CentraleSupélec

Introduction to Machine Learning

MA2823

Lecture 4

Linear and logistic regression

Fragkiskos Malliaros

Friday, October 12, 2017

Some Updates

- The project proposal is due on October 14
  - Upload in gradescope
  - Group submission (one member of every team; add full names of all team members)

- The first assignment is due on October 21
  - Upload in gradescope

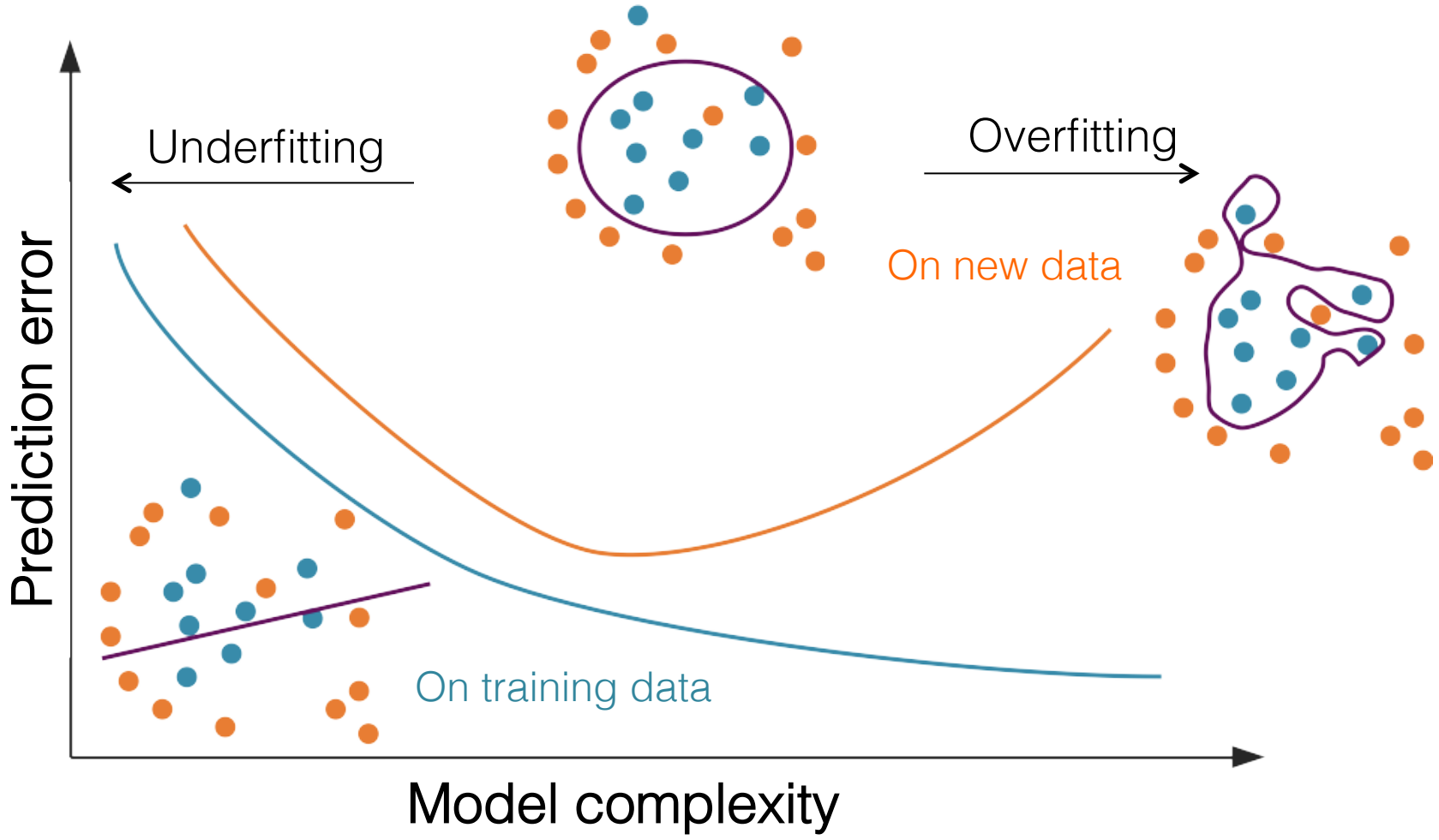
Acknowledgements

- The lecture is partially based on material by
  - Richard Zemel, Raquel Urtasun and Sanja Fidler (University of Toronto)
  - Chloé-Agathe Azencott (Mines ParisTech)
  - Julian McAuley (UC San Diego)
  - Dimitris Papailiopoulos (UW-Madison)
  - Jure Leskovec, Anand Rajaraman, Jeff Ullman (Stanford Univ.)
    - http://www.mmds.org
  - Panagiotis Tsaparas (UOI)
  - Evimaria Terzi (Boston University)
  - Andrew Ng (Stanford University)
  - Nina Balcan and Matt Gormley (CMU)

Thank you!

Last lecture

Generalization Error vs. Model Complexity



Fixed dataset size; varying model complexity

Evaluating model performance

Classification Model Evaluation

- Confusion matrix

		True class	
		-1	+1
Predicted class	-1	True Negatives	False Negatives
	+1	False Positives	True Positives

- False positives (false alarms) are also called type I errors
- False negatives (misses) are also called type II errors

Evaluation Measures (1/2)

- Sensitivity = Recall = True positive rate (TPR)

$$TPR = \frac{TP}{TP + FN}$$

# of positives

- Specificity = True negative rate (TNR)

$$TNR = \frac{TN}{FP + TN}$$

- Precision = Positive predictive value (PPV)

$$PPV = \frac{TP}{TP + FP}$$

# of predicted positives

- False discovery rate (FDR)

$$FDR = \frac{FP}{FP + TP}$$

Evaluation Measures (2/2)

- Accuracy

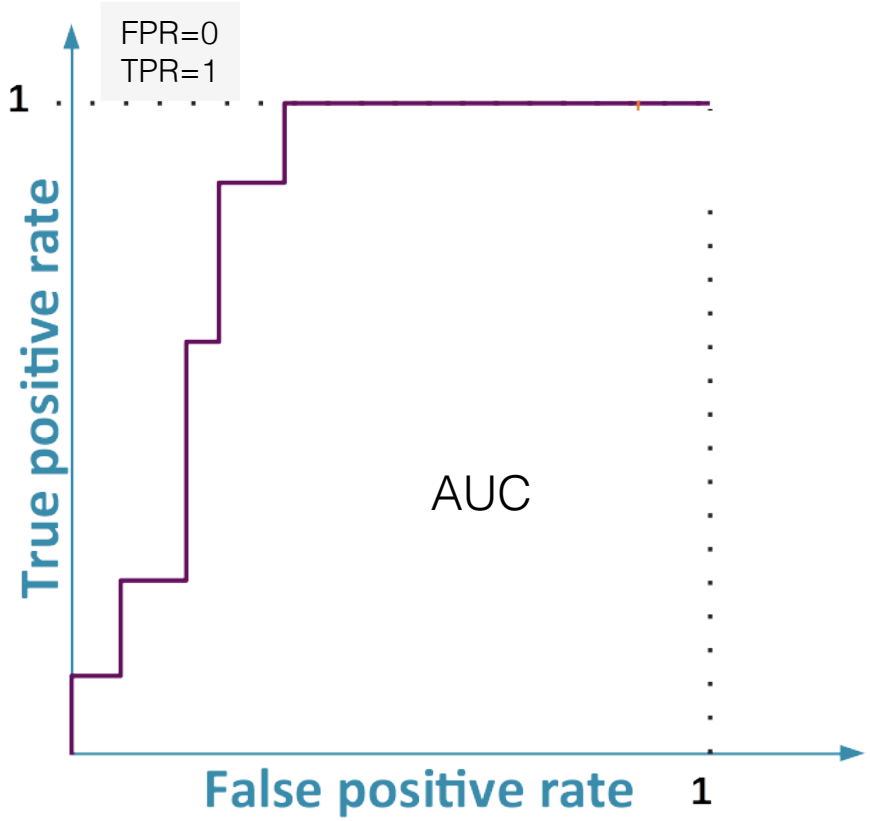
$$Acc = \frac{TP + TN}{TP + FN + FP + TN}$$

