# Machine Learning (CS 181):

# 2. Linear Regression and Foundations

# Contents

# Contents

# Machine Learning Setup

- Inputs
  - Input space: $\mathcal{X} = \mathbb{R}^m$
  - features, covariants, predictors, etc.
- Outputs
  - Output space: $\mathcal{Y}$
  - many different types of predictions
- Our Goal: Learn a model
  - $\hat{y} = h(\mathbf{x})$; model prediction

# Supervised Learning

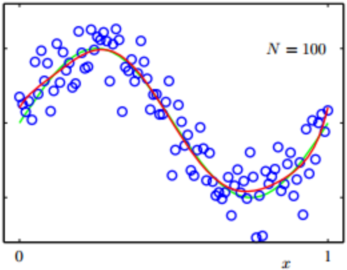
- Given set of input, output pairs

$$D = (\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

- Learn the best $h(\mathbf{x})$ based on $D$
- Predict $y$ for unseen $\mathbf{x}$ based on $h(\mathbf{x})$

# (1) Regression

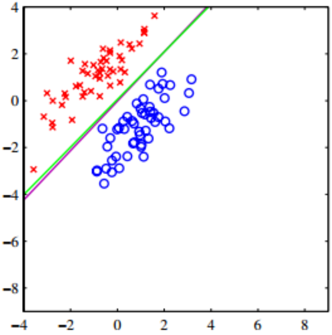
- Output space $\mathcal{Y}$ is real-valued
- Simplest case $\mathcal{Y} = \mathbb{R}$




Polynomial Regression

# (2) Classification

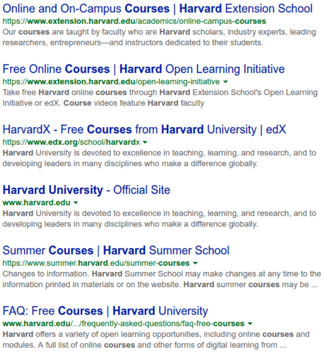
- Output space $\mathcal{Y}$ is a fixed set of classes.
- Simplest case $\mathcal{Y} = -1, 1$ (red/blue)




Binary Classification

# (3) Ordinal Regression / Ranking

- Output space $\mathcal{Y}$ is a ranking of choices.




Search Engine Ranking of Results

# (4) Structured Prediction

- Output space $\mathcal{Y}$ is a structure.

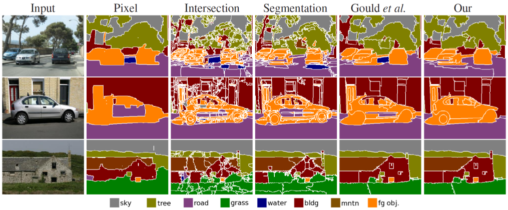



Image Segmentation

# Contents

# Regression Models

We begin by discussing regression.

$$D = (\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$$

- $\mathcal{Y} = \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^m$
- Our aim is function approximation, i.e. find $h(\mathbf{x})$ with "best" $\hat{y}$

Two natural approaches (more later):

- Non-Parametric: directly utilize $D$ for predictions
- Parametric: learn parameters of a model from $D$

# Regression Models

We begin by discussing regression.

$$D = (\mathbf{x}_1, y_1) \ldots (\mathbf{x}_n, y_n) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

- $\mathcal{Y} = \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^m$
- Our aim is function approximation, i.e. find $h(\mathbf{x})$ with "best" $\hat{y}$

Two natural approaches (more later):

- Non-Parametric: directly utilize $D$ for predictions
- Parametric: learn parameters of a model from $D$

# In Brief: A Non-Parametric Regression Approach

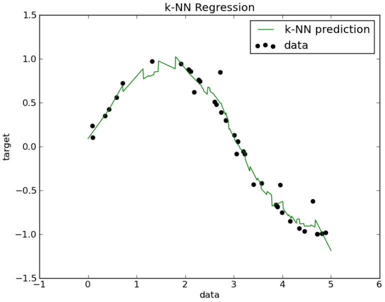
k-Nearest Neighbors learning rule

1. Given new input $\mathbf{x}$
2. Find $k$ "closest" training points $(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(k)}, y^{(k)})$
3. Return average output value

$$\hat{y} = h(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^{k} y_i$$

# K-Nearest Neighbors

- No need to “learn” a model
- Requires keeping around training data
- Need to determine size of $k$




kNN Regression

# In Contrast: Parametric Models

- Parametric Model; $\hat{y} = h(\mathbf{x}; \mathbf{w})$
- $\mathbf{w}$; model parameters, learned from

$$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

Training Procedure

1. Define what it means to "do well"
2. Identify a set of hypotheses (parameterized by $\mathbf{w}$)
3. Pick the best $\mathbf{w}^*$ by minimizing a loss function $\mathcal{L}_D(\mathbf{w})$ .

