CSC 411: Lecture 02: Linear Regression

Richard Zemel, Raquel Urtasun and Sanja Fidler

University of Toronto

(Most plots in this lecture are from Bishop's book)

Problems for Today

- What should I watch this Friday?
- **Goal**: Predict movie rating automatically!
- **Goal:** How many followers will I get?
- **Goal:** Predict the price of the house

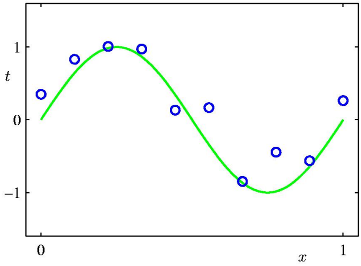
Regression

- What do all these problems have in common?
  - Continuous outputs, we'll call these t
    (e.g., a rating: a real number between 0-10, # of followers, house price)
- Predicting continuous outputs is called regression
- What do I need in order to predict these outputs?
  - Features (inputs), we'll call these x (or $\mathbf{x}$ if vectors)
  - Training examples, many $x^{(i)}$ for which $t^{(i)}$ is known (e.g., many movies for which we know the rating)
  - A model, a function that represents the relationship between x and t
  - A loss or a cost or an objective function, which tells us how well our model approximates the training examples
  - Optimization, a way of finding the parameters of our model that minimizes the loss function

Today: Linear Regression

- Linear regression
  - continuous outputs
  - simple model (linear)
- Introduce key concepts:
  - loss functions
  - generalization
  - optimization
  - model complexity
  - regularization

Simple 1-D regression

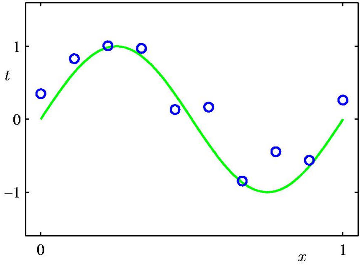


- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in x, but may be displaced in y

$$t(x) = f(x) + \epsilon$$

  with ϵ some noise
- In green is the "true" curve that we don't know
- Goal: We want to fit a curve to these points

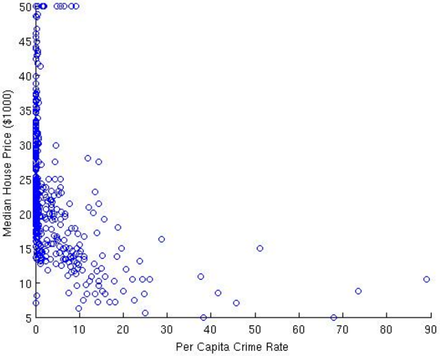
Simple 1-D regression



- Key Questions:
  - How do we parametrize the model?
  - What loss (objective) function should we use to judge the fit?
  - How do we optimize fit to unseen test data (generalization)?

Example: Boston Housing data

- Estimate median house price in a neighborhood based on neighborhood statistics
- Look at first possible attribute (feature): per capita crime rate



- Use this to predict house prices in other neighborhoods
- Is this a good input (attribute) to predict house prices?

Represent the Data

- Data is described as pairs $\mathcal{D} = \{(x^{(1)}, t^{(1)}), \cdots, (x^{(N)}, t^{(N)})\}$
  - $x \in \mathbb{R}$ is the input feature (per capita crime rate)
  - $t \in \mathbb{R}$ is the target output (median house price)
  - $^{(i)}$ simply indicates the training examples (we have N in this case)
- Here t is continuous, so this is a regression problem
- Model outputs y, an estimate of t

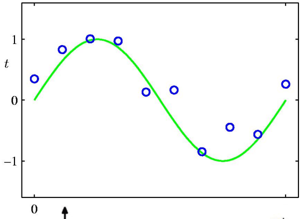
$$y(x) = w_0 + w_1 x$$

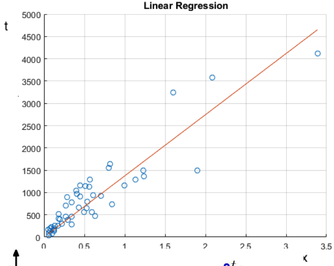
- What type of model did we choose?
- Divide the dataset into training and testing examples
  - Use the training examples to construct hypothesis, or function approximator, that maps x to predicted y
  - Evaluate hypothesis on test set

