Supervised Learning & Linear Regression

EECE454 Intro. to Machine Learning Systems

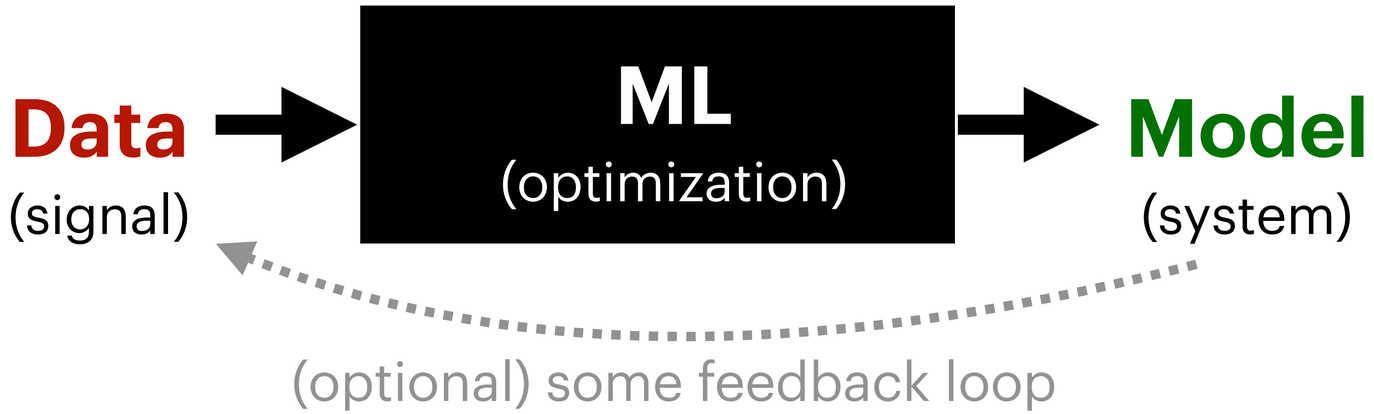
Fall 2024

Notice

- **Next week.** Video lectures
    - Will check attendance based on whether you watched
- **Attendance.** Checked electronically
- **Assignment#1.** Will be out during this weekend.
  (if I survive the conference deadlines)

Last class

- **System.** Linear Algebra
- **Optimization.** Matrix Calculus
- **Signals.** Probability and Statistics

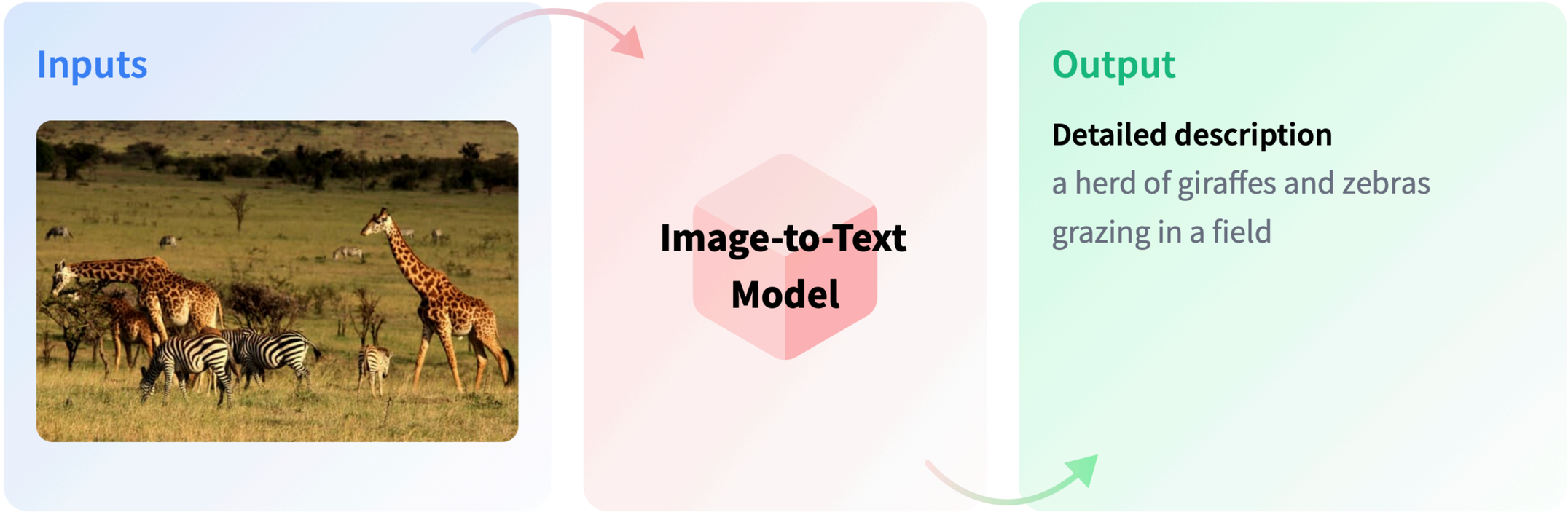


- **Today.** Start discussing classic ML algorithms
  - Basic framework of Supervised Learning
  - Simplest case: Linear Regression

A basic framework & supervised learning

Setup

- **Goal (general).** Given some input X, predict some output Y
  - Assumption. There is some (unknown-to-us) joint distribution P_{XY}
  - Example:



Source: HuggingFace

Setup

- **Goal (general).** Given some input X, predict some output Y
  - Assumption. There is some (unknown-to-us) joint distribution P_{XY}
- Roughly, two approaches: (c.f. Leo Breiman, "Statistical Modeling: The Two Cultures," 2001)
  - Algorithmic Modeling. Find a function $f(\cdot)$ such that, under P_{XY},
    it is likely to hold that $f(X) \approx Y$
    - Easier, in most cases
  - Data Modeling. Approximate the distribution $P_{Y|X}$ (often by approximating P_{XY} or $P_{X|Y}$)
    so that we can build various estimates based on it
    - Can do more in-depth analysis, such as uncertainty quantification

Setup

- **Goal (general).** Given some input X, predict some output Y
  - Assumption. There is some (unknown-to-us) joint distribution P_{XY}
- Roughly, two approaches: (c.f. Leo Breiman, "Statistical Modeling: The Two Cultures," 2001)
  - Algorithmic Modeling. Find a function $f(\cdot)$ such that, under P_{XY}, it is likely to hold that $f(X) \approx Y$
    - Easier, in most cases
  - Data Modeling. Approximate the distribution $P_{Y|X}$ (often by approximating P_{XY} or $P_{X|Y}$) so that we can build various estimates based on it
    - Can do more in-depth analysis, such as uncertainty quantification

We follow mostly this (cover data modeling later)

Setup (Algorithmic Modeling)

- **Goal (Rough).** Find a function $f(\cdot)$ such that, under P_{XY}, it is likely to hold that $f(X) \approx Y$
  - More precisely, we want to solve

$$\min_{f \in \mathcal{F}} \mathbb{E}_{P_{XY}}[\ell(f(X), Y)]$$

for some nice loss function $\ell(\cdot, \cdot)$ and a good set of predictors $\mathcal{F}$ (called hypothesis space)

Setup (Algorithmic Modeling)

- **Goal (Rough).** Find a function $f(\cdot)$ such that, under P_{XY}, it is likely to hold that $f(X) \approx Y$
  - More precisely, we want to solve

$$\min_{f \in \mathcal{F}} \mathbb{E}_{P_{XY}}[\ell(f(X), Y)]$$

  for some nice loss function $\ell(\cdot, \cdot)$ and a good set of predictors $\mathcal{F}$

- **Problem.** We do not know the true data-generating joint distribution P_{XY}
  - If we knew, we can simply choose the Bayes-optimal predictor.
  - Solution. We use training data to replace P_{XY}

Supervised Learning

- **Dataset.** In supervised learning, we assume that our training dataset consists of ***input-output*** pairs
  - That is, we have

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

  - Also called feature-label pairs.

