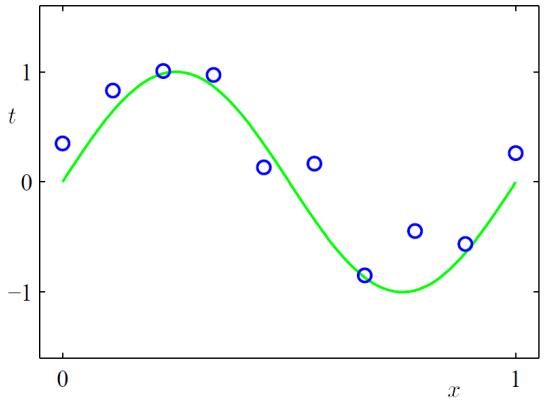
# Choosing Predictors

CPS 570

Ron Parr

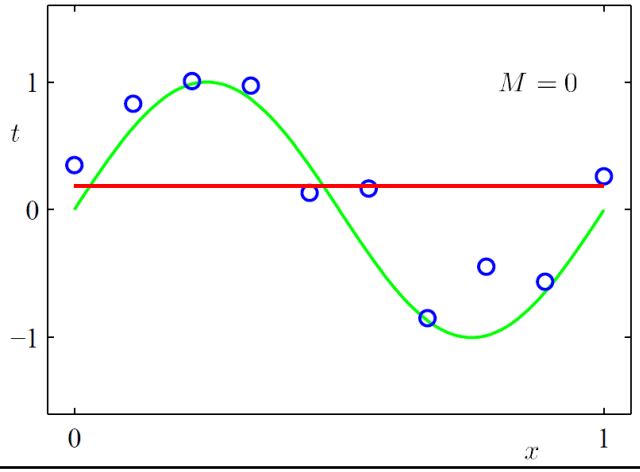
Regression figures provided by Christopher Bishop and © 2007 Christopher Bishop

