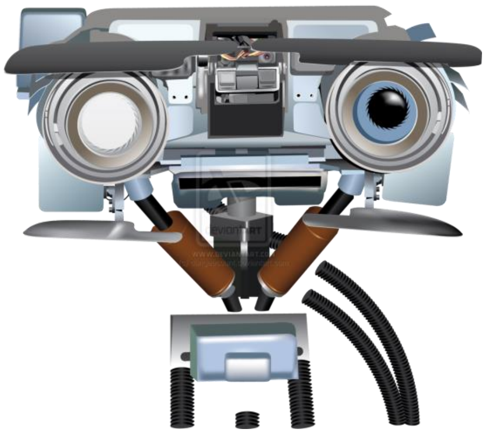


Classification

Recap

Decision Function

- Picks a class that has the maximum posterior probability of class given the feature vector *x (called Bayes classifier)*

- Models conditional distribution, $p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$

  and assign label C_1 if

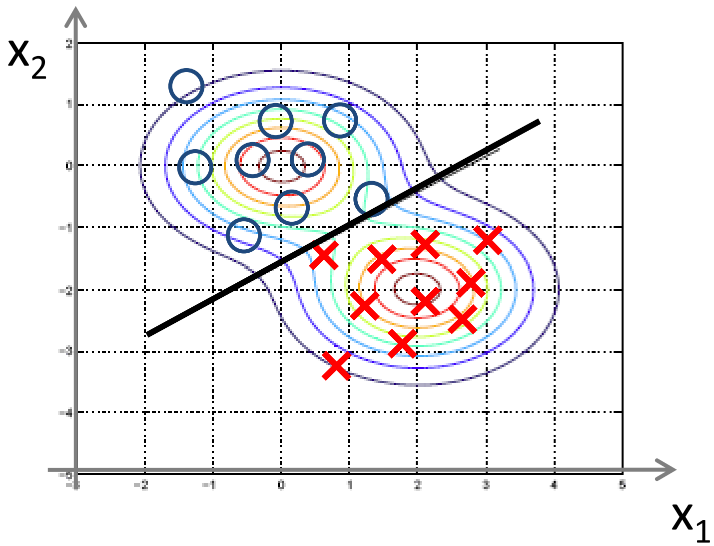
$$p(C_1|x) > p(C_2|x),$$

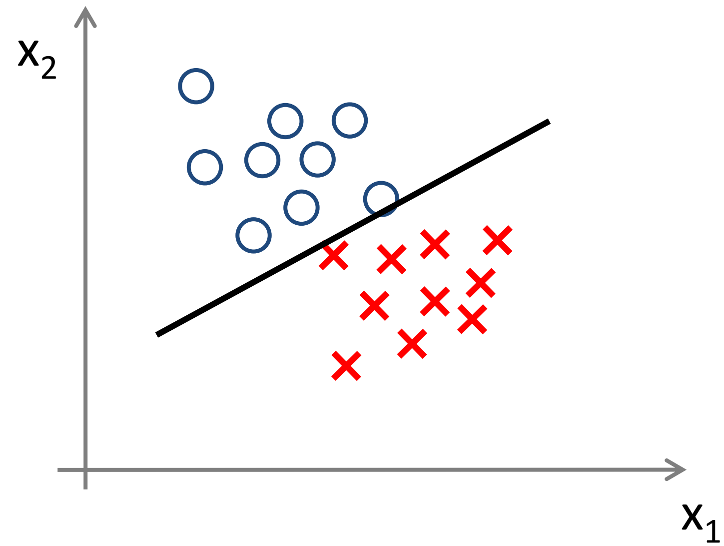
  or, equivalently, assign C_1 if the decision function $a \geq 0$, where

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

- The decision boundary between class C_1 and C_2 is all *x* where $a \geq 0$

Generative vs Discriminative





- **Generative**: model the class-conditional distribution and prior, derive decision boundary
- Pros: Can use it to generate new features
- Cons: more parameters O(n^2)

- **Discriminative**: model the decision boundary directly, e.g. Logistic Regression
- Pros: fewer parameters, O(n)
- Cons: Cannot generate new features

Do they produce the same classifier?