- F1-score: harmonic mean of precision and recall (sensitivity)

$$F1 = \frac{2TP}{2TP + FP + FN}$$

ROC Curve

- ROC = Receiver-Operator Characteristic
- Summarized by the area under the curve (AUROC or AUC)



- Plot TPR vs. FPR for all possible thresholds (typically generated by the classifier)

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN}$$

- Shows the trade off in sensitivity (TPR) and (1 – specificity) (FPR) for all possible thresholds (cutoff values between positive and negative class)

- The larger the AUC, the better is overall performance

Source: https://en.wikipedia.org/wiki/Receiver_operating_characteristic
Visual description of ROC curve: https://www.youtube.com/watch?v=OAl6eAyP-yo

Evaluation Measures in scikit-learn

http://scikit-learn.org/stable/modules/model_evaluation.html

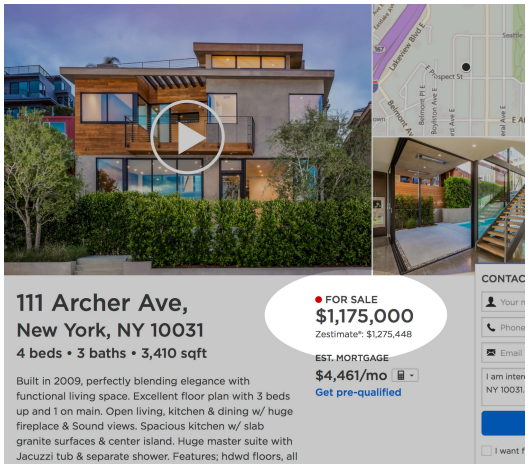
Linear regression

Regression – Motivation

- Application 1: Predict movie rating automatically

- Application 2: Can I predict the price of a 70 m^2 house in Rue de Rivoli?

- Application 3: How many followers will I get in twitter?



Zillow Kaggle competition ($1,2M)

Regression – Introduction (1/2)

- What do all those problems have in common?
  - Continues outputs y (e.g., a rating: a real number between 0-5, # of followers, house price)
- The task of predicting continues outputs is called regression

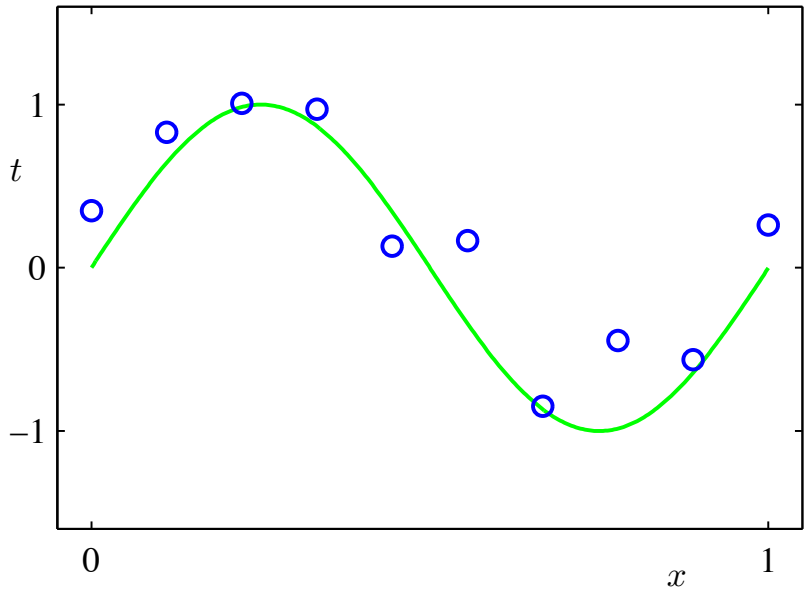
Regression is one of the simplest supervised learning approaches to learn relationships between input variables (features) and output variables (predictions)

Regression – Introduction (2/2)

- What do I need in order to predict these outputs?
  - Features (inputs): denoted by x
  - Training examples: many $x^{(i)}$ for which $y^{(i)}$ is known (e.g., many movies for which we know the rating)
  - A model: a function that represents the relationship between x and y
  - 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

Example

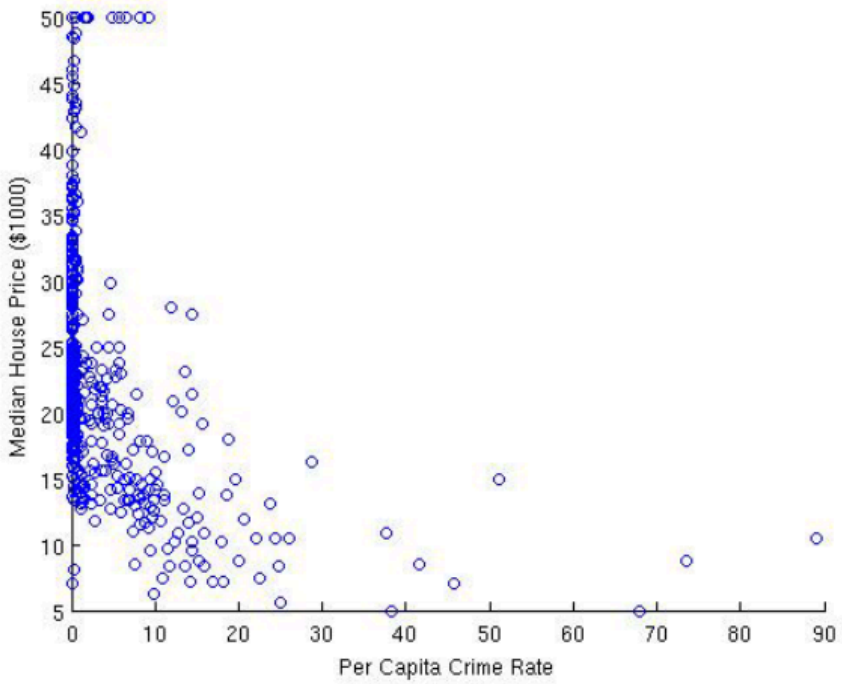
- Blue circles are data points (i.e., training examples)
  - Goal: we want to fit a curve (green) to these data points



- Key questions:
  - How do we parameterize the model?
  - What loss function should we use to judge the fit?
  - How do we optimize the fit to the unseen test data?

Example: Boston Housing Data

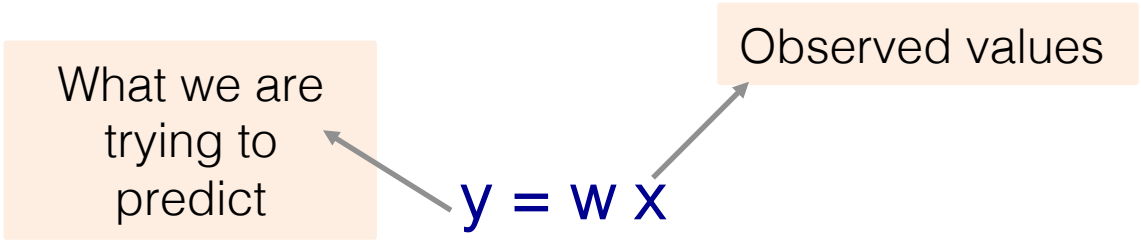
- Estimate median house price in a neighborhood based on neighborhood statistics
- Look at first possible attribute (feature): per capita crime rate