## What is the Best Choice of Polynomial?

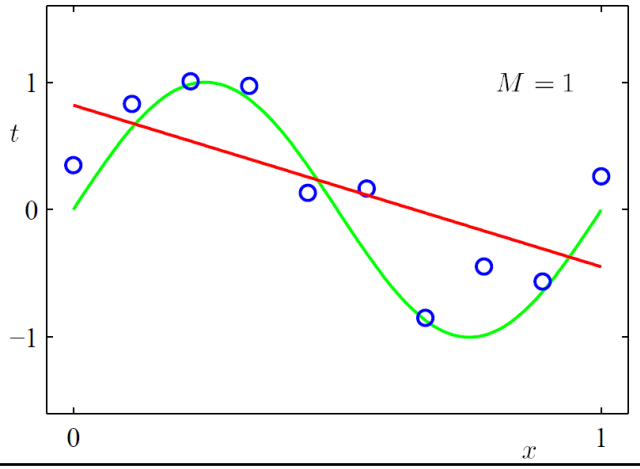



Noisy Source Data

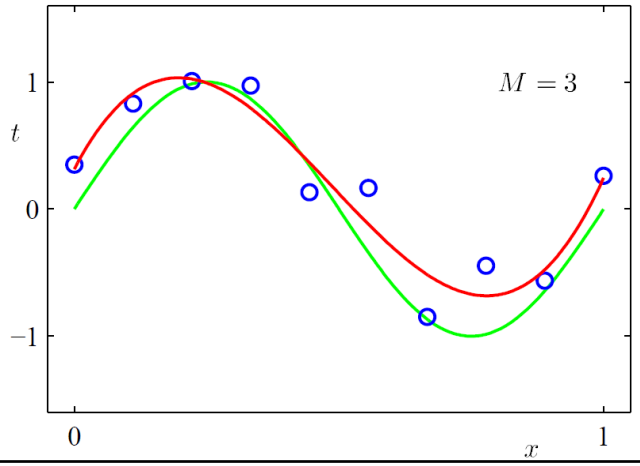
## Degree 0 Fit



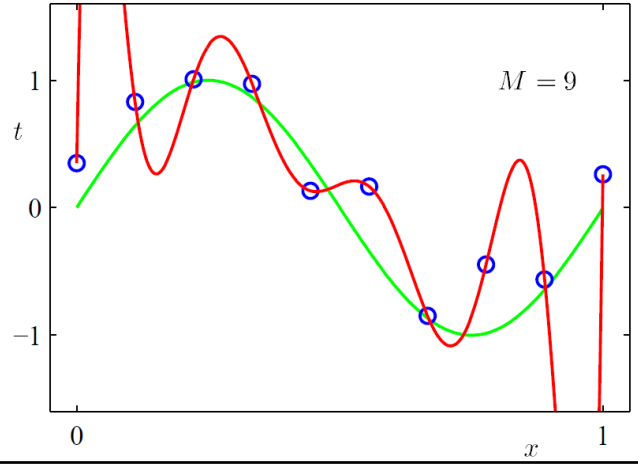

## Degree 1 Fit



## Degree 3 Fit

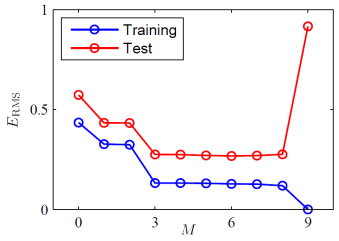



## Degree 9 Fit



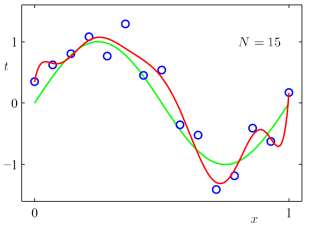

## Observations

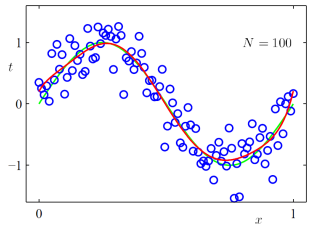
- Degree 3 is the best match to the source
- Degree 9 is the best match to the samples
- We call this **over-fitting**
- Performance on test data:




## What went wrong?

- Is the problem a bad choice of polynomial?
- Is the problem that we don't have enough data?
- Answer: Yes



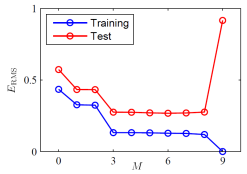



## Methods for Choosing Features

- Cross validation
- Regularization
  - Non-Bayesian ($L_1$, $L_2$, etc.)
  - Bayesian

## Cross Validation

- Suppose we have many possible hypothesis spaces, e.g., different degree polynomials
- Recall our empirical performance results:




- Why not use the data to find min of the red curve?

## Implementing Cross Validation

- Many possible approaches to cross validation
- Typical approach divides data into k equally sized chunks:
  - Do k instances of learning
  - For each instance hold out 1/k of the data
  - Train on (k-1)/k fraction of the data
  - Test on held out data
  - Average results
- Can also sample subsets of data with replacement
- Cross validation can be used to search range of hypothesis classes to find where **overfitting** starts

## Problems with Cross Validation

- Cross validation is a sound method, but requires a lot of data and/or is slow
- Must trade off two factors:
  - Want enough data within each run
  - Want to average over enough trials
- With scarce data:
  - Choose k close to n
  - Almost as many learning problems as data points
- With abundant data (then why are you doing cross validation?)
  - Choose k = a small constant, e.g., 10
  - Not too painful if you have a lot of parallel computing resources and a lot of data, e.g., if you are Google

## Regularization

- Cross validation may also be impractical if range of hypothesis classes is not easily enumerated a searched iteratively
- Regularization aims to avoid overfitting, while
  - Avoiding speed penalty of cross validation
  - Not assuming an ordering on hypothesis spaces
- …but you still need to do some kind of cross-validation in the end.

## Regularization

- Idea: Penalize overly complicated answers
- Ordinary regression minimizes:

$$\sum_{i=1}^{M}(y(x^{(i)};\mathbf{w})-t_i)^2$$

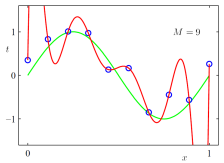
- $L_2$ Regularized regression minimizes:

$$\lambda\|\mathbf{w}\|_2+\sum_{i=1}^{M}(y(x^{(i)};\mathbf{w})-t^{(i)})^2$$

- Note: May exclude constants form the norm

## $L_2$ Regularization: Why?

$$\lambda\|\mathbf{w}\|_2+\sum_{i=1}^{M}(y(x^{(i)};\mathbf{w})-t^{(i)})^2$$




- For polynomials, extreme curves typically require extreme values
- In general, balances using full expressiveness of hypothesis space with performance
- Problem: How to choose $\lambda$ (cross validation?)