- Generative LDA approach will estimate μ1,μ2, and Σ to maximize joint likelihood p(x,y) and then compute the linear decision boundary, i.e., θ_j and θ_0 are functions of those parameters. In particular, θ_j and θ_0 are not completely independent.

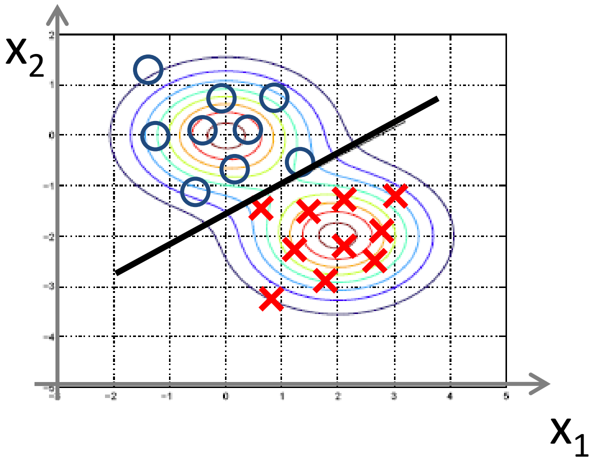
- Discriminative approach (logistic regression) will directly estimate θ_j and θ_0, without assuming any constraints between them, by maximizing conditional likelihood p(y|x)

- The two methods will give different decision boundaries, even both are linear.

Linear discriminant analysis

Assume

- $p(C_k) = \pi_k,\ \pi_k \geq 0,\ \sum_{k=1}^{2} \pi_k = 1$
- $p(x|C_k)$ is a Gaussian distribution

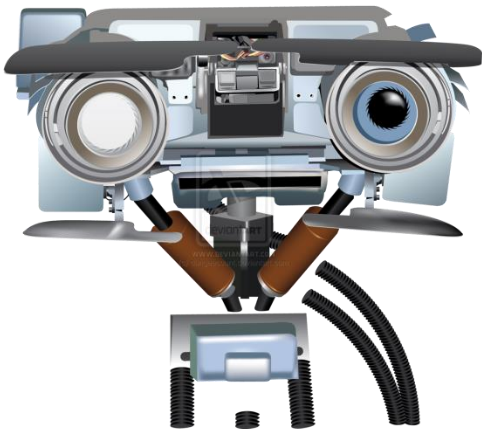


$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}$$

- p is the dimension of x and Σ_k is the covariance matrix. The vector x and the mean vector μ_k are both column vectors.

- $\Sigma_k = \Sigma,\ \forall k$. Making this assumption, decision boundary becomes linear

$$\log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l) = 0$$



Classification

Multiclass

Multiclass Classification

- Two choices

  "one versus the rest"

  "one versus one"

- Pros and cons of each approach

  one versus the rest: only needs to train K classifiers. Makes a huge difference if you have a lot of classes to go through.

  one versus one: only needs to train a smaller subset of data (only those labeled with those two classes would be involved). Makes a huge difference if you have a lot of data to go through.

- Bad about both of them

  Combining classifiers' outputs can be a bit tricky.