# In Contrast: Parametric Models

- Parametric Model; $\hat{y} = h(\mathbf{x}; \mathbf{w})$
- $\mathbf{w}$; model parameters, learned from

$$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$$

## Training Procedure

1. Define what it means to "do well"
2. Identify a set of hypotheses (parameterized by $\mathbf{w}$)
3. Pick the best $\mathbf{w}^*$ by minimizing a loss function $\mathcal{L}_D(\mathbf{w})$ .

# Linear Regression

- Parametric model where $h(\mathbf{x}; \mathbf{w})$ is a linear function of $\mathbf{x}$.
- Our "hypothesis" is that there is some good linear function of input to find prediction $\hat{\mathbf{y}}$
- We select this by choosing $\mathbf{w}$

# Linear Regression: Formally

Learn $h(\mathbf{x};\mathbf{w})$ with

- Input: $\mathbf{x}$ where $x_j \in \mathbb{R}$ for $j \in 1, \ldots, m$ features
- Parameters: $\mathbf{w} \in \mathbb{R}^m$, $w_0 \in \mathbb{R}$
- Model Function:

$$\begin{aligned} h(\mathbf{x};\mathbf{w}) &= w_0 + w_1 x_1 + \cdots + w_m x_m \\ &= \sum_{j=1}^{m} w_j x_j + w_0 \\ &= \mathbf{w}^\top \mathbf{x} + w_0 \end{aligned}$$

# Linear Regression: Loss

- Have model and data, need a loss (why?)
- Most common: squared loss

$$\begin{aligned}\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - h(\mathbf{x}_i;\mathbf{w})\big)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - \hat{y}_i\big)^2.\end{aligned}$$

- Training: find minimizer of this loss (least squares)

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

# Linear Regression: Loss

- Have model and data, need a loss (why?)
- Most common: squared loss

$$
\begin{aligned}
\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}\left(y_i - h(\mathbf{x}_i;\mathbf{w})\right)^2 \\
&= \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2.
\end{aligned}
$$

- Training: find minimizer of this loss (least squares)

$$
\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})
$$

# Linear Regression: Loss

- Have model and data, need a loss (why?)
- Most common: squared loss

$$\begin{aligned}\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - h(\mathbf{x}_i;\mathbf{w})\big)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - \hat{y}_i\big)^2.\end{aligned}$$

- Training: find minimizer of this loss (least squares)

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

# Linear Regression: Loss

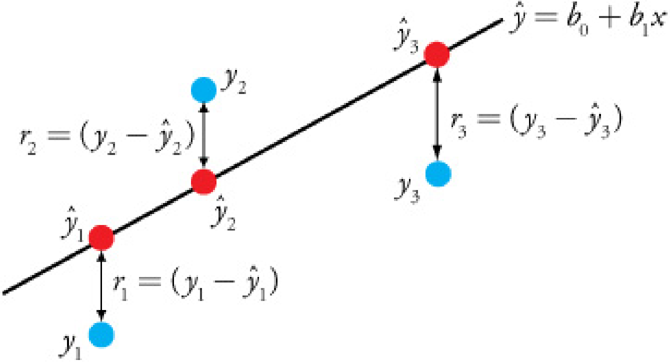
- Have model and data, need a loss (why?)
- Most common: squared loss

$$
\begin{aligned}
\mathcal{L}_D(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - h(\mathbf{x}_i;\mathbf{w})\big)^{2} \\
&= \frac{1}{2}\sum_{i=1}^{n}\big(y_i - \hat{y}_i\big)^{2}.
\end{aligned}
$$

- Training: find minimizer of this loss (least squares)

$$
\mathbf{w}^{\star} = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})
$$

# Least Squared Loss




Contributing loss terms with $m = 1$.