## The $L_2$ Regularized Solution

- Minimize:

$$\lambda\|\mathbf{w}\|_2+\sum_{i=1}^{M}(y(x^{(i)};\mathbf{w})-t^{(i)})^2$$

- Set gradient to 0, solve for w for features $\Phi$:

$$\mathbf{w}=(\Phi^T\Phi+\lambda I)^{-1}\Phi^T\mathbf{t}$$

- Compare with unregularized solution

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

## A Bayesian Perspective

- Suppose we have a space of possible hypotheses H
- Which hypothesis has the highest posterior:

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)}$$

- P(D) does not depend on H; maximize numerator
- Uniform P(H) is called Maximum Likelihood solution (model for which data has highest prob.)
- P(H) can be used for regularization

## Maximum Likelihood

- For many models, the empirical mean is also the maximum likelihood solution
- Suppose:
  - Data normally distributed
  - Unknown mean, variance
  - IID samples

$$P(D \mid H) = P(t^{(1)} \ldots t^{(m)} \mid \mu, \sigma)$$
$$= \prod_{i=1}^{m} \frac{e^{-\frac{(t^{(i)}-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$$

## Priors for Gaussians

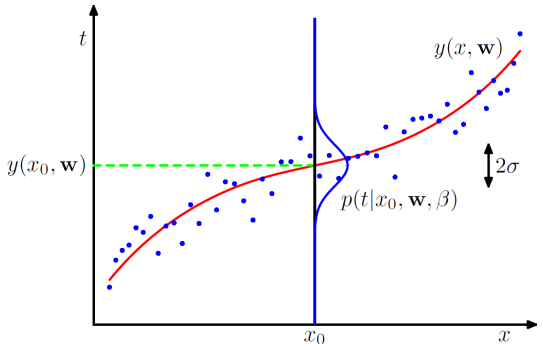
- Recall Bayes rule:

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)}$$

- Does it make sense to have a P(H) for Gaussians?
- Yes: Corresponds to some prior knowledge about the mean or variance
- Would like this knowledge to have a mathematically convenient form

## Bayesian Regression

- Assume that, given x, noise is Gaussian
- Homoscedastic noise model



## Maximum Likelihood Solution

$$P(D \mid H) = P(t^{(1)} \ldots t^{(m)} \mid y(x;\mathbf{w}),\sigma)$$

$$= \prod_{i=1}^{m} \frac{e^{\frac{-(t^{(i)} - y(x;w))^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$$

- ML fit for mean is just linear regression fit
- ML fit for mean does not depend upon σ

## Bayesian Solution

- Introduce prior distribution over weights

$$p(H) = p(w \mid \alpha) = N(w \mid 0, \frac{1}{\alpha} I)$$

- Posterior now becomes:

$$P(D \mid H)P(H) = P(t^{(1)} \ldots t^{(m)} \mid y(x;\mathbf{w}),\sigma)P(\mathbf{w})$$

$$= \prod_{i=1}^{m} \frac{e^{\frac{-(t^{(i)} - y(x;w))^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} \frac{e^{\frac{-\alpha \mathbf{w}^T \mathbf{w}}{2}}}{\frac{2\pi}{\alpha}^{(k+1)/2}}$$

## Comparing Regularized Regression with Bayesian Regresion

- $L_2$ Regularized Regression minimizes:

$$\lambda \left\| \mathbf{w} \right\|_2 + \sum_{i=1}^{M} (y(x^{(i)};\mathbf{w}) - t^{(i)})^2$$

- Bayesian Regression maximizes:

$$\prod_{i=1}^{m} \frac{e^{\frac{-(t^{(i)} - y(x;w))^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} \frac{e^{\frac{-\alpha \mathbf{w}^T \mathbf{w}}{2}}}{\frac{2\pi}{\alpha}^{(k+1)/2}}$$

- Observation: Take log of Bayesian regression criterion and these become identical (up to constants) with $\lambda = \alpha$.