Multiclass LDA

Generative model for multiclass classification

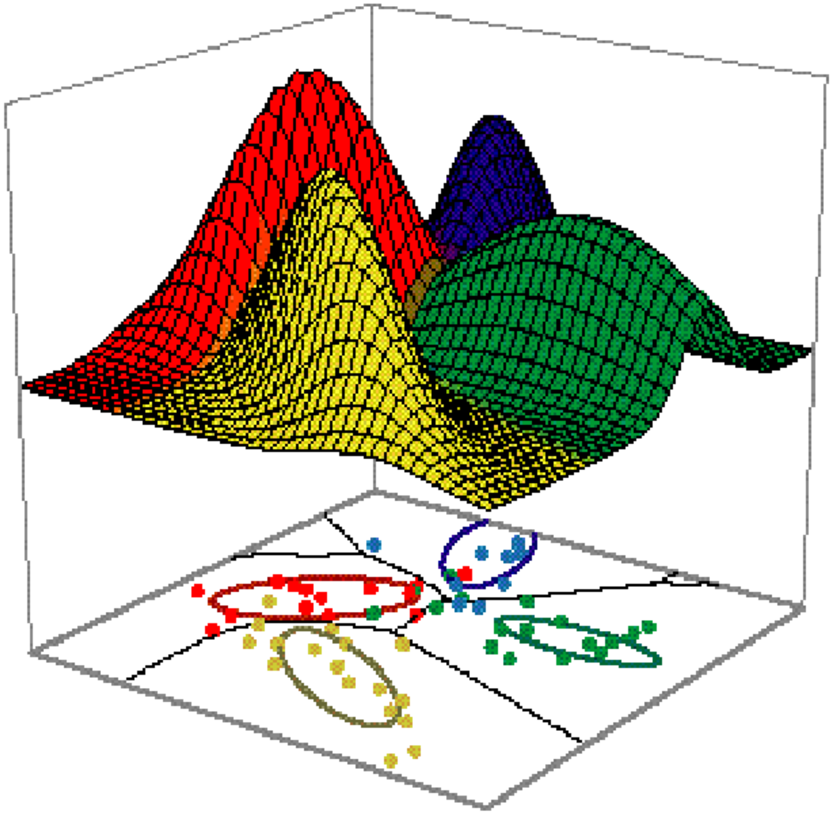
$$p(\boldsymbol{x}, y) = p(y)p(\boldsymbol{x}|y) = p(y)\frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_y)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_y)}$$

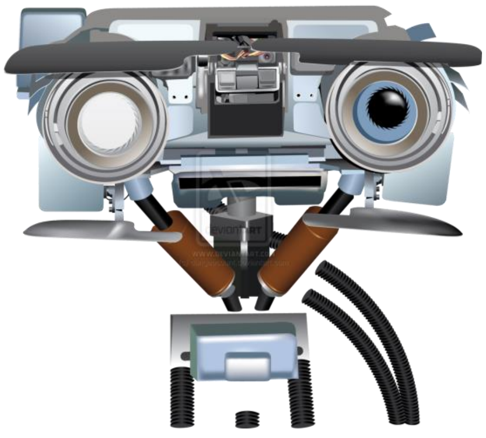
We use multiple multivariate Gaussian distributions, one for each class

- Namely, there are K Gaussians, with different means μk but the same covariance matrix Σ.
- Note that y denotes 1,2,...,K, corresponding to C1,C2,...,CK respectively.
- Thus, μy is the class Cy's mean parameter.
- As in linear discriminant analysis, we assume equal covariance matrix across all classes.

Unequal Covariances

- more general case of unequal covariances (here shown for four classes)
- the decision hypersurface is no longer a hyperplane, i.e. it is nonlinear



