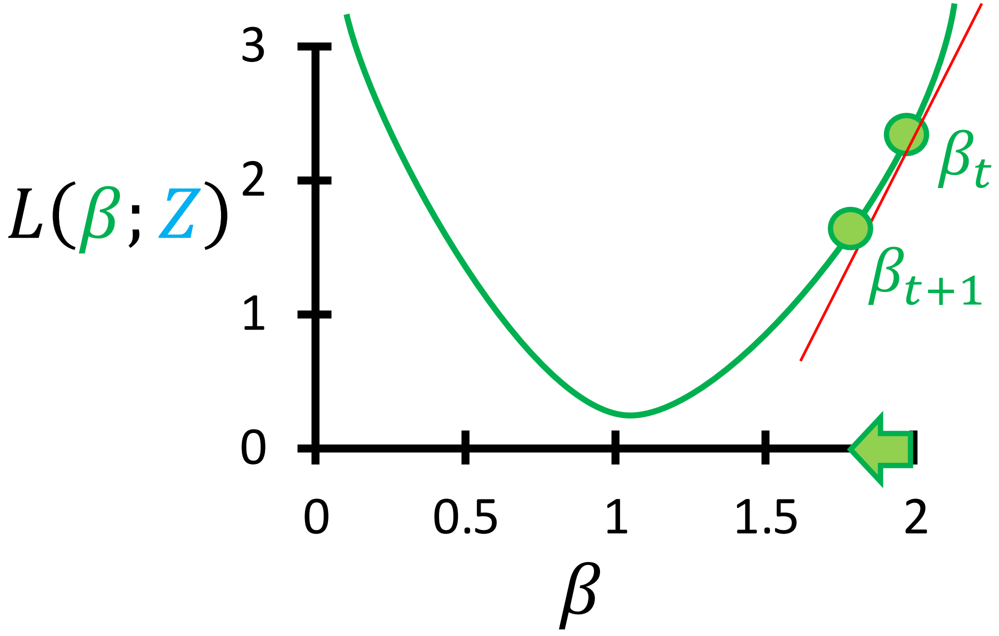
Algorithm: Linear Regression

- **Type:** Supervised learning
- **Model family:** Linear functions $f_\beta(x) = \beta^\top x$
- **Loss function:** MSE $L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2$
- **Optimizer:** Gradient descent
- **Hyperparameters:** Learning rate α, convergence threshold ϵ

Algorithm: Linear Regression

- Initialize $\beta_1 = \vec{0}$
- Repeat until $\|\beta_t - \beta_{t+1}\|_2 \leq \epsilon$:

$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)$$

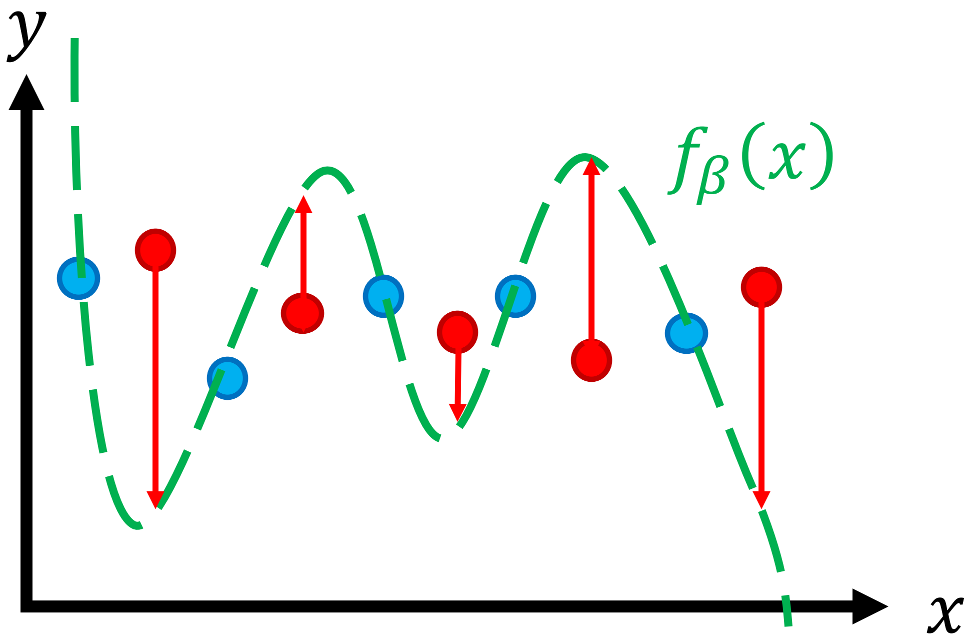


Algorithm: Linear Regression with Features

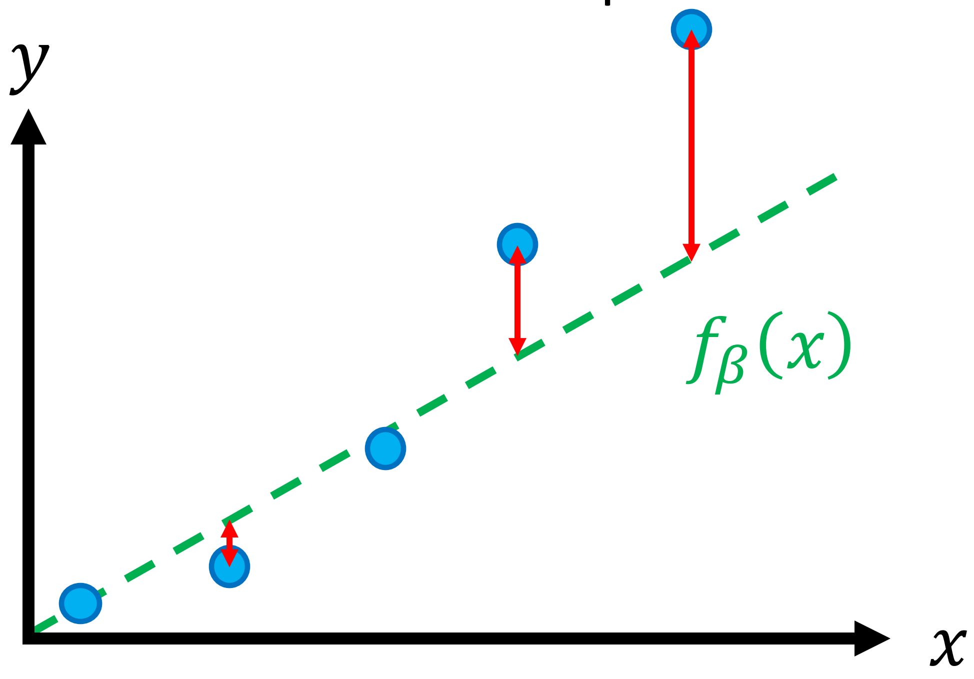
- **Type:** Supervised learning
- **Model family:** Linear functions $f_\beta(x) = \beta^\top \phi(x)$
- **Loss function:** MSE $L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \beta^\top \phi(x_i)\right)^2$
- **Optimizer:** Gradient descent
- **Hyperparameters:** Feature map ϕ