Supervised Learning

- **Dataset.** In supervised learning, we assume that our training dataset consists of ***input-output*** pairs
  - That is, we have

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

  - Also called feature-label pairs.
  - Example. ImageNet dataset.

n02097047 (196)

n01682714 (40)

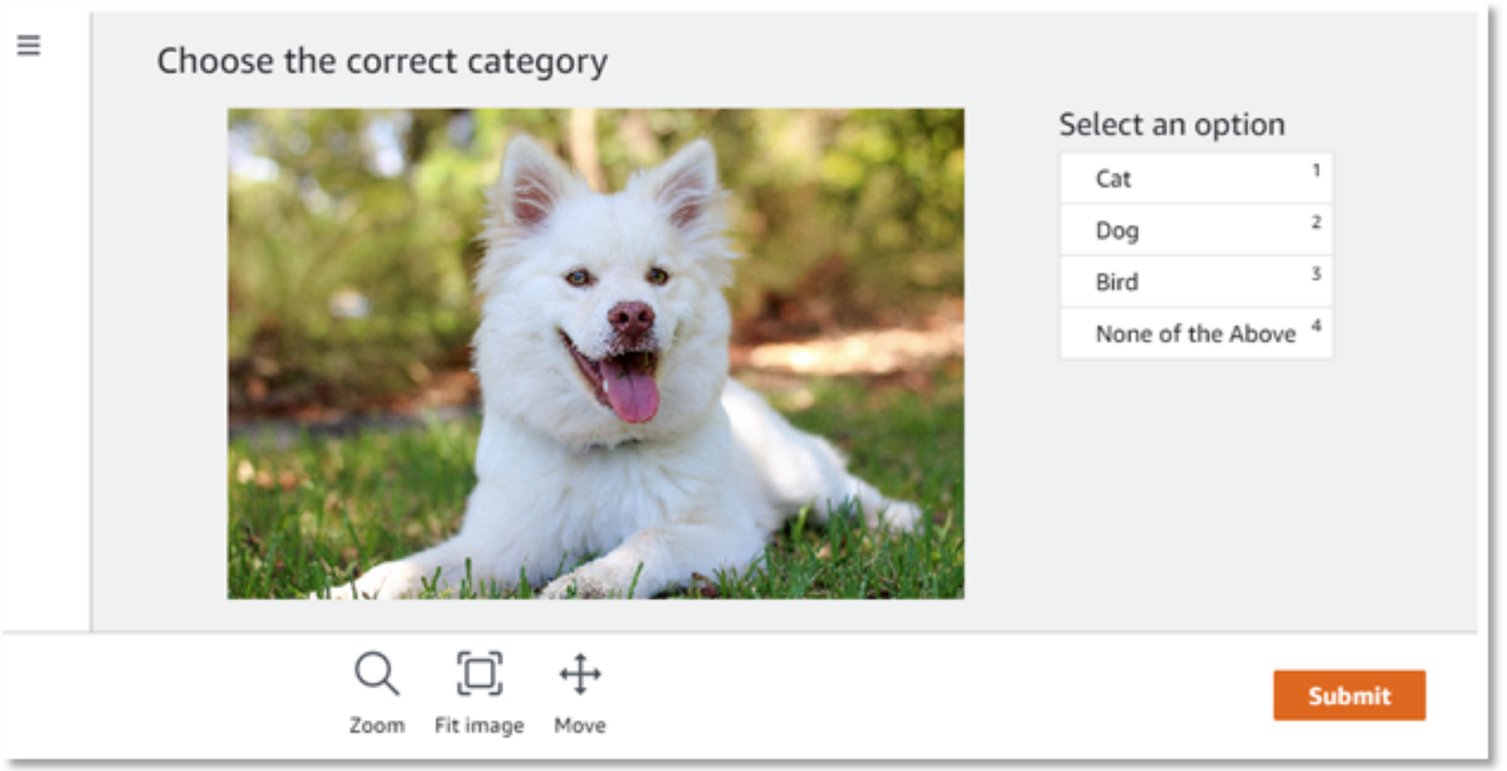
n03134739 (522)

imagenet1000_clsidx_to_labels.txt

```
{0: 'tench, Tinca tinca',
 1: 'goldfish, Carassius auratus',
 2: 'great white shark, white shark, man-eater, ma
 3: 'tiger shark, Galeocerdo cuvieri',
 4: 'hammerhead, hammerhead shark',
 5: 'electric ray, crampfish, numbfish, torpedo',
 6: 'stingray',
 7: 'cock',
 8: 'hen',
 9: 'ostrich, Struthio camelus',
 10: 'brambling, Fringilla montifringilla',
 11: 'goldfinch, Carduelis carduelis',
 12: 'house finch, linnet, Carpodacus mexicanus',
 13: 'junco, snowbird',
 14: 'indigo bunting, indigo finch, indigo bird, P
 15: 'robin, American robin, Turdus migratorius',
 16: 'bulbul',
 17: 'jay',
 18: 'magpie',
 19: 'chickadee',
```

Supervised Learning

- **Collection.**
  - Hire human annotators
    - e.g., Amazon MTurk
  - Crawl human-generated data
    - e.g., Image Captions
  - Utilize “very good” models
  - Synthetic data generation



- In a sense, human has provided supervision for the machine (thus called supervised learning)

Supervised Learning

- Given this dataset, we perform the **empirical risk minimization**

$$\min_{f \in \mathcal{F}} \mathbb{E}_{P_n}[\ell(f(X), Y)] \quad = \quad \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \quad \text{(+ regularizers)}$$

  - Intuition. The law of large numbers:

$$\frac{1}{n} \sum_{i=1}^{n} g(X_i) \quad \longrightarrow \quad \mathbb{E}_{P_X}[g(X)]$$

$$\frac{1}{n} \sum_{i=1}^{n} \ell(f(X_i), Y) \quad \longrightarrow \quad \mathbb{E}_{P_{XY}}[\ell(f(X), Y)]$$

    - Requires assuming that (x_i, y_i) are drawn i.i.d. from P_{XY}

Supervised Learning

$$\frac{1}{n}\sum_{i=1}^{n}\ell(f(X_i),Y) \quad \longrightarrow \quad \mathbb{E}_{P_{XY}}[\ell(f(X),Y)]$$

- Before we proceed, take some time to think about…
  - How fast would this convergence be?
    - Hint: Concentration inequalities
  - Would it be optimal to treat all data equally, e.g., weigh by 1/n?
    - Hint: Think about very rare cases

Testing

- **Problem.** We hope that $\mathbb{E}[\ell(Y, \hat{f}(X))]$ is small... but how do we know if we succeeded?