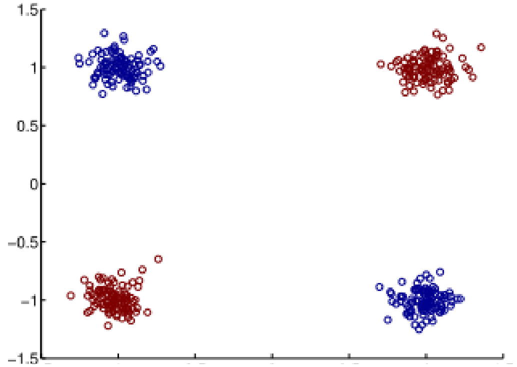


Regularization

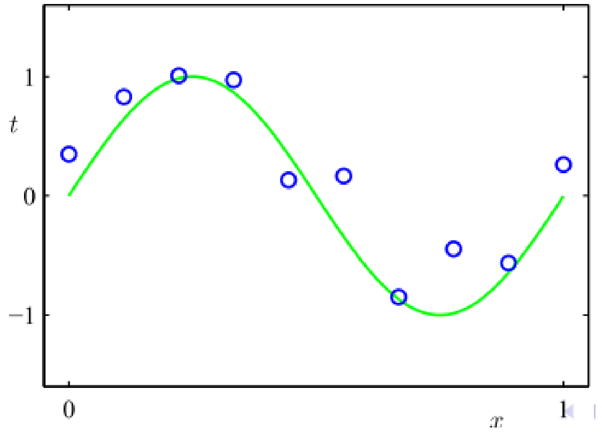
Overfitting

What to do if data is nonlinear?

Example of nonlinear classification



Example of nonlinear regression

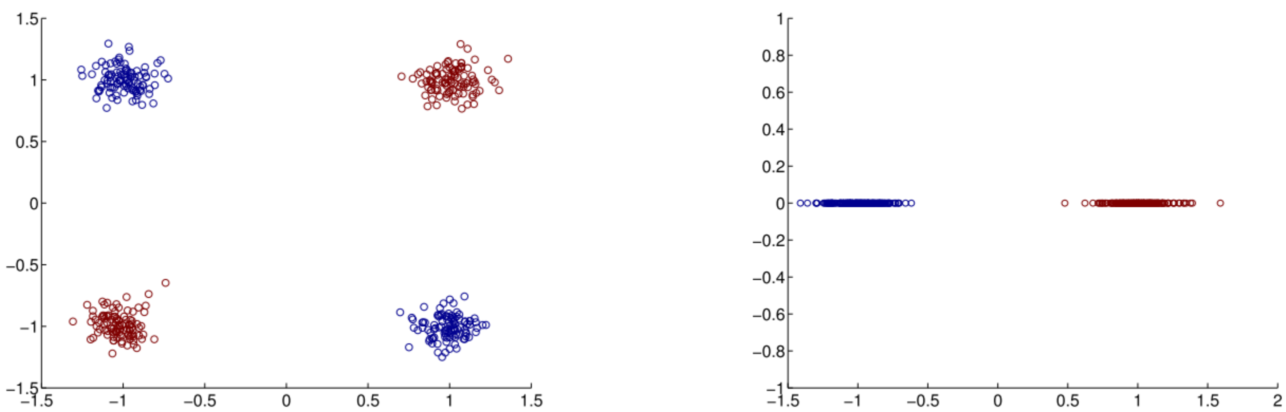


Nonlinear basis functions

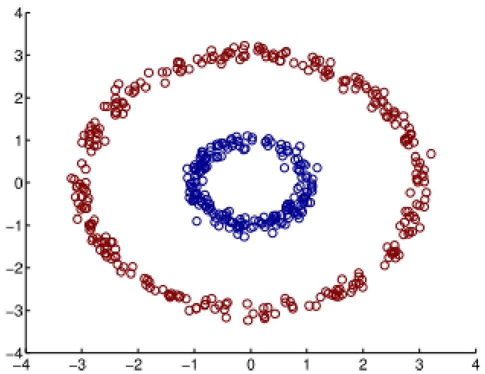
Transform the input/feature

$$\varphi(x) : x \in R^2 \rightarrow z = x_1 \cdot x_2$$

Transformed training data: linearly separable!