Concept: Bias-Variance Tradeoff

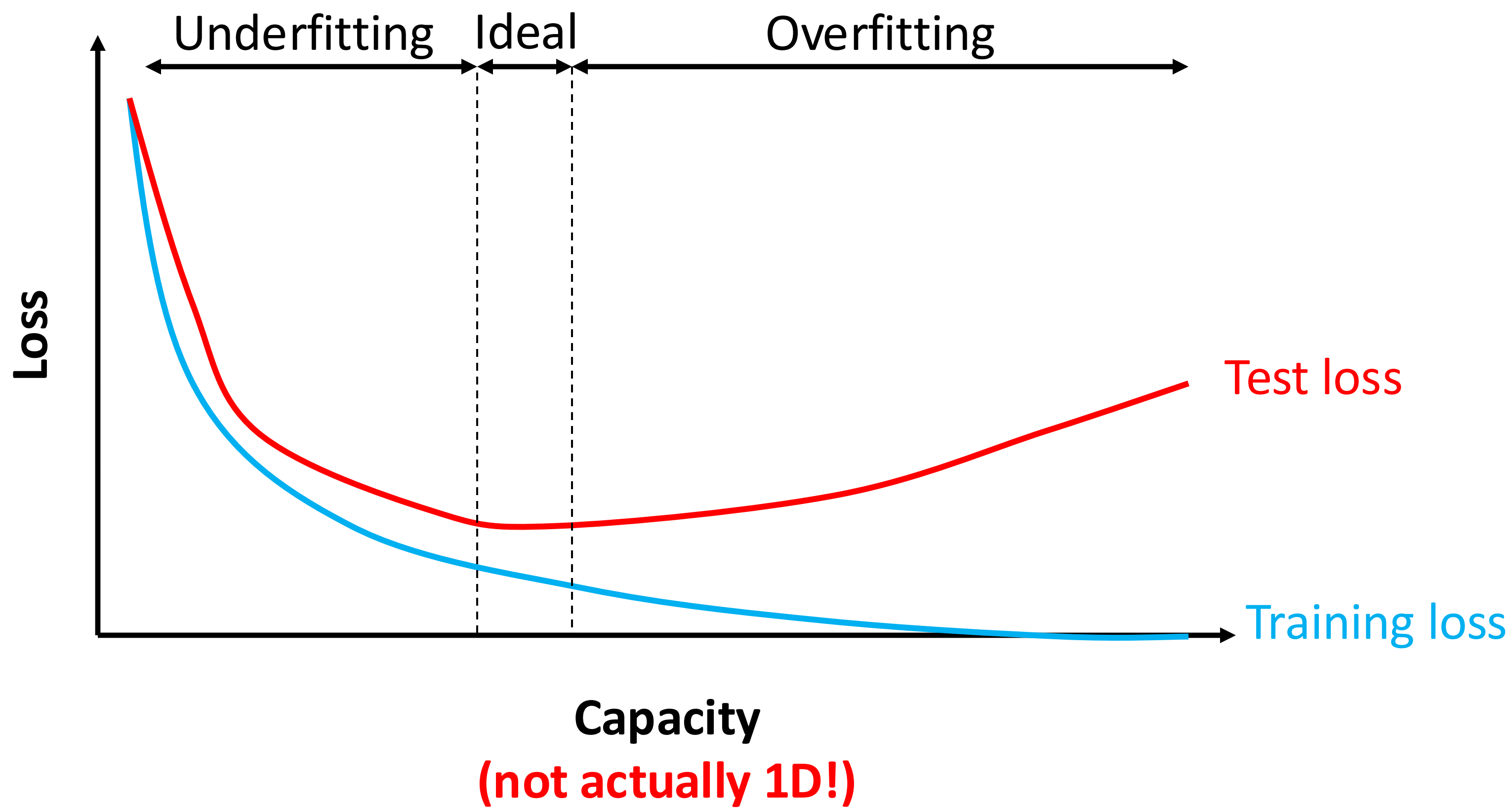
- **Overfitting (high variance)**
  - High capacity model capable of fitting complex data
  - Insufficient data to constrain it



- **Underfitting (high bias)**
  - Low capacity model that can only fit simple data
  - Sufficient data but poor fit