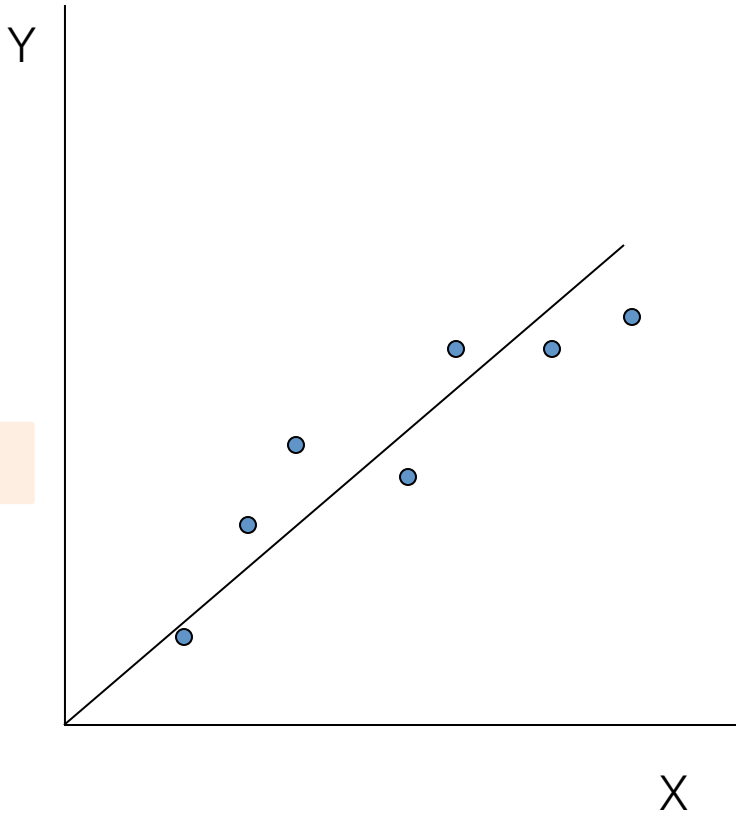
- Use those data to predict prices in other neighborhoods
- Is this a good attribute to predict house prices?

Linear Regression

- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:



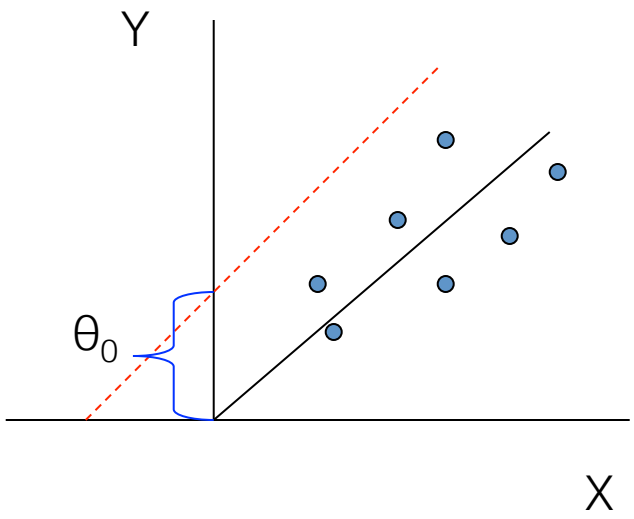
where w is a parameter of the model



Linear Regression – The General Model

- So far we assumed that the line passes through the origin
- What if the line does not?
- Simply change the model to

$$y = \theta_0 + \theta_1 x$$



- (The above can be generalized to multi-dimensional data points)
- Idea: we can use least squares to determine θ_0, θ_1

Linear Regression – Problem Formulation

Data: Inputs are continuous vectors of length K (dimensions, features). Outputs are continuous scalars

$$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \text{ where } \mathbf{x} \in \mathbb{R}^K \text{ and } y \in \mathbb{R}$$

Prediction: Output is a linear function of the inputs

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_K x_K$$

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}$$

(We assume x_1 is 1)

Learning: finds the parameters θ that minimize some objective function

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

Least-Squares Regression (1/2)

- Learning: finds the parameters that minimize some objective function

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

Very general optimization setup. Many ways to solve it

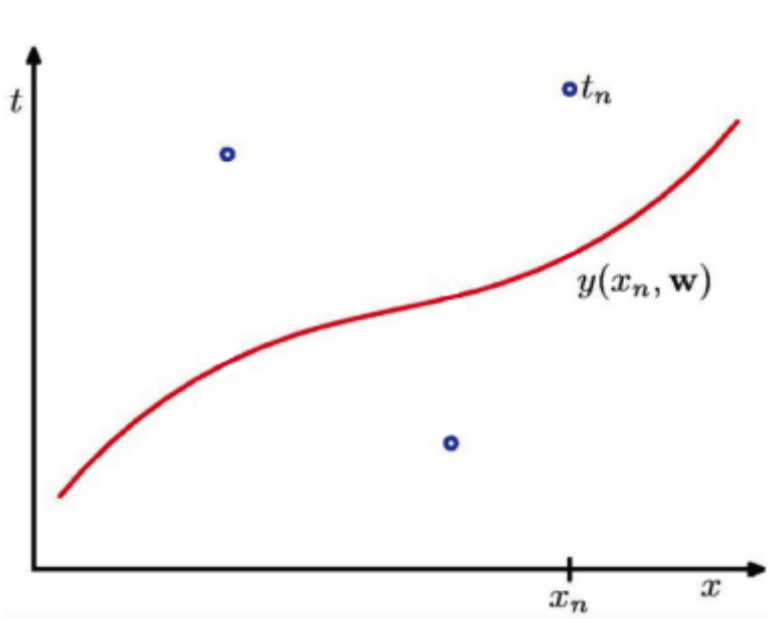
[Objective function] We minimize the sum of the squares:

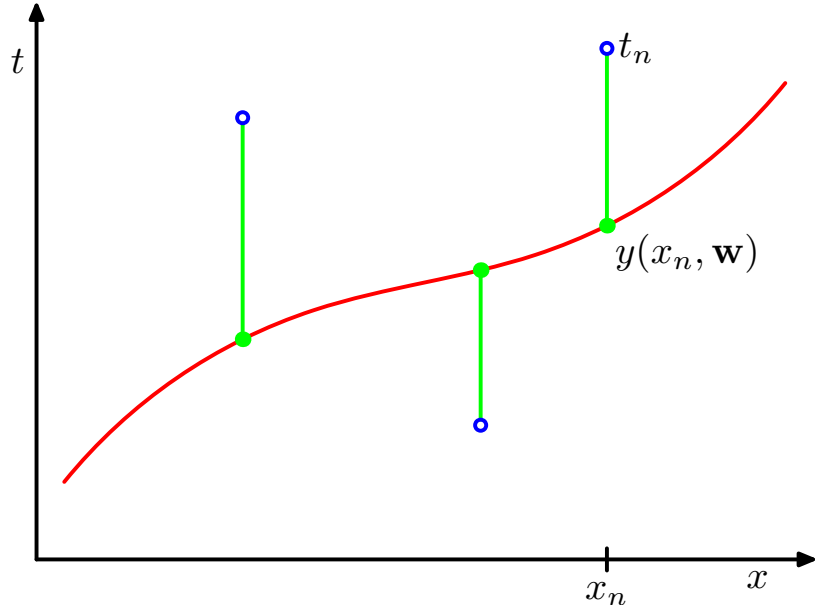
$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2$$

Why?

