Outline

• Motivation

• Shallow Neural Networks

• Linear Regression

• Analytic Solution

• Gradient Descent
Deep Learning
Convolutional Neural Networks

(source: Universität Bonn)
Recurrent Neural Networks
RNN Beer Review

(a) Real Review

Poured from 12oz bottle into half-liter Pilsner Urquell branded pilsner glass. **Appearance:** Pours a cloudy **golden-orange** color with a small, quickly dissipating white head that leaves a bit of lace behind. **Smell:** Smells HEAVILY of citrus. By heavily, I mean that this smells like kitchen cleaner with added wheat. **Taste:** Tastes heavily of citrus- lemon, lime, and orange with a hint of wheat at the end. **Mouthfeel:** Thin, with a bit too much carbonation. Refreshing. **Drinkability:** If I wanted lemonade, then I would have bought that.

(b) Synthetic review

Poured from a 12oz bottle into a 16oz Samuel Adams Perfect Pint glass. **Appearance:** Very pale **golden color** with a thin, white head that leaves little lacing. **Smell:** Very mild and inoffensive aromas of citrus. **Taste:** Starts with the same tastes of the citrus and fruit **flavors of orange and lemon and the orange taste** is all there. There is a **little bit of wheat** that is pretty weak, but it is sort of harsh (in a good way) and ends with a slightly bitter aftertaste. **Mouthfeel:** Light body with a little alcohol burn. Finish is slightly dry with some lingering spice. **Drinkability:** A decent beer, but not great. I don’t think I would rate this anytime soon as it says that there are other Belgian beers out there, but this is a good choice for a warm day when it's always available in the North Coast Brewing Company party.

http://deepx.ucsd.edu/#/home/beermind
Supervised Learning

"Time underlies many interesting human behaviors"

$f()$

{0,1}
{A,B,C…}
captions,
email mom,
fire nukes,
eject pop tart
One Function-Generator: Programmers
Learning Functions from Examples
Outline

• Motivation

• **Shallow Neural Networks**

• Linear Regression

• Analytic Solution

• Gradient Descent
One Class of Functions: Neural Networks
Ingredients for a Neural Networks Algorithm

- Inputs
- Outputs
- Architecture
- Objective Function
- Learning Rule
Components of a Neural Networks Algorithm

- Inputs
- Outputs
- Architecture
- Objective Function
- Learning Rule

\{ Data \}

\{ Model \}
The Simplest Neural Network
Anatomy of Simplest Net

Activation function $f(\cdot)$

Weighted sum $\sum$

Output

Weights ($w$)

Inputs
What’s Missing?

- Activation function
- Objective function
- Learning rule
Linear Models

- Linear Regression
  - Linear activation
  - Square loss
- Perceptron
  - Threshold function
  - Hinge loss
- Logistic Regression
  - Sigmoid Activation
  - Cross-entropy loss
Outline

• Motivation
• Shallow Neural Networks
• **Linear Regression**
• Analytic Solution
• Gradient Descent
Linear Regression
Generating Output (point)

For a Single Example:

• Output: $\hat{y} = w^T x$
• Weights: $w \in \mathbb{R}^d$
• Input: $x \in \mathbb{R}^d$
Generating Output (dataset)

- Output: \( \hat{y} = Xw \in \mathbb{R}^n \)
- Weights: \( w \in \mathbb{R}^d \)
- Input: \( X \in \mathbb{R}^{n \times d} \)
What loss function should we minimize?

- Ground truth \( y \) and predictions \( \hat{y} \) are real valued.
- We need a measure \( \ell(y, \hat{y}) \) of how well \( \hat{y} \) approximates \( y \).
Square Loss

• One popular loss function is squared loss

\[(y - \hat{y})^2\]

• Over a whole dataset, the total squared error

\[\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2\]

• Common to report mean squared error

\[\|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 / n\]

• Pros: convex, optimized by mean - any cons?
Outline

• Motivation
• Shallow Neural Networks
• Linear Regression
• Analytic Solution
• Gradient Descent
Analytic Solution

\[ \ell(y, \hat{y}) = \|y - Xw\|_2^2 \]

\[ = \sum_{i=1}^{n} (y_i - w^T x_i)^2 \]

\[ = (y - Xw)^T (y - Xw) \]

\[ \frac{\partial \ell(y, \hat{y})}{\partial w} = 2(-X)^T (y - Xw) = 0 \]

\[ -2X^T y + 2X^T Xw = 0 \]

\[ X^T Xw = X^T y \]

\[ w = (X^T X)^{-1} X^T y \]
When can’t we use this/any analytic solution?

• [class participation]
When can’t we use this/any analytic solution?

• The multiplication $X^T X$ can be very slow for large datasets (many examples)

• The matrix inversion $(X^T X)^{-1}$ can be slow for high dimensional data (many features)

• Many optimizations (e.g. all modern neural networks) cannot be solved in closed form
Outline

• Motivation

• Shallow Neural Networks

• Linear Regression

• Analytic Solution

• Gradient Descent
Gradient descent

- Start with initial (random) guess $w_1$

- For $T$ rounds, update
  
  $$w \leftarrow w - \eta \nabla_w \ell(y, \hat{y})$$

  $$\nabla_w \ell(y, \hat{y}) = \frac{d\ell(y, \hat{y})}{dw}$$
Stochastic Gradient Descent

• Taking the derivative over the full data set is expensive

• Gradient descent requires one gradient per step

• What if we use a random subset of the data to estimate the gradient?

• Guaranteed to converge under mild conditions
Stochastic Gradient Descent

• For $t$ in $1 \ldots T$
  
  $t \in 1, \ldots T$
  
  Sample row $x_i \in x_1, \ldots x_n$
  
  Calculate prediction $\hat{y}_i = w^T x_i$
  
  Calculate loss $\ell(y_i, \hat{y}_i) = \|y_i - \hat{y}_i\|_2^2$
  
  Apply update $w \leftarrow w - \eta \nabla_w \ell(y_i, \hat{y}_i)$