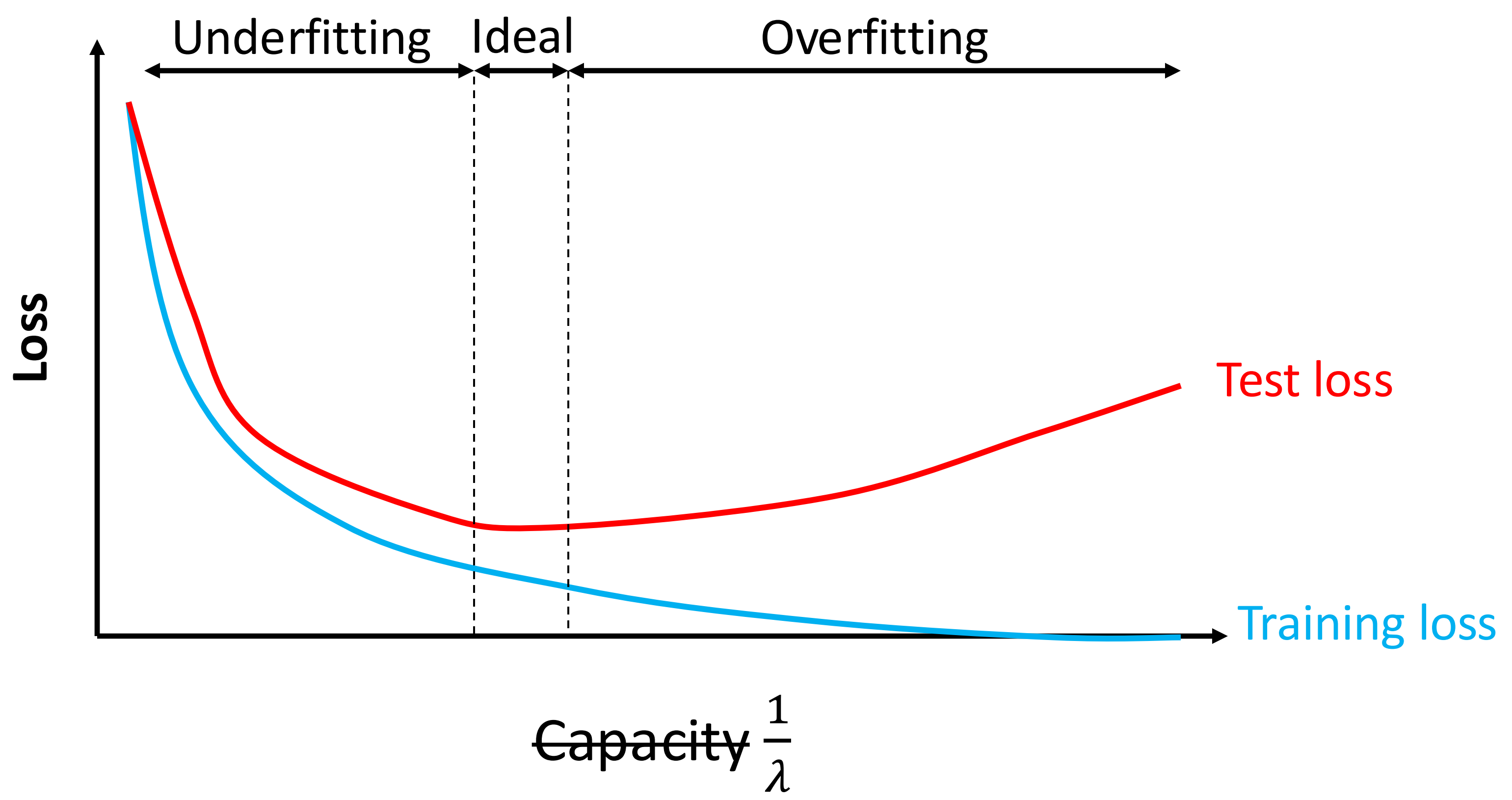


Concept: Bias-Variance Tradeoff



Slide by Padhraic Smyth, UCIrvine

Concept: Bias-Variance Tradeoff



Algorithm: L_2 Regularized Linear Regression

- **Type:** Supervised learning
- **Model family:** Linear functions $f_\beta(x) = \beta^\top x$
- **Loss function:** MSE $L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \cdot \|\beta\|_2^2$
- **Optimizer:** Gradient descent
- **Hyperparameters:** Regularization weight λ

Concept: Maximum Likelihood Estimation

- **Model family:** What is the likelihood $p(\,y \mid x\,)$?
- **Optimizer:** How do we minimize the negative log likelihood (NLL)?