– It reduces distance between true measurements and the predicted hyperplane (line in 1-D)

Least-Squares Regression (2/2)





Define a model:

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

In our case, it's a linear model: $y(x) = \theta_0 + \theta_1 x$

Optimizing the Objective

- Learning: Three approaches to solving $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$

- Approach 1: Gradient Descent (or Batch GD)
  - Take larger steps to the opposite direction of the gradient
    (For a single update, it computes the gradient using the whole dataset)

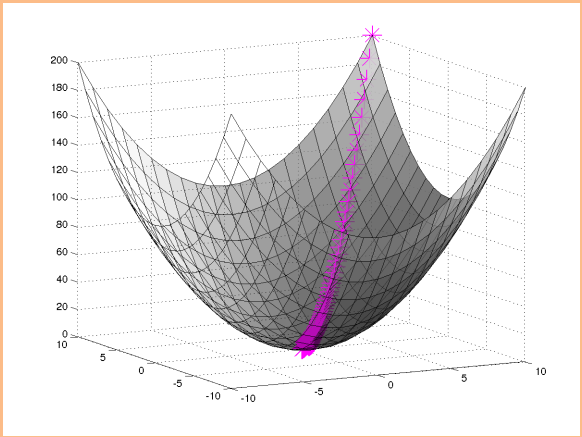
- Approach 2: Stochastic Gradient Descent (or Incremental GD)
  - Take many small steps to the opposite direction of the gradient
    (Updates the gradient using a single sample of the training set)

- Approach 3: Closed form solution
  - Set derivatives equal to zero and solve for parameters

A1: (Batch) Gradient Descent

Algorithm 1 Gradient Descent

1: **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
5: $\quad$**return** $\boldsymbol{\theta}$



In order to apply GD to Linear Regression, we need to compute the gradient of the objective function (i.e., vector of partial derivatives)

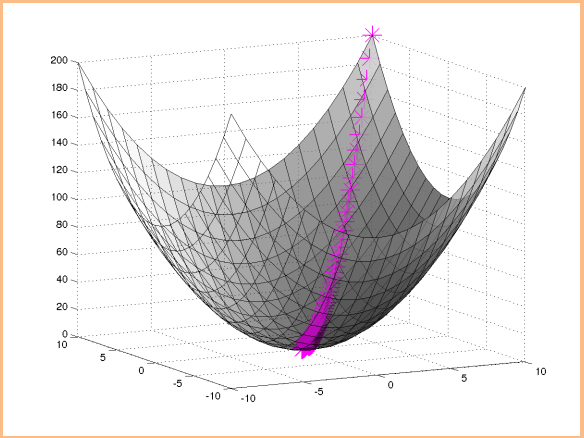
$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_K} J(\boldsymbol{\theta}) \end{bmatrix}$$

More details about the derivative in a while

Gradient Descent – Convergence

Algorithm 1 Gradient Descent

1: **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
5: $\quad$**return** $\boldsymbol{\theta}$



- There are many possible ways to detect convergence. For example, we could check whether the L2 norm of the gradient is below some small tolerance

$$||\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})||_2 \leq \epsilon$$

- Alternatively we could check that the reduction in the objective function from one iteration to the next is small

A2: Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

1: **procedure** SGD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad$**for** $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
5: $\quad\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
6: $\quad$**return** $\boldsymbol{\theta}$

- Applied to Linear Regression, SGD is called the Least Mean Squares (LMS) algorithm
- We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$
$$\text{where } J^{(i)}(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2.$$

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

1: **procedure** SGD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad$**for** $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
5: $\quad\quad\quad$**for** $k \in \{1, 2, \ldots, K\}$ **do**
6: $\quad\quad\quad\quad\theta_k \leftarrow \theta_k - \lambda \frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta})$
7: $\quad$**return** $\boldsymbol{\theta}$

- Applied to Linear Regression, SGD is called the Least Mean Squares (LMS) algorithm
- We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$
$$\text{where } J^{(i)}(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2.$$

Partial Derivatives for Linear Regression (1/2)

Let $J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$
where $J^{(i)}(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2$.

$$
\begin{aligned}
\frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) &= \frac{d}{d\theta_k} \frac{1}{2}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\
&= \frac{1}{2} \frac{d}{d\theta_k}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k}(\sum_{k=1}^{K} \theta_k x_k^{(i)} - y^{(i)}) \\
&= \boxed{(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})} x_k^{(i)}
\end{aligned}
$$

Partial Derivatives for Linear Regression (2/2)

Let $J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$
where $J^{(i)}(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2$.

$$\frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) = (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}$$

Used by SGD (aka. LMS)

$$\frac{d}{d\theta_k} J(\boldsymbol{\theta}) = \frac{d}{d\theta_k} \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

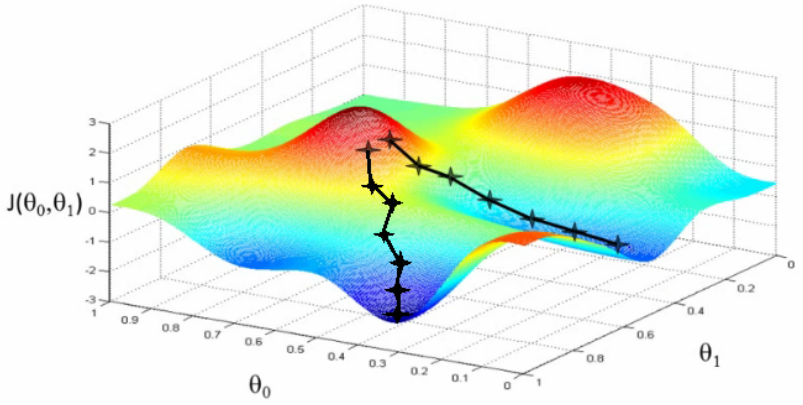
$$= \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}$$

Used by Gradient Descent

Least Mean Squares (LMS)

Algorithm 3 Least Mean Squares (LMS)

procedure LMS($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
  while not converged **do**
    for $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
      for $k \in \{1, 2, \ldots, K\}$ **do**
        $\theta_k \leftarrow \theta_k - \lambda(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}$
  return $\boldsymbol{\theta}$



A3: Closed Form (Analytical) Solution

- For some objectives we can also find the optimal solution analytically
  - This is the case for linear least-squares regression

- Basic idea:
  - Write the cost function J(θ) in a matrix form
  - Compute the derivative of J(θ) using the matrix form and set to zero
  - Then, $\theta = (X^\top X)^{-1} X^\top y$

Source: https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf

Least Squares – Summary

- Learning: Three approaches to solving $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$
- Approach 1: Gradient Descent (or Batch GD)
  - Take larger steps to opposite direction of the gradient
  - Pros: conceptually simple, guaranteed convergence
  - Cons: batch, often slow to converge
- Approach 2: Stochastic Gradient Descent (or Incremental GD)
  - Take many small steps to opposite direction of the gradient
  - Pros: memory efficient, fast convergence, less prone to local optima
  - Cons: convergence in practice requires tuning of hyper-parameters
- Approach 3: Closed form solution
  - Set derivatives equal to zero and solve for parameters
  - Pros: one shot algorithm!
  - Cons: does not scale to large datasets (matrix inverse is bottleneck)