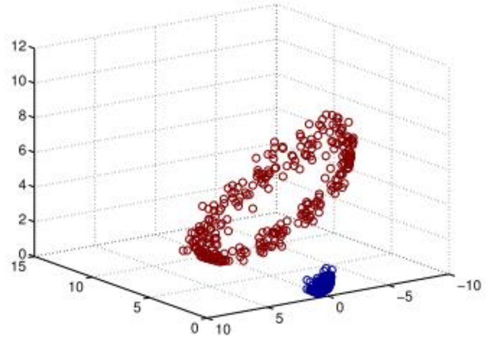
Another example



How to transform the input/feature?

$$\varphi(x) : x \in R^2 \rightarrow z = \begin{bmatrix} {x_1}^2 \\ x_1 \cdot x_2 \\ {x_2}^2 \end{bmatrix}$$

Transformed training data: linearly separable



Intuition: suppose $\theta = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Then $\theta^T z = {x_1}^2 + {x_2}^2$

i.e., the sq. distance to the origin!

General nonlinear mapping

- We can use a nonlinear mapping

$$\varphi(x) : x \in R^N \ \rightarrow \ z \in R^M$$

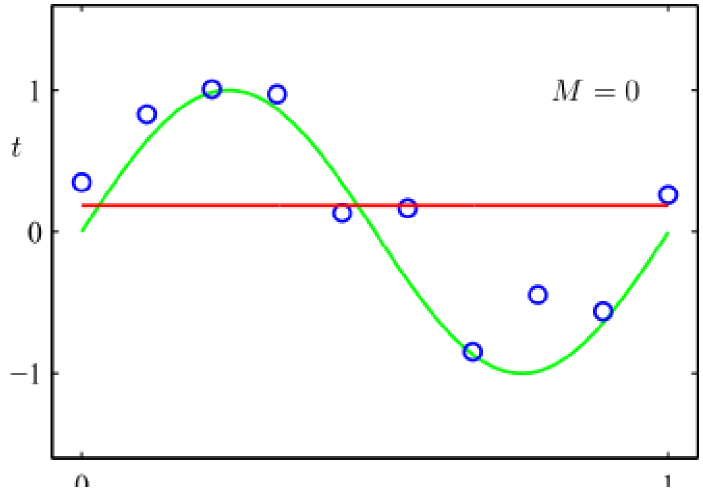
- where M is the dimensionality of the new feature/input z (or $\varphi(x)$)
- Note that M could be either greater than D or less than, or the same