Concept: Maximum Likelihood Estimation

- **Model family:** Most likely label

$$f_\beta(x) = \arg\max_y p_\beta(\, y \mid x \,)$$

- **Loss function:** Negative log likelihood (NLL)

$$\ell(\beta; Z) = -\sum_{i=1}^{n} \log p_\beta(\, y_i \mid x_i \,)$$

Algorithm: Linear Regression

- **Likelihood:** A Gaussian distribution

$$p_\beta(y \mid x) = N(y; \beta^\top x, 1) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(\beta^\top x - y)^2}{2}}$$

- **Optimizer:** Gradient descent

Algorithm: Linear Regression

- **Model family:**

$$f_\beta(x) = \beta^\top x$$

- **Negative log likelihood:**

$$\ell(\beta; Z) = \frac{n \log(2\pi)}{2} + \sum_{i=1}^{n} (\beta^\top x_i - y_i)^2$$

Algorithm: Logistic Regression

- **Likelihood:** Bernoulli distribution with

$$p_\beta(Y = 1 \mid x) = \frac{1}{1 + e^{-\beta^\top x}} = \sigma(\beta^\top x)$$

$$p_\beta(Y = 0 \mid x) = 1 - \sigma(\beta^\top x)$$

- **Optimizer:** Gradient descent

Algorithm: Logistic Regression

- **Model family:**

$$f_\beta(x) = 1(\beta^\top x \geq 0)$$

- **Negative log likelihood:**

$$\ell(\beta; Z) = -\sum_{i=1}^{n} y_i \log\big(\sigma(\beta^\top x_i)\big) + (1 - y_i) \log\big(1 - \sigma(\beta^\top x_i)\big)$$

Concept: Non-Parametric Models

- **Parametric models**
  - Model family has the form $\{ f_\beta \mid \beta \in \mathbb{R}^d \}$

- **Non-parametric models**
  - Very high capacity model families that can fit “arbitrary” functions
  - Specifically, parameter dimension grows with size of dataset Z
  - **Example:** For decision trees, # internal nodes is $O(|Z|)$