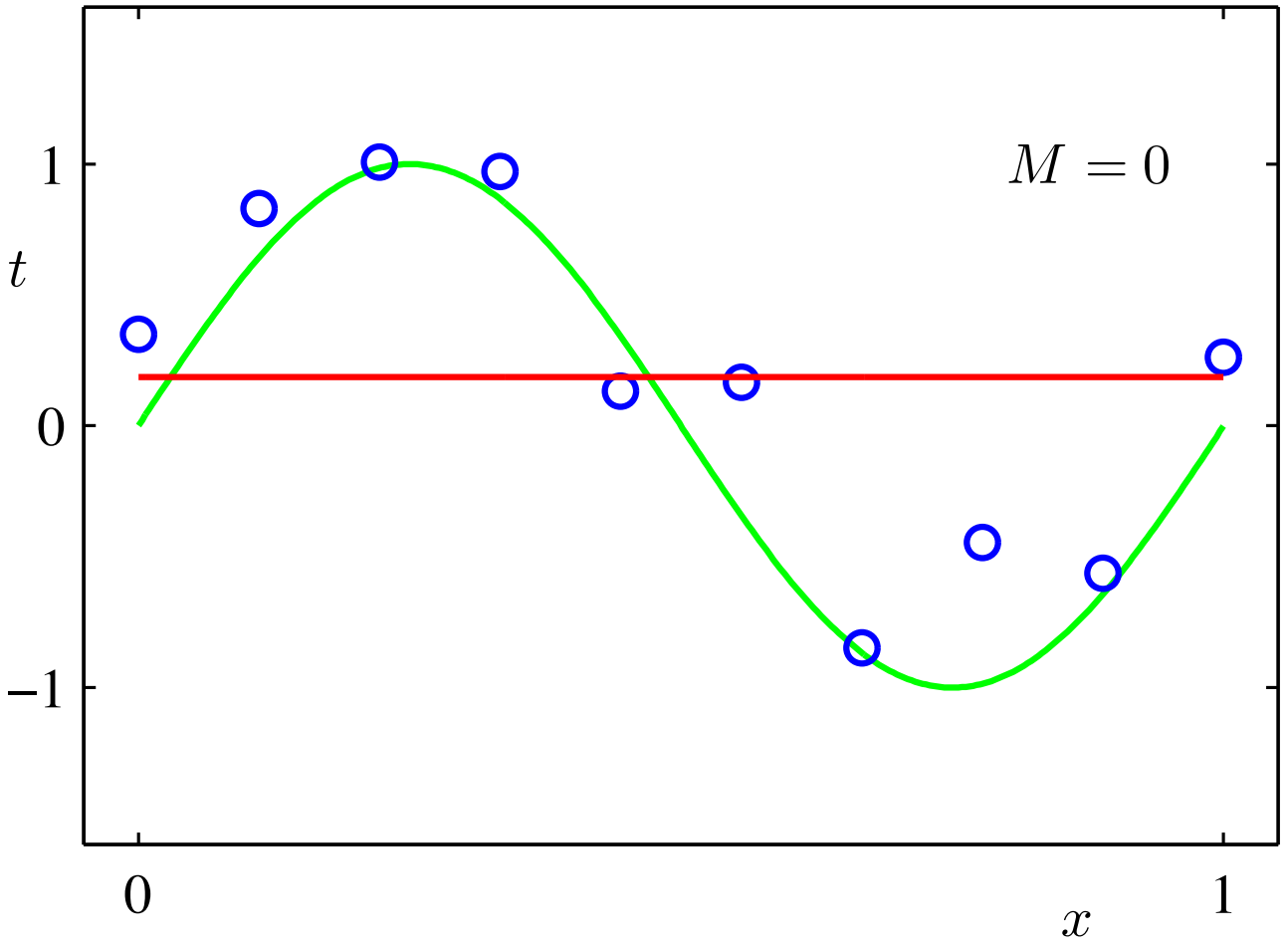
More Powerful Models – Fitting a Polynomial

- 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 x
- Example: an M-th order polynomial function of one dimensional feature x

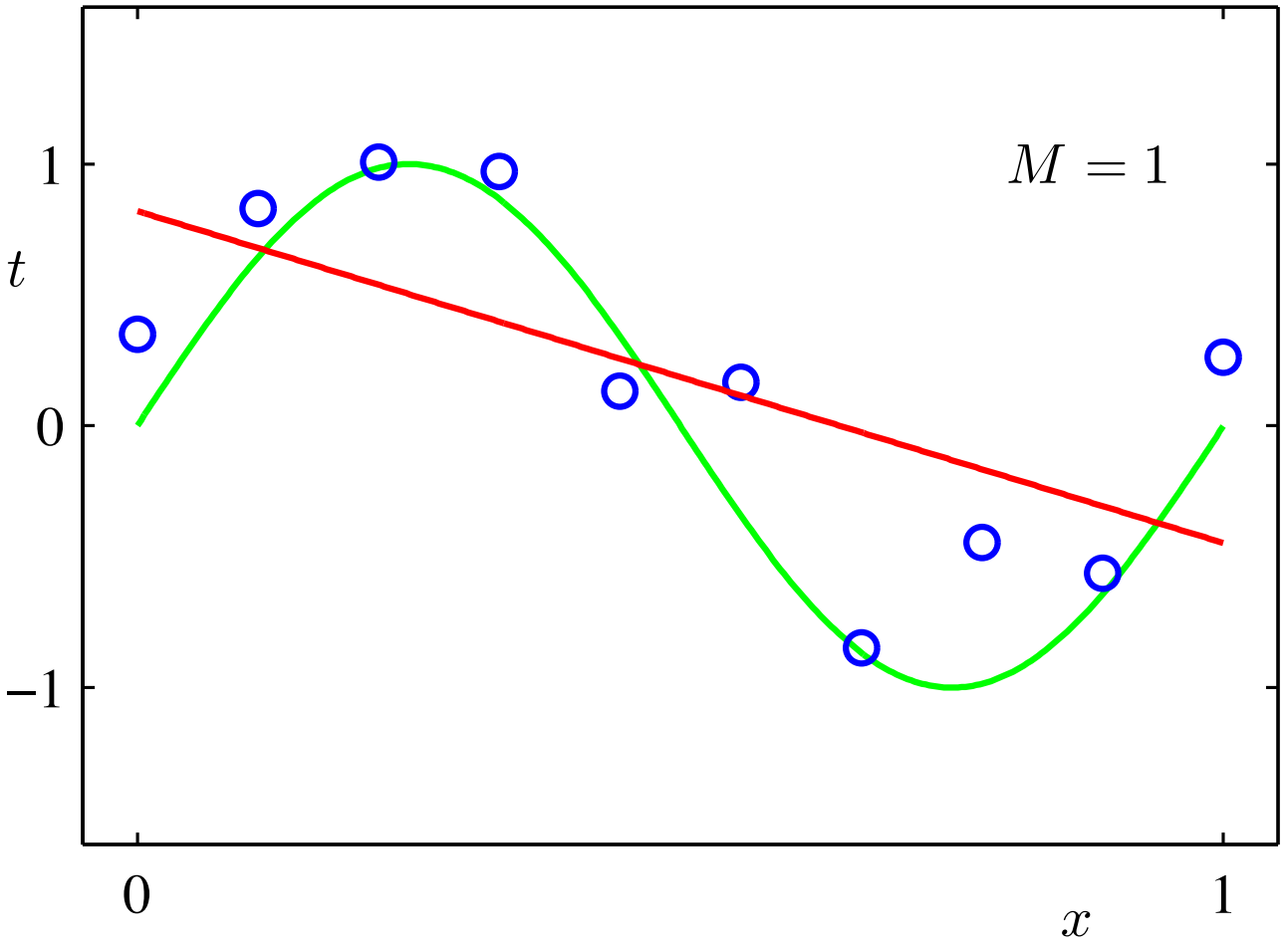
$$y(x, \theta) = \theta_0 + \sum_{i=1}^{M} \theta_j x^j$$