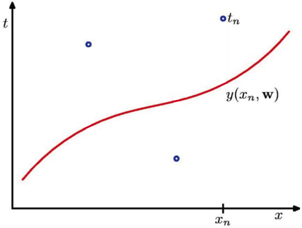
Noise

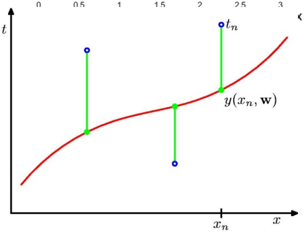
- A simple model typically does not exactly fit the data
  - lack of fit can be considered noise
- Sources of noise:
  - Imprecision in data attributes (input noise, e.g., noise in per-capita crime)
  - Errors in data targets (mis-labeling, e.g., noise in house prices)
  - Additional attributes not taken into account by data attributes, affect target values (latent variables). In the example, what else could affect house prices?
  - Model may be too simple to account for data targets

Least-Squares Regression









- Define a model

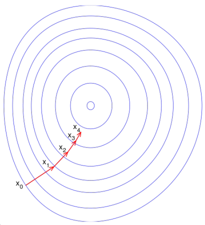
$$y(x) = \text{function}(x, \mathbf{w})$$

Linear: $y(x) = w_0 + w_1 x$

Optimizing the Objective

- One straightforward method: gradient descent
  - initialize $\mathbf{w}$ (e.g., randomly)
  - repeatedly update $\mathbf{w}$ based on the gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \frac{\partial \ell}{\partial \mathbf{w}}$$



- λ is the learning rate
- For a single training case, this gives the LMS update rule (Least Mean Squares):

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda(t^{(n)} - y(x^{(n)}))x^{(n)}$$

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \underbrace{\left(t^{(n)} - y(x^{(n)})\right)}_{\text{error}} x^{(n)}$$

- Note: As error approaches zero, so does the update ($\mathbf{w}$ stops changing)

Optimizing Across Training Set

- Two ways to generalize this for all examples in training set:
    1. Batch updates: sum or average updates across every example n, then change the parameter values

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda \sum_{n=1}^{N} (t^{(n)} - y(x^{(n)}))x^{(n)}$$

    2. Stochastic/online updates: update the parameters for each training case in turn, according to its own gradients

Algorithm 1 Stochastic gradient descent

1: Randomly shuffle examples in the training set
2: **for** $i = 1$ to N **do**
3: Update:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda(t^{(i)} - y(x^{(i)}))x^{(i)} \qquad \text{(update for a linear model)}$$

4: **end for**

Analytical Solution?

- For some objectives we can also find the optimal solution analytically
- This is the case for linear least-squares regression
- How?
- Compute the derivatives of the objective wrt $\mathbf{w}$ and equate with 0
- Define:

$$\mathbf{t} = [t^{(1)}, t^{(2)}, \ldots, t^{(N)}]^T$$

$$\mathbf{X} = \begin{bmatrix} 1, x^{(1)} \\ 1, x^{(2)} \\ \ldots \\ 1, x^{(N)} \end{bmatrix}$$

- Then:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$$

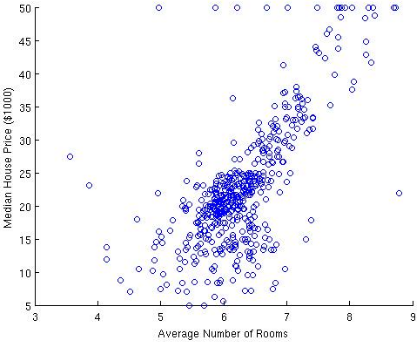
(work it out!)

Multi-dimensional Inputs

- One method of extending the model is to consider other input dimensions

$$y(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2$$

- In the Boston housing example, we can look at the number of rooms



Linear Regression with Multi-dimensional Inputs

- Imagine now we want to predict the median house price from these multi-dimensional observations
- Each house is a data point n, with observations indexed by j:

$$\mathbf{x}^{(n)} = \left(x_1^{(n)}, \cdots, x_j^{(n)}, \cdots, x_d^{(n)}\right)$$

- We can incorporate the bias w_0 into $\mathbf{w}$, by using $x_0 = 1$, then

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{d} w_j x_j = \mathbf{w}^T \mathbf{x}$$

- We can then solve for $\mathbf{w} = (w_0, w_1, \cdots, w_d)$. How?
- We can use gradient descent to solve for each coefficient, or compute $\mathbf{w}$ analytically (how does the solution change?)