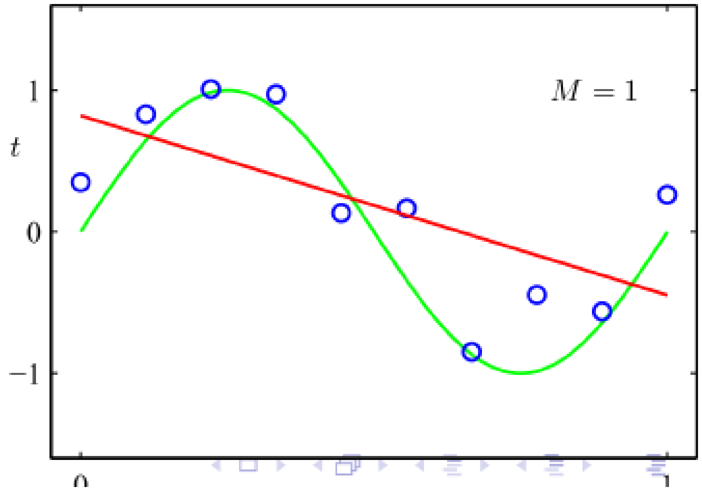
Example with regression

Polynomial basis functions

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}$$

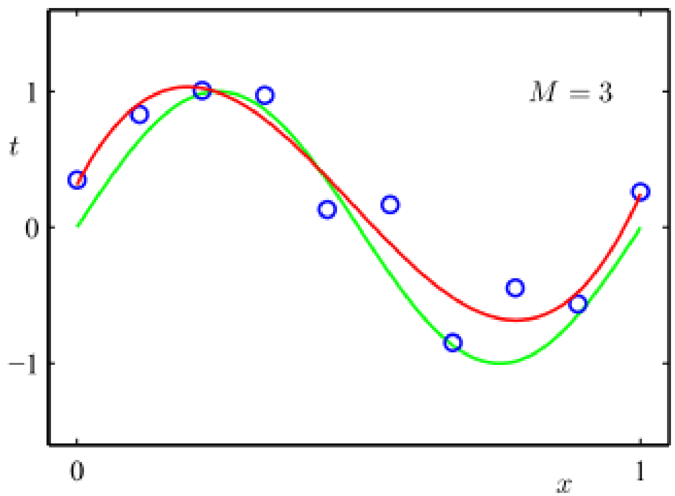
Fitting samples from a sine function: *underrfitting* as $f(x)$ is too simple



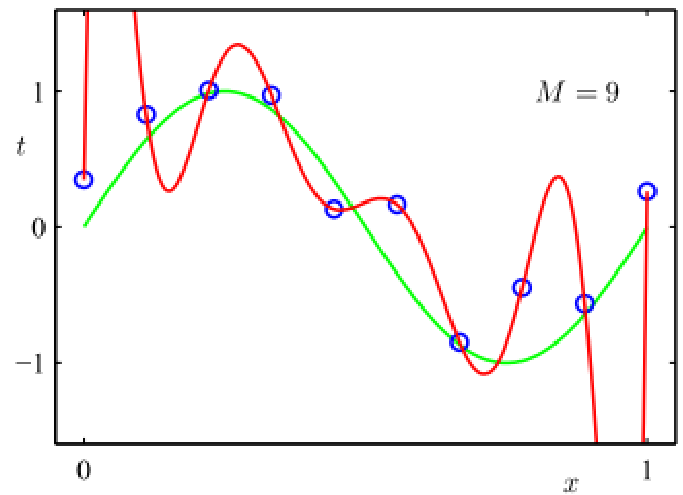


Add more nonlinear basis functions

M=3



M=9: *overfitting*



- Being too adaptive leads to better results on the training data, but not so great on data that has not been seen!