  - where x^j is the j-th power of x
- Use SGD to optimize the model

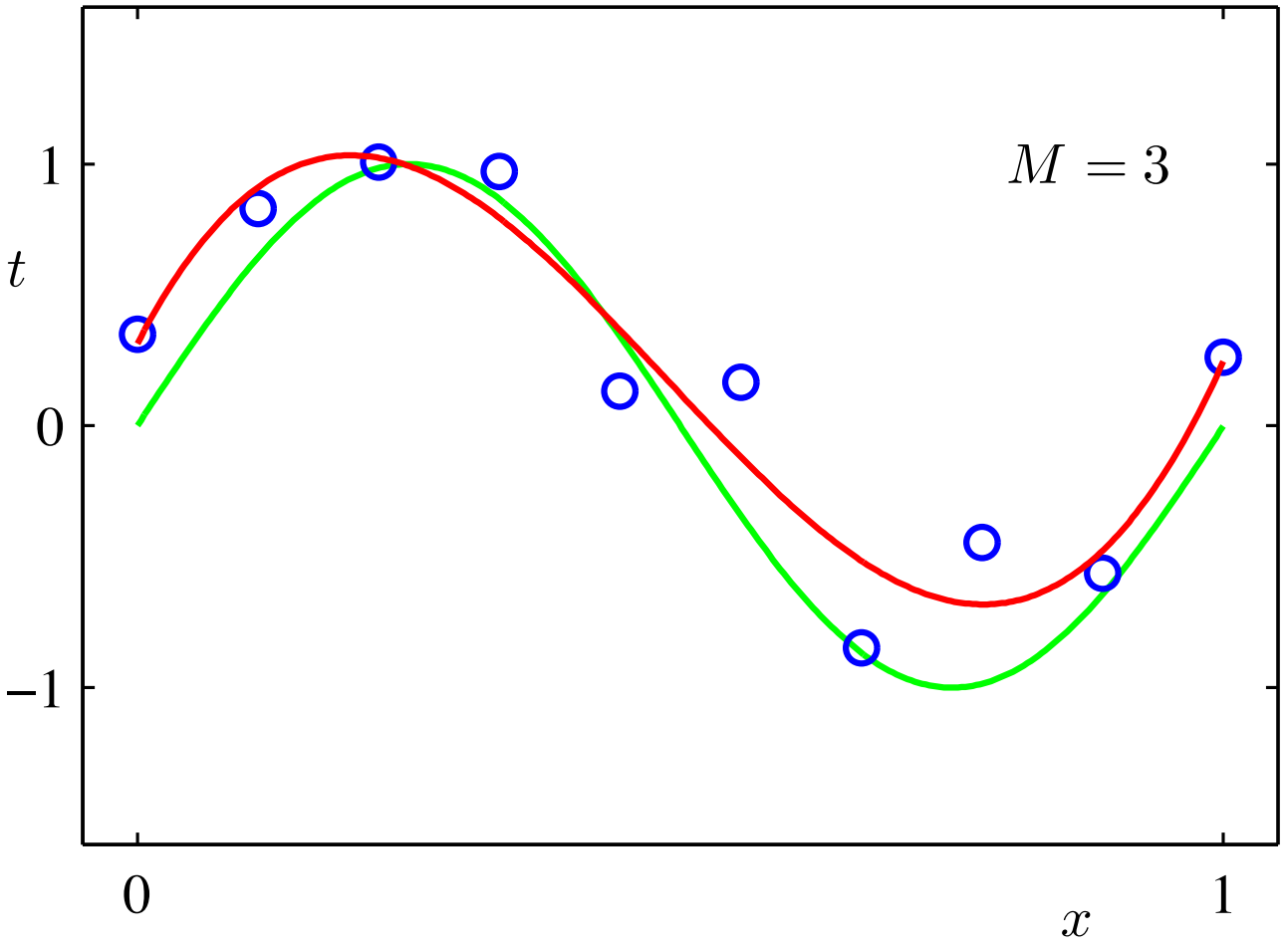
0^{th} Order Polynomial



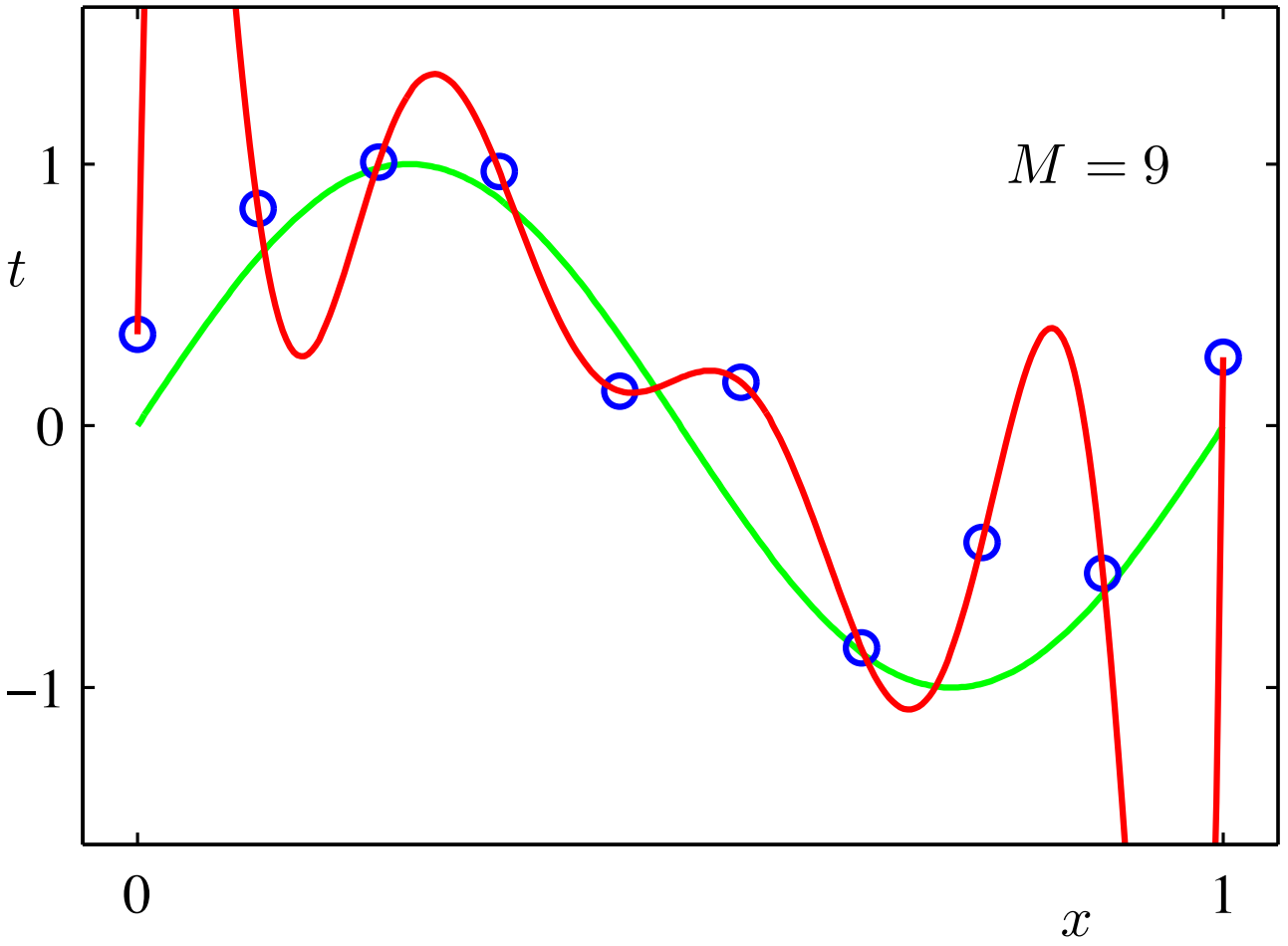
1st Order Polynomial



3rd Order Polynomial

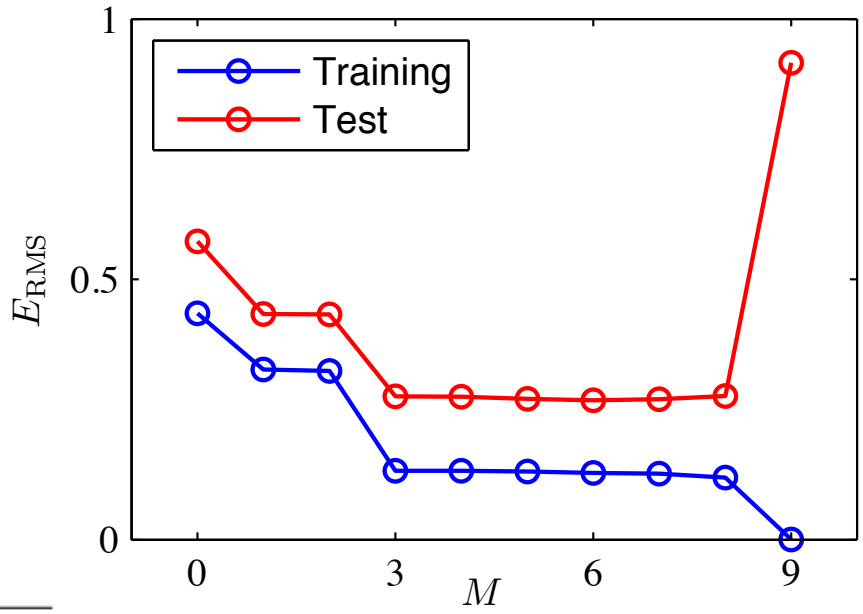


9th Order Polynomial



Generalization

- Generalization: model's ability to perform well on unseen data
- The last model with M=9 overfits the data (it models also noise)
- Not a problem if we have lots of training examples
- Let's see how the mean-squared error (MSE) behaves