Testing

- **Problem.** We hope that $\mathbb{E}[\ell(Y, \hat{f}(X))]$ is small... but how do we know if we succeeded?

- Answer. We usually keep some data as a test dataset $D^{\text{test}} = \{(\tilde{x}_1, \tilde{y}_1), \ldots, (\tilde{x}_k, \tilde{y}_k)\}$

  - We validate that the test loss is small

$$\frac{1}{k} \sum_{i=1}^{k} \ell(\hat{f}(\tilde{x}_i), \tilde{y}_i)$$

  - Typically, we split the whole data into **train/val/test** with the 8:1:1 ratio (or 7:1:2, in the past)

    - If the dataset is small, consider cross-validation (not covered today)

Considerations in selecting ML algorithms

Which algorithm should we use?

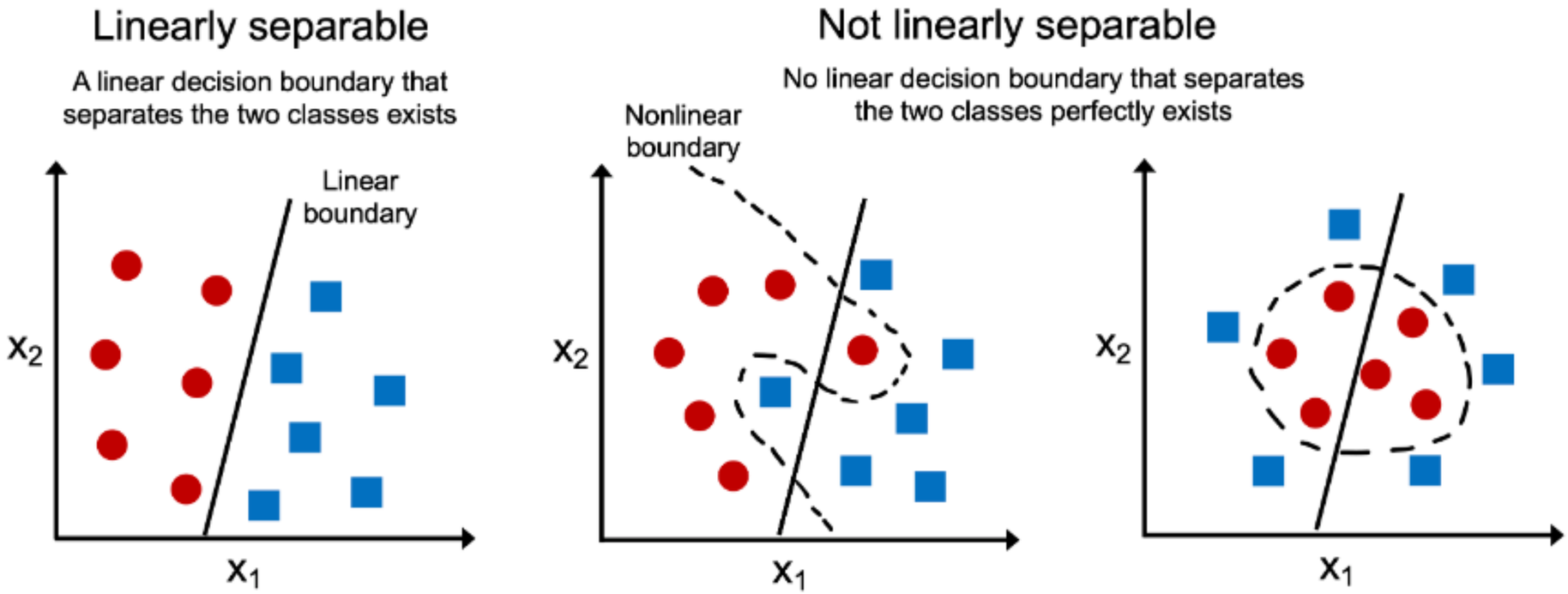
$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \quad \text{(+ regularizers)}$$

- Basically about designing the components of this optimization formula

Which algorithm should we use?

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \quad \text{(+ regularizers)}$$

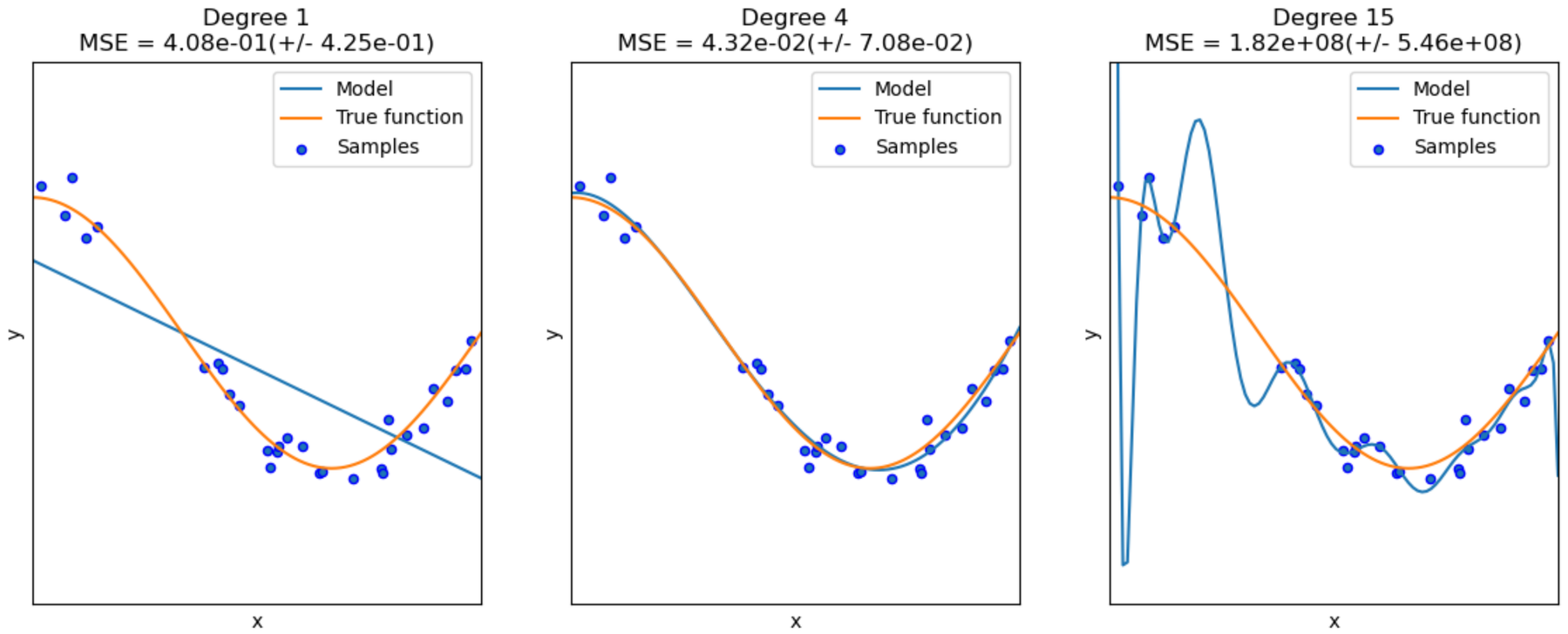
- **Model Size** (= Richness of hypothesis space $\mathcal{F}$)
  If too small, even the best $\hat{f}(\cdot)$ cannot fit the reality well.