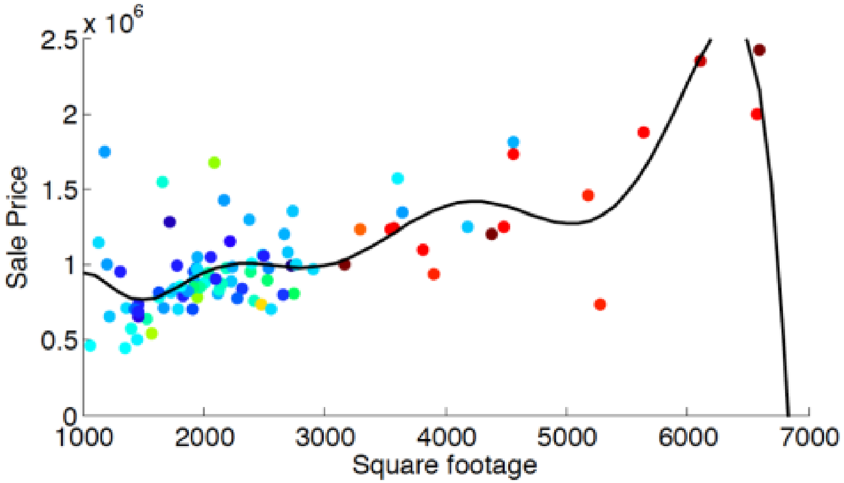
Overfitting

Parameters for higher-order polynomials are very large

	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

Overfitting disaster

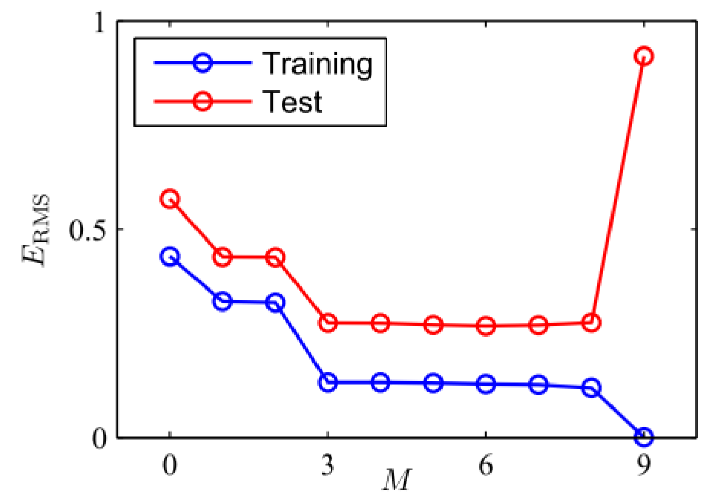
Fitting the housing price data with M = 3



Note that the price would goes to zero (or negative) if you buy bigger houses!
This is called poor generalization/overfitting.

Detecting overfitting

Plot model complexity versus objective function on test/train data



As model becomes more complex, performance on training keeps improving while on test data it increases

Horizontal axis: measure of model complexity

In this example, we use the maximum order of the polynomial basis functions.

Vertical axis: For regression, it would be SSE or mean SE (MSE)

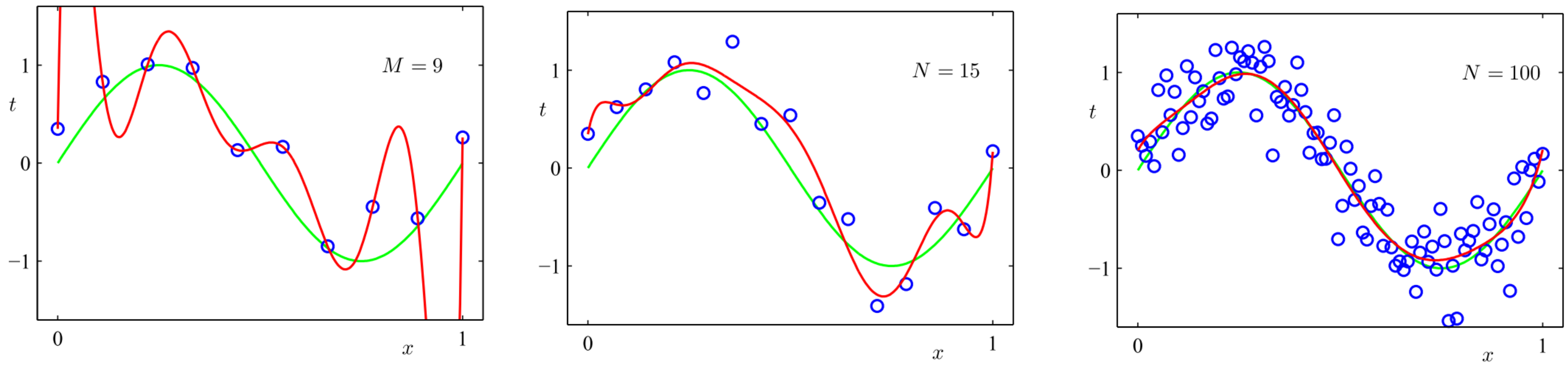
For classification, the vertical axis would be classification error rate or cross-entropy error function

Overcoming overfitting

- Basic ideas
  - Use more training data
  - Regularization methods
  - Cross-validation

Solution: use more data

M=9, increase N



What if we do not have a lot of data?

Solution: Regularization

- Use regularization:
  - Add $\lambda\|\theta\|_2{}^2$ term to SSE cost function
  - Penalizes large θ
  - λ controls amount of regularization
- Next, we will derive regularized linear regression from Bayesian linear regression

Go to Adrew Ng's lecture on Regularization