$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y_t)^2}{n}}.$$

Polynomial Coefficients

Model complexity

Model parameters

	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

More complex model

Occam's Razor

- *"Among competing hypotheses (models), the one with the fewest assumptions should be selected"*

- A "simple" model is one where θ has few non-zero parameters
  - Only a few features are relevant (sparsity)
  - $||\theta||_1$ will be small (L1 norm)

- A "simple" model is one where θ is almost uniform
  - Few features are significantly more relevant than others
  - $||\theta||_2^2$ will be small (L2 norm)

Occam's Razor (wikipedia): https://en.wikipedia.org/wiki/Occam%27s_razor

Regularization

- Regularization is the process of penalizing model complexity during training

- The loss function will be similar as before, with an additional factor, called regularization term, that controls model complexity

- Two main approaches in regularized linear regression:
  - Ridge regression: adds an L2 regularization term to linear regression
  - LASSO regression: adds an L1 regularization term to linear regression

Ridge Regression

- Adds an L2 regularizer to Linear Regression

$$J_{\mathrm{RR}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_2^2$$

$$= \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2 + \lambda\sum_{k=1}^{K}\theta_k^2$$

prefers parameters close to zero

Hyperparameter λ: how much should we trade-off accuracy versus complexity

Model complexity

LASSO Regression

- Adds an L1 regularizer to Linear Regression

$$J_{\text{LASSO}}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_1$$

$$= \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)})^2 + \lambda\sum_{k=1}^{K}|\theta_k|$$

yields sparse parameters

Hyperparameter λ: how much should we trade-off accuracy versus complexity

Model complexity

scikit-learn

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

Classification as regression

Classification vs. Regression

Given $\mathcal{D} = \{x^i, y^i\}_{i=1,\ldots,n}$ find f such that $f(x) \approx y$

- Empirical error of f on the training set, given a loss function

$$\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y^i, f(\mathbf{x}^i))$$

Binary classification

$y^i \in \{0, 1\}$

Multiclass classification

$y^i \in \{0, 1, \ldots, k\}$

Regression

$y^i \in \mathbb{R}$

Classification as Regression (1/2)

- Idea: suppose that we have a binary classification problem

$$y \in \{-1, 1\}$$

- Assuming the standard model for linear regression

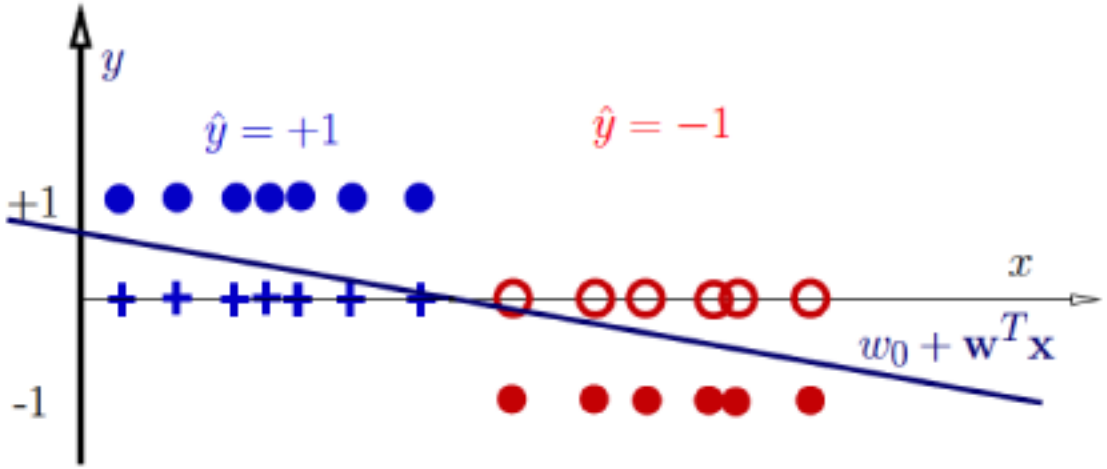
$$y(\mathbf{x}) = \theta^\top \mathbf{x}$$

- How to obtain θ?
  - Use the least-squares presented earlier

- How do I compute the label for a new example?

Classification as Regression (2/2)

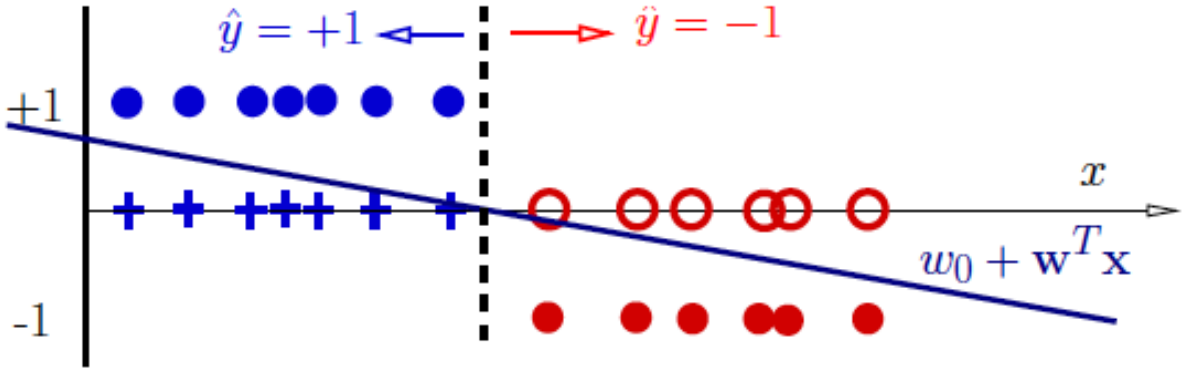
- One dimensional example. The classifier has the form $y=w_0+w^Tx$



- A reasonable decision rule

$$\hat{y} = \begin{cases} 1 & \text{if } w_0 + w^T x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Decision Rules



- How can I mathematically write this rule?

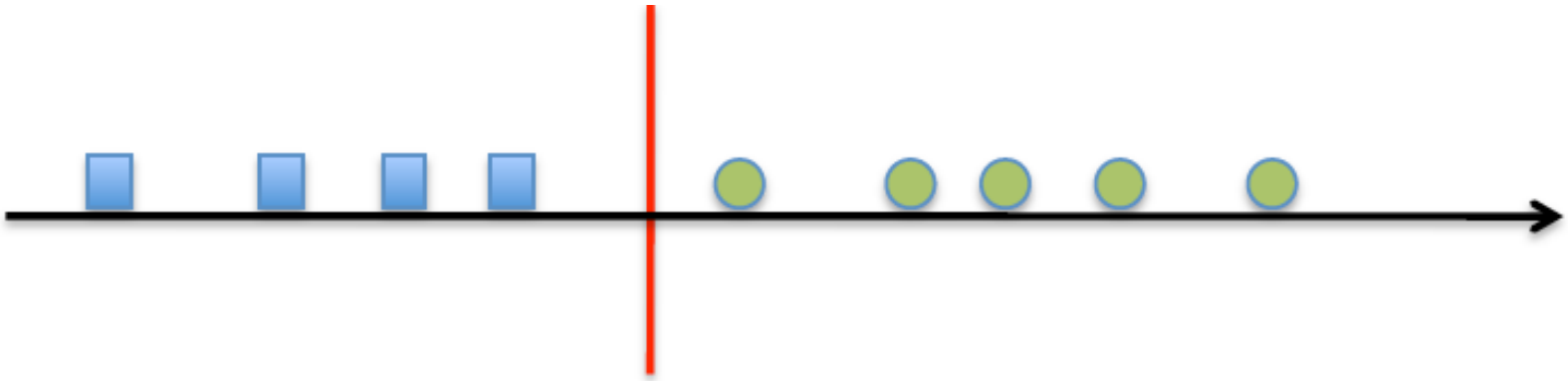
$$y(x) = \text{sign}(w_0 + w^T x)$$

- This specifies a linear classifier: it has a linear boundary (hyperplane)