Which algorithm should we use?

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \quad \text{(+ regularizers)}$$

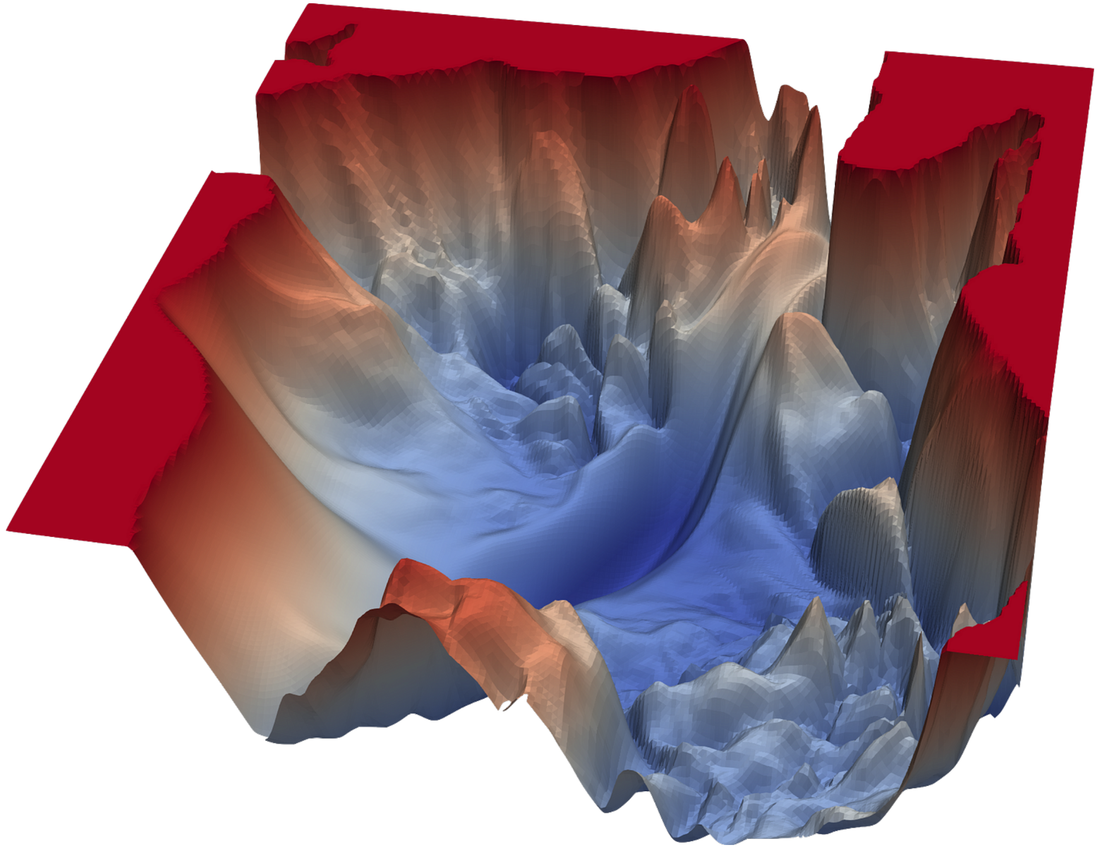
- **Model Size** (= Richness of hypothesis space $\mathcal{F}$)
  If too large, can overfit the training data + large inference cost



Which algorithm should we use?

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \quad \text{(+ regularizers)}$$

- **Optimization** (= difficulty of solving ERM)
  - Often highly customized for each model class
  - For highly complicated, nonlinear models …
    - Explicit solution not available
    - Takes a long time to compute the optimum (high training cost)



Which algorithm should we use?

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \quad \text{(+ regularizers)}$$

- **Loss function / Regularizer**
  - Affects how difficult the optimization is
    - e.g., non-continuous loss
  - Affects overfitting
    - e.g., soft penalty to overfitting
  - Affects desirable properties
    - e.g., robustness, sparsity

Throughout the course

- We study popular ML models one-by-one
- Try to clearly understand…
  - Which hypothesis space it uses
  - Which optimizer it uses
  - Which loss / regularizer it uses
- **This and next class.** Linear models, Naïve Bayes, Nearest Neighbors

- Note. Many of these choices heavily depend on task.
  - e.g., regression vs. classification, image vs. text vs. tabular, …