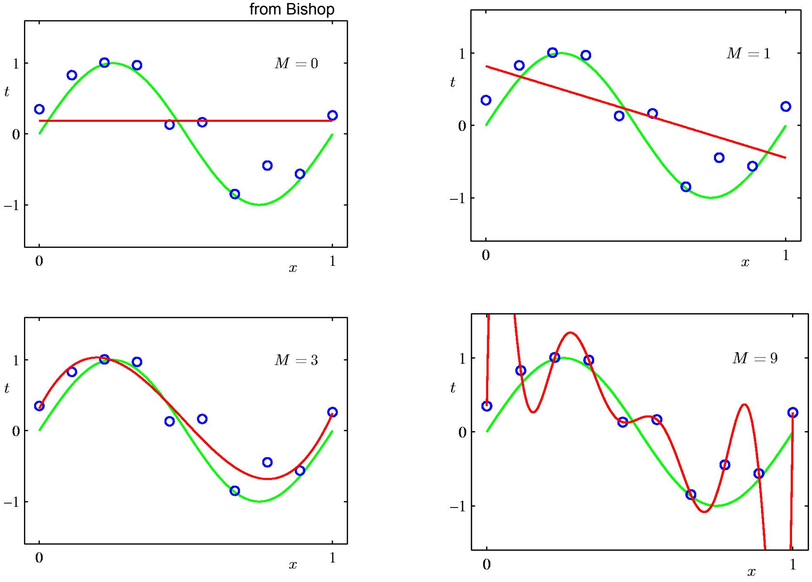
More Powerful Models?Fitting a Polynomial

- What if our linear model is not good? How can we create a more complicated model?
- We can create a more complicated model by defining input variables that are combinations of components of $\mathbf{x}$
- Example: an M-th order polynomial function of one dimensional feature x:

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j x^j$$

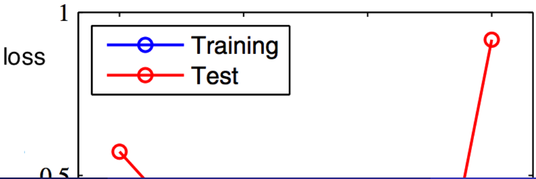
  where x^j is the j-th power of x
- We can use the same approach to optimize for the weights $\mathbf{w}$
- How do we do that?

Which Fit is Best?



Generalization

- Generalization = model's ability to predict the held out data
- What is happening?
- Our model with $M = 9$ overfits the data (it models also noise)
- Not a problem if we have lots of training examples
- Let's look at the estimated weights for various M in the case of fewer examples
- The weights are becoming huge to compensate for the noise
- One way of dealing with this is to encourage the weights to be small (this way no input dimension will have too much influence on prediction). This is called regularization



Regularized Least Squares

- Increasing the input features this way can complicate the model considerably
- Goal: select the appropriate model complexity automatically
- Standard approach: regularization

$$\tilde{\ell}(\mathbf{w}) = \sum_{n=1}^{N} [t^{(n)} - (w_0 + w_1 x^{(n)})]^2 + \alpha \mathbf{w}^T \mathbf{w}$$

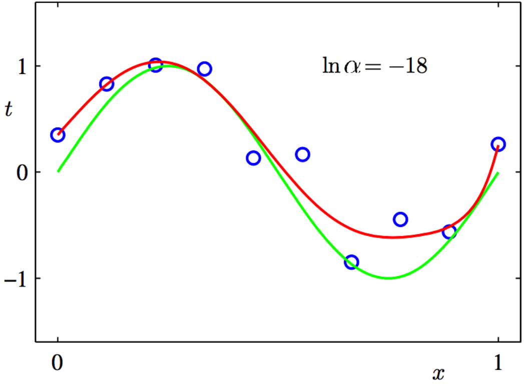
- Intuition: Since we are minimizing the loss, the second term will encourage smaller values in $\mathbf{w}$
- When we use the penalty on the squared weights we have ridge regression in statistics
- Leads to a modified update rule for gradient descent:

$$\mathbf{w} \leftarrow \mathbf{w} + 2\lambda[\sum_{n=1}^{N} (t^{(n)} - y(x^{(n)}))x^{(n)} - \alpha \mathbf{w}]$$

- Also has an analytical solution: $\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \alpha\,\mathbf{I})^{-1} \mathbf{X}^T \mathbf{t}$ (verify!)

Regularized least squares

- Better generalization
- Choose α carefully



1-D regression illustrates key concepts

- Data fits – is linear model best (model selection)?
  - Simple models may not capture all the important variations (signal) in the data: underfit
  - More complex models may overfit the training data (fit not only the signal but also the noise in the data), especially if not enough data to constrain model
- One method of assessing fit: test generalization = model's ability to predict the held out data
- Optimization is essential: stochastic and batch iterative approaches; analytic when available

So...

- Which movie will you watch?