$$w_0 + w^T x = 0$$

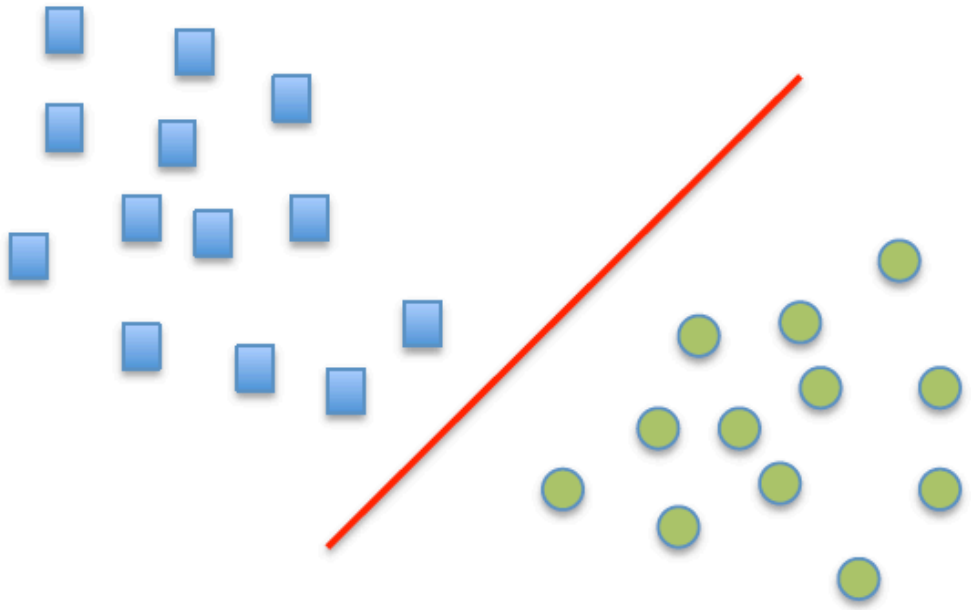
which separates the space into two half-spaces

Example in 1-D



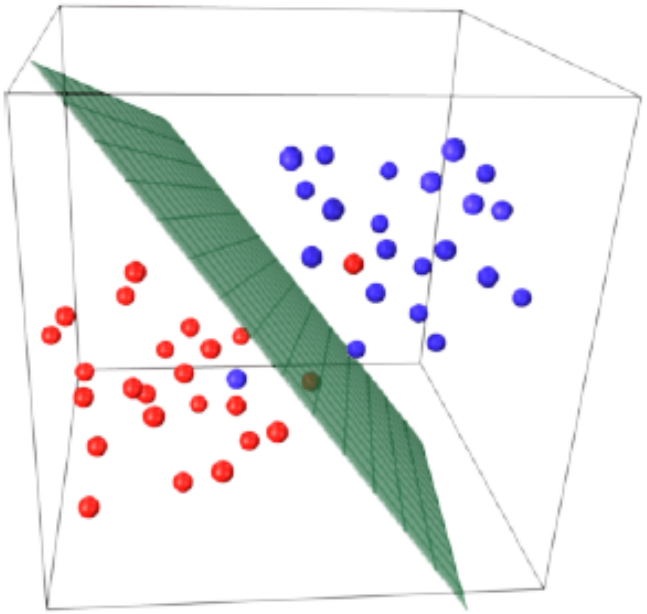
In 1-D, the linear boundary is simply a threshold

Example in 2-D



In 2-D, the linear boundary is a line

Example in 3-D



In 3-D, the linear boundary is a plane

Logistic regression

Logistic Regression (1/2)

- Focus on binary classification. Logistic regression aims to model

$$p(\text{label}|\text{data})$$

by training a classifier of the form

$$y_i = \begin{cases} 1 & \text{if } \theta^\top x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Intuitively, it does not make sense for y_i to take values larger than 1 or smaller than 0 (so, the sign() function presented before (classification as a regression task) is not very useful)

Logistic Regression (2/2)

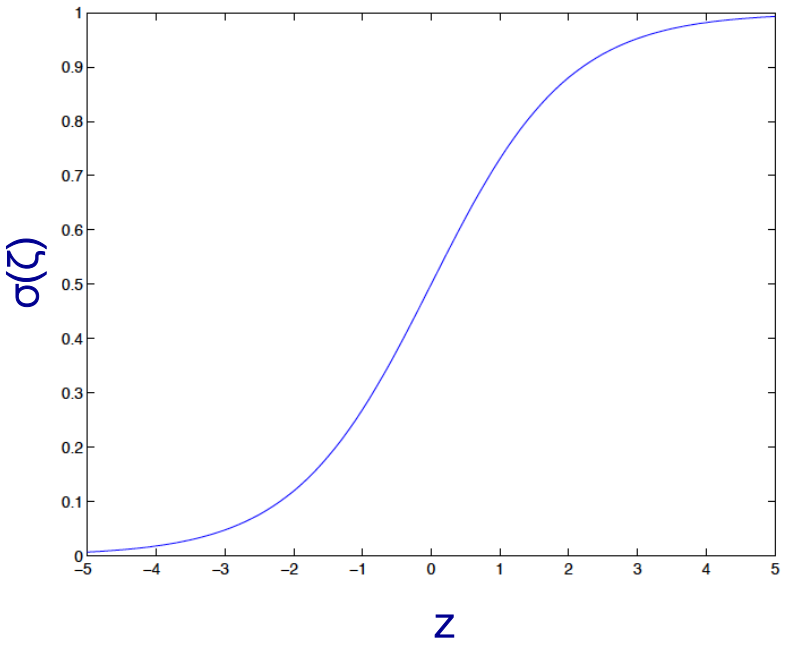
- How to turn a real-valued expression $\theta^\top x \in \mathbb{R}$ into a probability

$$p_\theta(y|x) \in [0, 1]$$

- Replace the sign() with the sigmoid or logistic function

$$y(x) = \sigma(\theta^\top x) = \frac{1}{1+e^{-\theta^\top x}}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$



Problem Formulation

- Given the logistic regression model, how do we fit θ for it?
- Probabilistic interpretation (Bernoulli distribution – success/failure):
  - $p(y=1|x;\theta) = y(x)$ [probability of success (i.e., y=1)]
  - $p(y=0)|x;\theta) = 1 - y(x)$ [probability of failure (i.e., y=0)]
  - Combine together: $p(y|x;\theta) = (y(x))^{y} (1-y(x))^{1-y}$

Recall that $y(x)$ is the hypothesis $\sigma(\theta^T x)$ (probability that the label of x is 1)

- How to do training?
  - Assuming that the m training examples were generated independently, we can write down the likelihood of the parameters

$$L(\theta) = p(y|X;\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$$

$$= \prod_{i=1}^{m} (y(x^{(i)}))^{y^{(i)}} (1 - y(x^{(i)}))^{1-y^{(i)}}$$

Likelihood Maximization

- We can learn the parameters of the model by maximizing the likelihood (Maximum Likelihood Estimation (MLE))

$$\mathcal{L}(\theta) = \log L(\theta)$$

- concave function
- log is a monotonically increasing: the log of a function achieves the maximum value at the same points as the function itself

$$= \sum_{i=1}^{m} y^{(i)} \log y(x^{(i)}) \; + \; (1 - y^{(i)}) \log(1 - y(x^{(i)}))$$

- How do we maximize the likelihood?
  - Take the derivative and do gradient ascent (maximization problem): $\theta = \theta + \lambda \nabla_\theta \mathcal{L}(\theta)$
  - Or, take the negative of the log likelihood function $-\mathcal{L}(\theta)$ and perform (stochastic) gradient descent

Source:https://homes.cs.washington.edu/~marcotcr/blog/concavity/

Summary

- Extension to multiple classes
  - Softmax regression
  - Other way to deal with multiclass classification?
- Quick to train
- Fast at classification
- Good accuracy for many simple datasets
- Resistant to overfitting
  - Regularized logistic regression

Softmax regression: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

scikit-learn

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Next Class

- Probabilistic classifiers

Thank You!