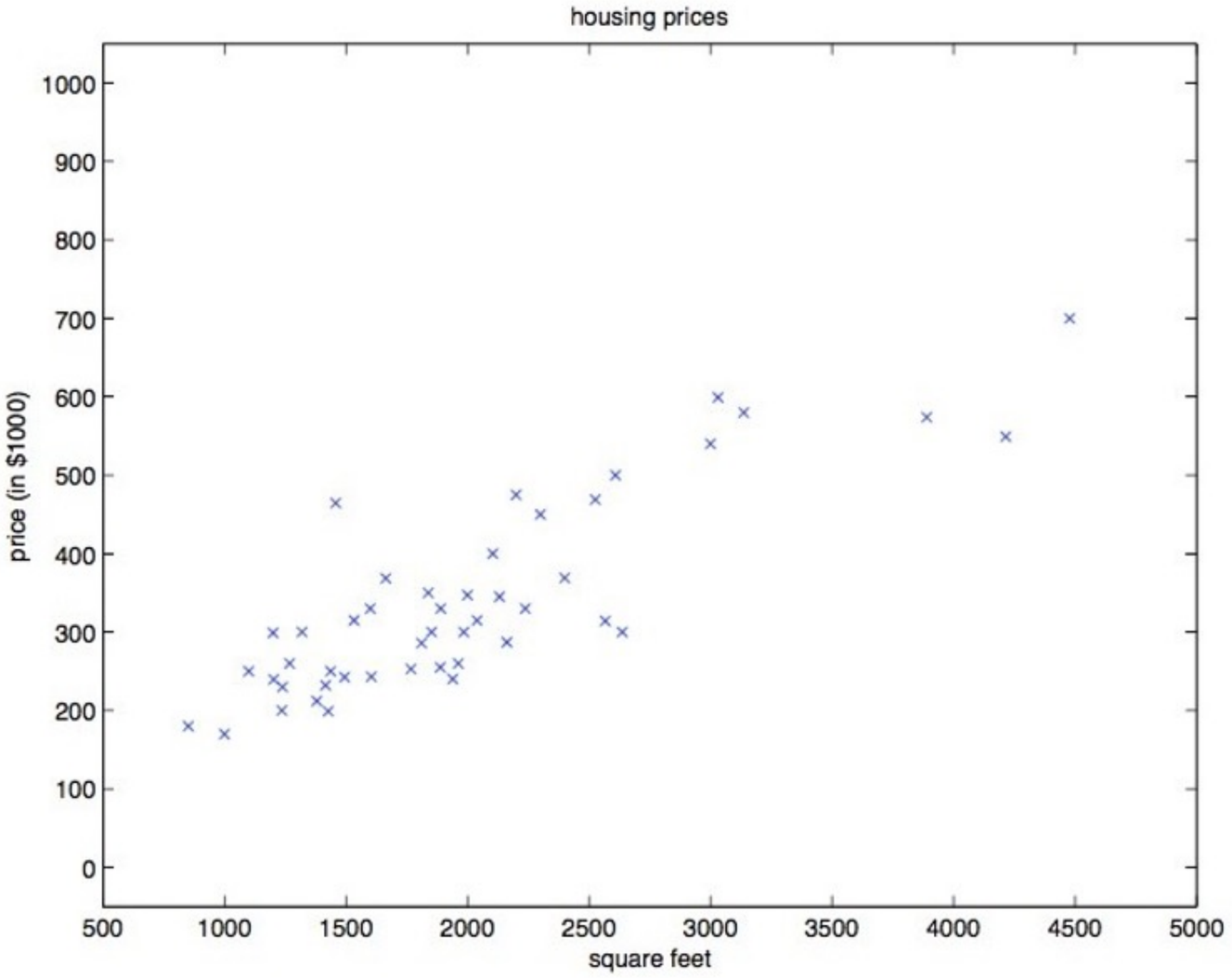
Linear Regression

Linear Regression

- **Goal.** Model the relationship between several continuous variables
  - Input $x \in \mathbb{R}^d$ and output $y \in \mathbb{R}^m$
  - Example. House price prediction

$$f(\text{area}) = \text{price}$$

Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



Linear Regression

- **Model.** We use a linear model $f(\cdot)$
  - If $x \in \mathbb{R}$ and $y \in \mathbb{R}$,

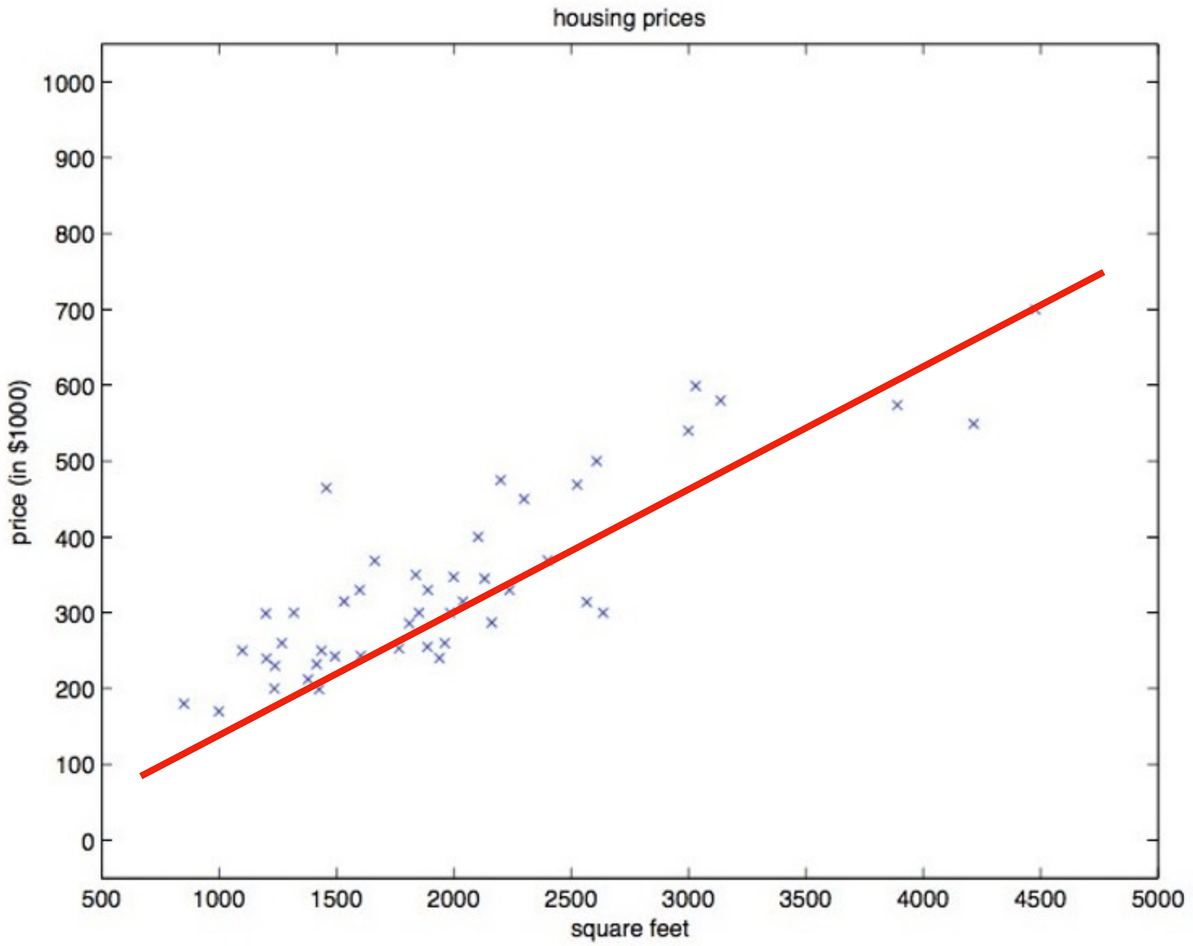
    $$f(\mathbf{x}) = w \cdot x + b, \qquad w \in \mathbb{R}, c \in \mathbb{R}$$

  - If $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$,

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

  - If $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^m$,

    $$f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \qquad \mathbf{W} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m$$