# Contents

# Optimization

Minimizing loss functions can be hard...

Optimization is a central part of machine learning.

- Closed-form solutions (Rare)
- Gradient descent
- Linear and quadratic programming
- Newton-like methods
- Various global optimization ideas and heuristics
- Stochastic optimization
- Lots more...

# Main Tool: Gradients

Vector of derivatives ($g : \mathbb{R}^n \mapsto \mathbb{R}$):

$$\frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial}{\partial z_1} g(\mathbf{z}) \\ \frac{\partial}{\partial z_2} g(\mathbf{z}) \\ \vdots \\ \frac{\partial}{\partial z_n} g(\mathbf{z}) \end{bmatrix}$$

# Matrix Calculus Identities

$$\frac{\partial}{\partial \mathbf{z}} \mathbf{z}^\top \mathbf{a} = \frac{\partial}{\partial \mathbf{z}} \mathbf{a}^\top \mathbf{z} = \mathbf{a}$$

$$\frac{\partial}{\partial \mathbf{z}} \mathbf{z}^\top \mathbf{A} \mathbf{z} = (\mathbf{A} + \mathbf{A}^\top) \mathbf{z}$$

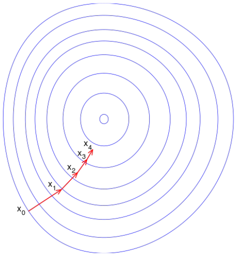
Many other identities and derivations in Matrix Cookbook

# Preview: Gradient Descent

Minimize loss by repeated gradient steps (when no closed form):

1. Compute gradient of loss with respect to parameters $\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$
2. Update parameters with rate $\eta$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$$




Gradient steps on a simple $m = 2$ loss function.

# Back to Linear Regression: Matrix Version

- Design matrix: $\mathbf{X} \in \mathbb{R}^{n \times m}$,

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{bmatrix} \tag{1}$$

- Target vector: $\mathbf{y} \in \mathbb{R}^{n(\times 1)}$,

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2}$$

# Least Squares Loss (Vector Form)

Nice case, closed-form solution.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - h(\mathbf{x}_i;\mathbf{w}))^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{w}^\top\mathbf{x}_i)^2.$$

$$\begin{aligned}
\frac{\partial\mathcal{L}(\mathbf{w})}{\partial\mathbf{w}} &= -\sum_{i=1}^{n}(y_i - \mathbf{w}^\top\mathbf{x}_i)\mathbf{x}_i && (3)\\
&= -\sum_{i=1}^{n} y_i\mathbf{x}_i + \sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top\mathbf{w} && (4)\\
&= -\sum_{i=1}^{n} y_i\mathbf{x}_i + \left(\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top\right)\mathbf{w}. && (5)
\end{aligned}$$

# Least Squares Loss (Matrix Form)

Same as above, but using matrix calculus identities.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})$$

$$\begin{aligned}
2 \times \frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) \\
&= \frac{\partial}{\partial \mathbf{w}}\left(\mathbf{y}^\top\mathbf{y} - 2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}\right) \\
&= \frac{\partial}{\partial \mathbf{w}}\mathbf{y}^\top\mathbf{y} - \frac{\partial}{\partial \mathbf{w}}2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \frac{\partial}{\partial \mathbf{w}}\mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} \\
&= 0 - 2\mathbf{X}^\top\mathbf{y} + (\mathbf{X}^\top\mathbf{X} + (\mathbf{X}^\top\mathbf{X})^\top)\mathbf{w} \\
&= -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\mathbf{w}
\end{aligned}$$

# Least Squares Matrix

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}) = -\sum_{i=1}^{n} y_i \mathbf{x}_i + \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top\right)\mathbf{w}. \tag{6}$$

$$= -\mathbf{X}^\top \mathbf{y} + \mathbf{X}^\top \mathbf{X}\mathbf{w} \tag{7}$$

Set to 0, and solve for optimal parameters $\mathbf{w}^*$

$$\mathbf{w}^\star = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

- (FYI: Known as Moore-Penrose pseudo-inverse

  $$(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$$

  Generalization of inverse for non-square matrix).

# Demo