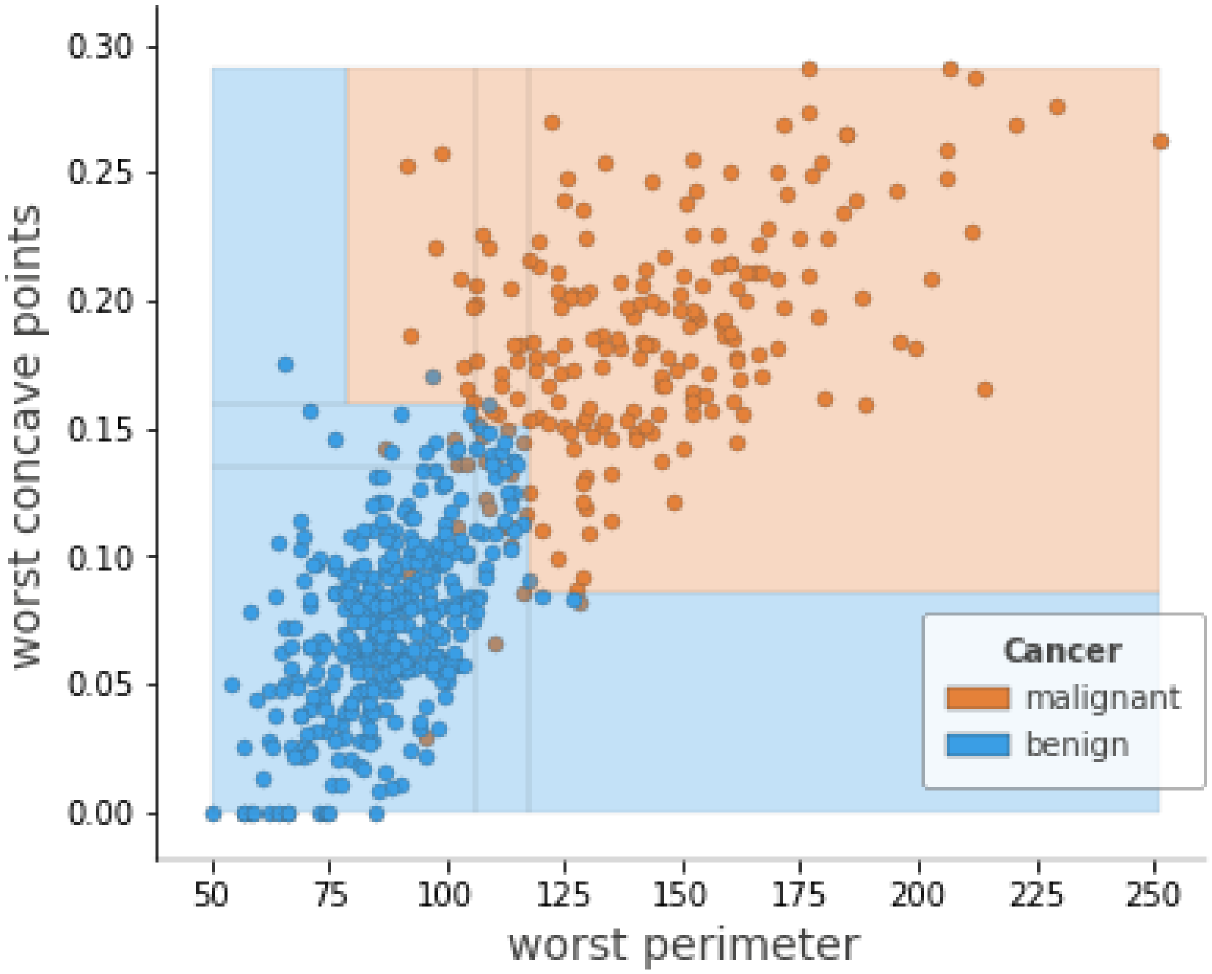
Algorithm: KNN

- **Type:** Supervised
- **Model family:** Aggregate labels of k nearest points
  - Parameters are the dataset ("nonparametric")
- **Loss function:** MSE, accuracy, etc.
- **Optimizer:** N/A
- **Hyperparameters:** Aggregation/distance functions, k

Algorithm: Decision Trees

- **Type:** Supervised
- **Model family:** Decision trees ($x_i = c$ for categorical, $x_i \leq t$ for real)
- **Loss function:** MSE, accuracy, etc.
- **Optimizer:** CART algorithm
  - Recursively choose nodes based on split that maximizes information gain
  - Early stopping (e.g., minimum gain) or prune using validation set
- **Hyperparameters:** Gain metric, maximum depth, minimum gain

Algorithm: Decision Trees



Algorithm: Neural Networks

- **Type:** Supervised, unsupervised
- **Model family:** Custom composition of parametric layers
  - Nonlinearities
  - Linear/fully-connected, convolution, pooling, recurrent, self-attention
- **Loss function:** Any differentiable loss
- **Optimizer:** Gradient descent (compute gradient via backpropagation)
  - **Tweaks:** Momentum, adaptive learning rates, schedules, residual connections, initialization, batch normalization, dropout, early stopping
  - Make sure you know how to take partial derivatives!