Linear Regression

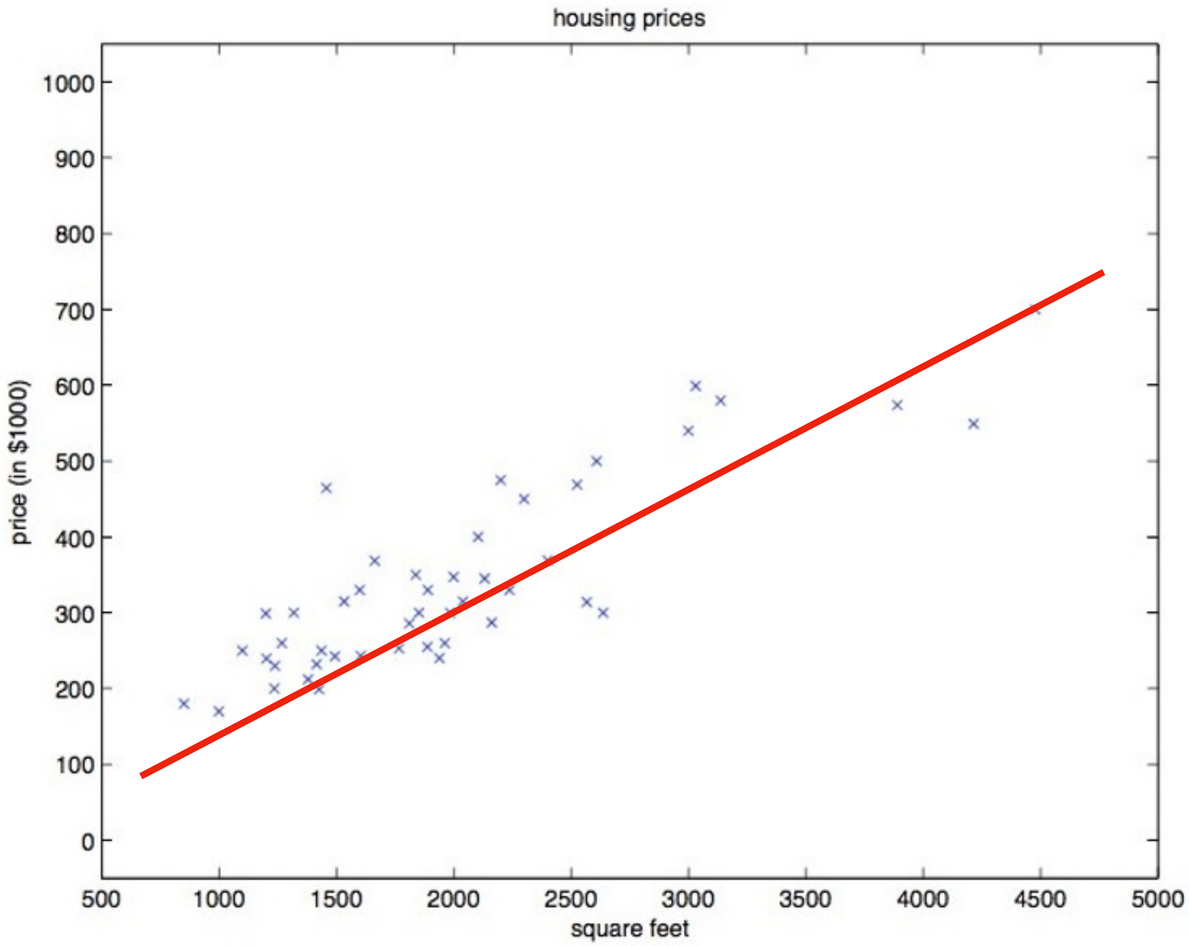
- **Model.** We use a linear model $f(\cdot)$
  - If $x \in \mathbb{R}$ and $y \in \mathbb{R}$,

    $f(\mathbf{x}) = w \cdot x + b, \qquad w \in \mathbb{R}, c \in \mathbb{R}$
  - If $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$,

    $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \qquad \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$
  - If $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^m$,

    $f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}, \qquad \mathbf{W} \in \mathbb{R}^{m \times d}, \mathbf{b} \in \mathbb{R}^m$

Our hypothesis space
(parameter space, model space)



Linear Regression

- **Loss.** We will use the squared ℓ_2 loss, i.e., $\ell(\hat{\mathbf{y}}, \mathbf{y}) = \|y - \hat{y}\|_2^2$
  - Known as ordinary least squares
- For a dataset $D = \{(x_i, y_i)\}_{i=1}^n$, we are solving

$$\min_{w,b} \frac{1}{2n} \sum_{i=1}^{n} \Big(y_i - (w \cdot x_i + b) \Big)^2$$

Linear Regression

- **Loss.** We will use the squared ℓ_2 loss, i.e., $\ell(\hat{\mathbf{y}}, \mathbf{y}) = \|y - \hat{y}\|_2^2$
  - Known as ordinary least squares
- For a dataset $D = \{(x_i, y_i)\}_{i=1}^n$, we are solving

$$\min_{w,b} \frac{1}{2n} \sum_{i=1}^{n} \Big(y_i - (w \cdot x_i + b) \Big)^2$$

- **Question.** Why least squared?
  - Easy to solve
    - Quadratic function
  - Nice interpretation
    - Maximum likelihood estimate under Gaussian noise (talk about this later)

Linear Regression

- **Fun fact.** If X and Y are jointly Gaussian random variables,
  we know that the MMSE estimator is always linear
  - Thus linear models are a sufficiently rich hypothesis space for such data
    - No underfitting expected
  - Proof. Homework!

Linear Regression: Optimization (or Training)

1D, bias-free case

$$\min_{w \in \mathbb{R}} \underbrace{\frac{1}{2n} \sum_{i=1}^{n} \Big(y_i - (w \cdot x_i) \Big)^2}_{=: J(w)}$$

- This is a quadratic function.
  - The minimum is where derivatives are zero (critical point)

$$\frac{\partial J}{\partial w}(w) = 0$$

1D, bias-free case

$$\frac{\partial J}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}(w \cdot x_i - y_i)x_i = 0 \quad \Rightarrow \quad w\left(\sum x_i^2\right) = \sum y_i x_i$$

$$\Rightarrow \quad w = \frac{\sum y_i x_i}{\sum x_i^2}$$

- We can find an explicit formula for the critical point
  - Not always possible
    - What if we used $\ell(\hat{y}, y) = (y - \hat{y})^6$?
  - No gradient computation needed, luckily
  - Needs several multiplications & summations for optimization (i.e., training)

Multivariate case

- Consider a slightly more general case of $\mathbf{x} \in \mathbb{R}^d$, $y \in \mathbb{R}$

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}^1} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \mathbf{w}^\top \mathbf{x}_i + b \right)^2$$

- This looks messy, so we simplify a bit:

Multivariate case

- Consider a slightly more general case of $\mathbf{x} \in \mathbb{R}^d, y \in \mathbb{R}$

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}^1} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \mathbf{w}^\top \mathbf{x}_i + b \right)^2$$

- This looks messy, so we simplify a bit:
  - **Trick 1.** Parameter stacking
    - Define $\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \theta = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$

$$\Rightarrow \qquad J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (y - \theta^\top \tilde{\mathbf{x}})^2$$

Multivariate case

- Consider a slightly more general case of $\mathbf{x} \in \mathbb{R}^d$, $y \in \mathbb{R}$

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}^1} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \mathbf{w}^\top \mathbf{x}_i + b \right)^2$$

- This looks messy, so we simplify a bit:

  - **Trick 2.** Data stacking

  - Define $\mathbf{X} = \begin{bmatrix} \tilde{\mathbf{x}}_1^\top \\ \cdots \\ \tilde{\mathbf{x}}_n^\top \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ \cdots \\ y_n \end{bmatrix}$

$$\Rightarrow \qquad J(\theta) = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\theta\|^2$$

Multivariate case

$$J(\theta) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\theta\|^2$$

- Now we examine the critical point, where the gradient is zero.

$$\begin{aligned}
\nabla J(\theta) &= \frac{1}{2n}\nabla\left((\mathbf{y} - \mathbf{X}\theta)^\top(\mathbf{y} - \mathbf{X}\theta)\right) \\
&= \frac{1}{2n}\nabla\left(\mathbf{y}^\top\mathbf{y} + \theta^\top\mathbf{X}^\top\mathbf{X}\theta - 2\mathbf{y}^\top\mathbf{X}\theta\right) \\
&= \frac{1}{2n}\left(2\theta^\top\mathbf{X}^\top\mathbf{X} - 2\mathbf{y}^\top\mathbf{X}\right) = 0
\end{aligned}$$

Multivariate case

$$J(\theta) = \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\theta\|^2$$

- Now we examine the critical point, where the gradient is zero.

$$\begin{aligned}\nabla J(\theta) &= \frac{1}{2n}\nabla\Big((\mathbf{y} - \mathbf{X}\theta)^\top(\mathbf{y} - \mathbf{X}\theta)\Big) \\ &= \frac{1}{2n}\nabla\Big(\mathbf{y}^\top\mathbf{y} + \theta^\top\mathbf{X}^\top\mathbf{X}\theta - 2\mathbf{y}^\top\mathbf{X}\theta\Big) \\ &= \frac{1}{2n}\Big(2\theta^\top\mathbf{X}^\top\mathbf{X} - 2\mathbf{y}^\top\mathbf{X}\Big) = 0\end{aligned}$$

- Thus, the critical point condition is:

$$\mathbf{X}^\top\mathbf{X}\theta = \mathbf{X}^\top\mathbf{y}$$

Multivariate case

$$\mathbf{X}^\top \mathbf{X}\theta = \mathbf{X}^\top \mathbf{y}$$

- If the matrix $\mathbf{X}^\top \mathbf{X}$ happens to be invertible, then we have a unique solution

$$\hat{\theta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Multivariate case

$$\mathbf{X}^\top \mathbf{X}\theta = \mathbf{X}^\top \mathbf{y}$$

- If the matrix $\mathbf{X}^\top \mathbf{X}$ happens to be invertible, then we have a unique solution

$$\hat{\theta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- If not invertible, there exists infinitely many critical points (which are all minima, luckily).
  - One solution. The above takes the form of $\mathbf{A}\theta = \mathbf{b}$
    - Thus simply use QR decomposition

Multivariate case

$$\mathbf{X}^\top \mathbf{X}\theta = \mathbf{X}^\top \mathbf{y}$$

- If the matrix $\mathbf{X}^\top \mathbf{X}$ happens to be invertible, then we have a unique solution

$$\hat{\theta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- If not invertible, there exists infinitely many critical points (which are all minima, luckily).
  - One solution. The above takes the form of $\mathbf{A}\theta = \mathbf{b}$
    - Thus simply use QR decomposition
    - This gives you Moore-Penrose pseudo-inverse $(\mathbf{X}^\top \mathbf{X})^\dagger$
      which gives you a minimum norm solution among all possible θ

Multivariate case

$$\hat{\theta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- **Fun exercise.** Count the number of FLOPs to compute the optimum parameter (i.e., compute the training cost)
  - Hint. This depends on the order of computation!

Alternative way to optimize: Gradient descent

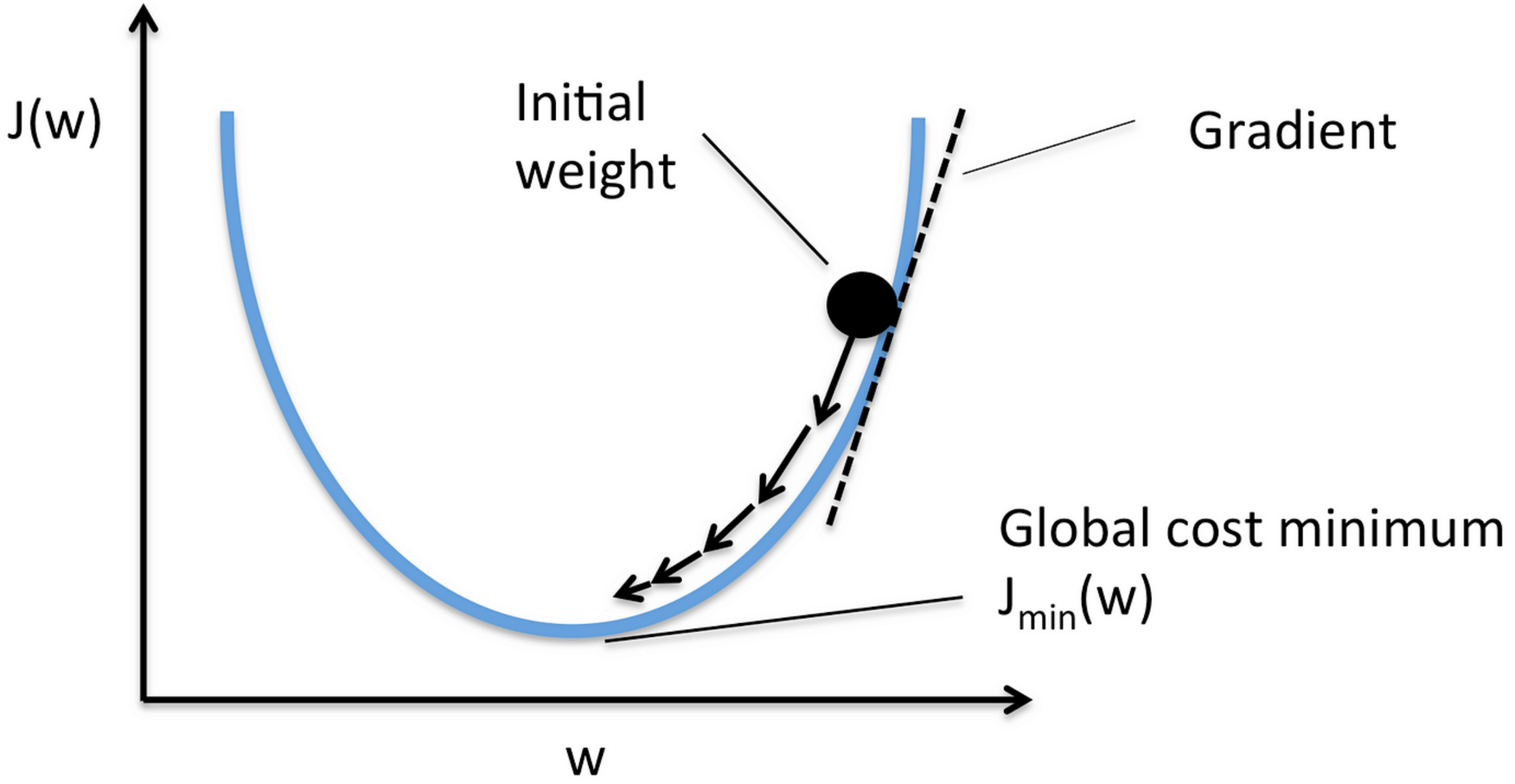
Gradient Descent

- **Rough idea.** Repeat taking steps in the downward direction.