## What $L_2$ Regularization Does

- Also known as
  - "shrinkage"
  - Tikhanov Regularization

$$\lambda \left\| \mathbf{w} \right\|_2 + \sum_{i=1}^{M} (y(x_i;\mathbf{w}) - t_i)^2$$

- Trades performance on training set for lower parameter values
- Squaring favors lots of small weights over a few large ones

## LASSO

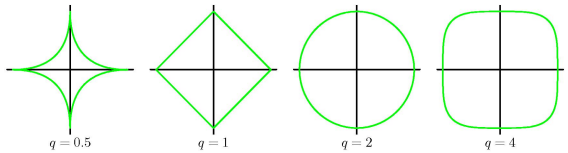
- The general form of regularized regression:

$$\lambda f(\|\mathbf{w}\|) + \sum_{i=1}^{M} (y(x^{(i)};\mathbf{w}) - t_i)^2$$

- What if we used the 1-norm instead 2-norm for f:

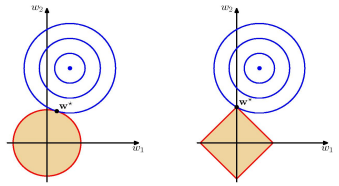
$$\lambda \|\mathbf{w}\|_1 + \sum_{i=1}^{M} (y(x^{(i)};\mathbf{w}) - t^{(i)})^2$$

## Norm Balls




q-norm balls for different values of q

## Regularization and Norm Balls





- $L_2$ ball
  - Smooth
  - Chance of hitting 0 values is vanishingly small
- $L_1$ ball
  - Pointy
  - Chance of hitting all non-0 values vanishingly small

## What $L_1$ Regularization Does

- Trades performance on training data against $L_1$ norm of the weights
- Favors sparse solutions
- Relationship to compressed sensing:
  - Compressed sensing aims to find a sparse combination of basis functions that are consistent with observations
  - Formulated as an $L_1$ minimization problem

## Implementing LASSO

- Several different approaches are possible:
  - Minimize weighted sum of training error and L1 norm on weights
  - Minimize training error subject to a strict bound on L1 norm of weights
- Both can easily be implemented as a convex program
- Also possible to solve incrementally using an algorithm called LARS

## Working with 1-norm

- Suppose you want to minimize the 1-norm of a vector **x** within a linear program
- Minimize: $\sum_i e_i$
- Subject to: $\forall i : e_i \geq x_i$
  $: e_i \geq -x_i$

## Bayesian Interpretation

- Note that we can always come up with a Bayesian interpretation of any regularization parameter f:

$$\lambda f(\|\mathbf{w}\|) + \sum_{i=1}^{M} (y(x^{(i)}; \mathbf{w}) - t_i)^2$$

- Assume Gaussian noise
- Choose a prior on the weights which differentiates to f
- Lasso = assumption of Laplace (double exponential) distribution:

$$p(x \mid \mu, b) = \frac{1}{b} e^{-\frac{|x-\mu|}{b}}$$

## Bayesian vs. Non-Bayesian Regularization

- Is there really a difference?
- Bayesian view is arguably more justified, but
- Can't we always find a Bayesian interpretation of anything by taking an integral and calling it a prior?
- But do all priors have frequentist counterparts?
- What about hyper-priors?
  - Priors on priors
  - Actually makes sense if # of parameters is decreasing
  - Actually works!

## More thoughts on Bayesian approaches

- Priors open the door to a rich and potentially well motivated way to introduce prior knowledge
- Hyperpriors may reduce or completely eliminate the need for cross validation

- Main drawback: Many priors do not reduce to clean optimization problems

## Which is better $L_1$ or $L_2$?

- No clear winner
- $L_2$:
  - Easier to implement
  - Sometimes gives better performance on test data
- $L_1$:
  - More expensive (no direct solution)
  - Gives more understandable answers
  - Good choice if you have reason to believe the true answer is sparse

## Why not $L_0$ norm?

- $L_0$ norm is the best norm to use for sparseness
- Counts number of non-zero parameters
- Problem: This is not tractable

- In many scenarios, e.g. compressed sensing, it has been shown the $L_1$ is a reasonable approximation to $L_0$

## Other ways to get sparseness

- Forward selection:
  - Start with a small feature set
  - Gradually add features until performance (checked with cross validation) stops improving
- Backward elimination:
  - Start with all features
  - Gradually remove features
- Issues:
  - Both methods can be slow
  - Both methods are greedy

## Conclusions

- Regularization trades training set performance against solution complexity
- Can reduce the need for cross validation, but
  - Regularization parameters still must be chosen
  - Hyperpriors might help here
- $L_2$ regularization favors many small weights
- $L_1$ regularization favors few/sparse weights
- $L_2$ and $L_1$ both have Bayesian counterparts