Gradient Descent

- **Rough idea.** Repeat taking steps in the downward direction.

  - Pick a random initial parameter $\theta^{(0)}$, and use the gradient to update $\theta^{(1)}, \theta^{(2)}, \ldots$



Gradient Descent

- **Rough idea.** Repeat taking steps in the downward direction.
  - Pick a random initial parameter $\theta^{(0)}$, and use the gradient to update $\theta^{(1)}, \theta^{(2)}, \ldots$
  - Intuition. Gradient = direction of fastest increase

    $\Rightarrow$ Negative gradient = direction of fastest decrease
    - Take a step toward that direction, with some step size η

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta J(\theta^{(t)})$$

Gradient Descent

- **Rough idea.** Repeat taking steps in the downward direction.
  - Pick a random initial parameter $\theta^{(0)}$, and use the gradient to update $\theta^{(1)}, \theta^{(2)}, \ldots$
  - Intuition. Gradient = direction of fastest increase

    $\Rightarrow$ Negative gradient = direction of fastest decrease
    - Take a step toward that direction, with some step size η

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla_\theta J(\theta^{(t)})$$

    - Plugging in the gradient formula, we get

$$\theta \leftarrow \theta - \frac{\eta}{n}\left(\mathbf{X}^\top \mathbf{X}\theta - \mathbf{X}^\top \mathbf{y}\right)$$

Remarks

$$\theta \leftarrow \theta - \frac{\eta}{n}\left(\mathbf{X}^\top \mathbf{X}\theta - \mathbf{X}^\top \mathbf{y}\right)$$

- **Theoretical.** For certain cases, GD is guaranteed to converge
    - Usually requires diminishing step size

Remarks

$$\theta \leftarrow \theta - \frac{\eta}{n}\left(\mathbf{X}^\top \mathbf{X}\theta - \mathbf{X}^\top \mathbf{y}\right)$$

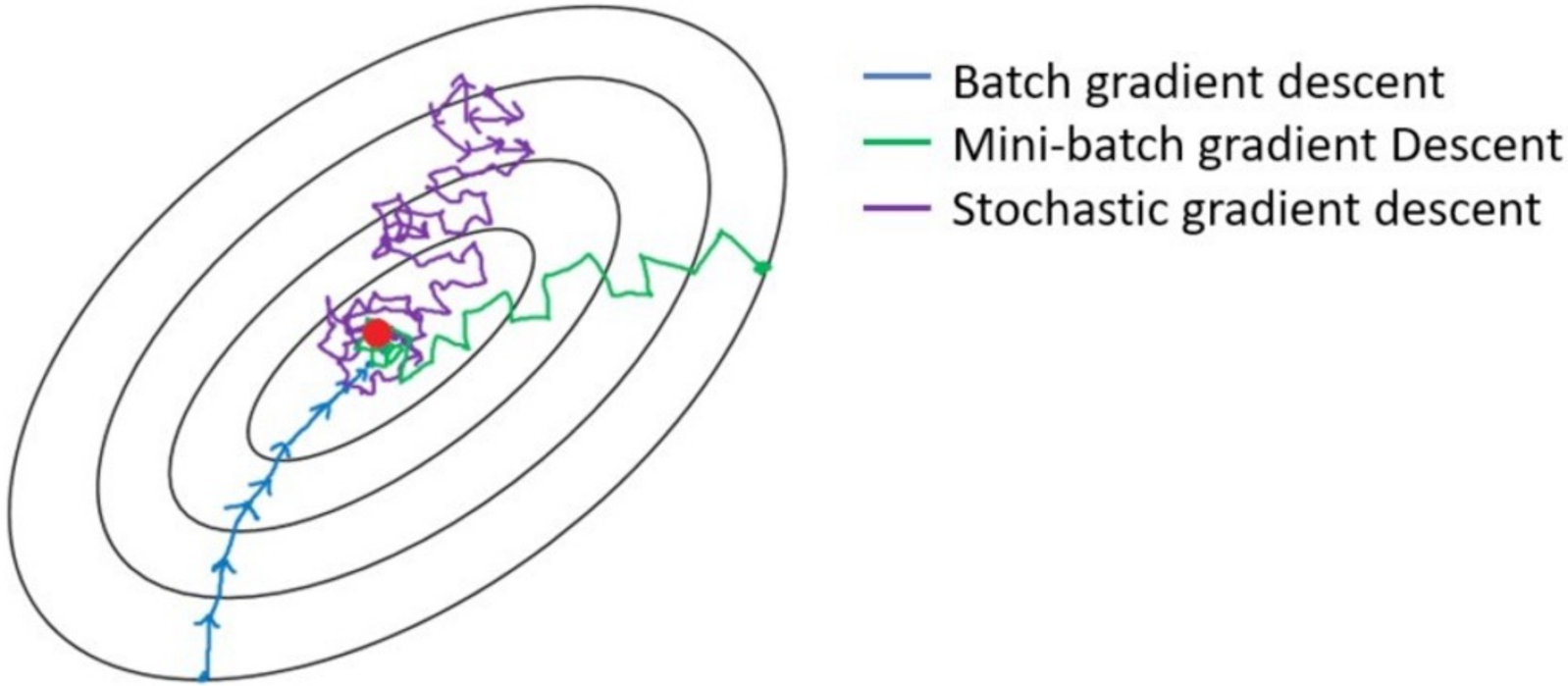
- **Theoretical.** For certain cases, GD is guaranteed to converge
  - Usually requires diminishing step size
- **Computational.** How computationally heavy is GD?
  - One can pre-compute and re-use $\mathbf{A} := \frac{\eta}{n}\mathbf{X}^\top \mathbf{X}$ and $\mathbf{b} := \frac{\eta}{n}\mathbf{X}^\top \mathbf{y}$ over all iterations

$$\theta \leftarrow (\mathbf{I} - \mathbf{A})\theta - \mathbf{b}$$

  - The pre-computing cost is almost same as solving explicitly (thus little merit)
    - Will become handy in cases where no explicit solution is available

Remarks

- **SGD.** You don't need full data for GD
  - Use a randomly drawn subset of k samples in each iteration ($k \ll n$)
    - Called mini-batch GD (or stochastic GD when k=1)
    - This saves much RAM!



Wrapping up

- A basic background for machine learning
  - Empirical risk minimization
  - Supervised learning
- Linear regression
  - Explicit solution
  - Gradient descent

Next up

- Naïve Bayes
- Logistic Regression
- Nearest Neighbors

Cheers