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:

  ![Graph](image.png)

  • 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 may require a lot of data and/or can be 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 (hold out as few as one point)
  – 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

• CV may be impractical if range of hypothesis classes not easily enumerated and searched

• Regularization aims to avoid overfitting, while
  – Avoiding speed penalty of cross validation
  – Not assuming an ordering on hypothesis spaces

• ...but perhaps you still need to do some kind of cross-validation in the end.

Regularization – “Shrinkage”

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

\[ \sum_{i=1}^{N} (y(x^{(i)}, w) - t_i)^2 \]

• L₂ Regularized regression minimizes:

\[ \lambda \|w\|_2^2 + \sum_{i=1}^{N} (y(x^{(i)}, w) - t_i)^2 \]

• Note: May exclude coefficient constant a constant term from the norm
L₂ Regularization: Why?

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

The L₂ Regularized Solution

- Minimize:
  \[
  \lambda \|w\|_2^2 + \sum_{i=1}^M (y(x^{(i)}; w) - t^{(i)})^2
  \]
- Set gradient to 0, solve for \( w \) for features \( \Phi \):
  \[
  w = (\Phi^\top \Phi + \lambda I)^{-1} \Phi^\top t
  \]
- Compare with unregularized solution
  \[
  w = (\Phi^\top \Phi)^{-1} \Phi^\top t
  \]
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

Bayesian Regression

• Assume that, given \( x \), noise is Gaussian
Maximum Likelihood Solution

\[
P(D | H) = P(t^{(1)} \ldots t^{(m)} | y(x; w), \sigma) = \prod_{i=1}^{m} e^{-\frac{(t^{(i)} - y(x; w))}{2\sigma^2}} = e^{-\frac{(t^{(i)} - y(x; w))}{2\sigma^2}} \sqrt{2\pi\sigma^2}
\]

* ML fit for mean is just linear regression fit
* ML fit for mean does not depend upon \( \sigma \)

Bayesian Solution With Prior on \( H \)

* Introduce prior distribution over weights

\[
p(H) = p(w | \alpha) = N(w | 0, \frac{1}{\alpha})
\]

* Numerator now becomes:

\[
P(D | H)P(H) = P(t^{(1)} \ldots t^{(m)} | y(x; w), \sigma)P(w) = \prod_{i=1}^{m} e^{-\frac{(t^{(i)} - y(x; w))}{2\sigma^2}} - \frac{1}{\alpha} e^{-\frac{\|w\|^2}{2}}
\]
Comparing Regularized Regression with Bayesian Regression

- L2 Regularized Regression minimizes:
\[ \lambda \|w\|^2 + \sum_{i=1}^{m} (y(x_i;w) - t_i)^2 \]

- Bayesian Regression maximizes:
\[ \prod_{i=1}^{m} \frac{e^{-\frac{(t_i-y(x_i;w))^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} \cdot \frac{e^{-\frac{cw^T w}{2}}}{2\pi^{(k+1)/2}} \]

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

Summary: What L2 Regularization Does

- Also known as Tikhanov Regularization
\[ \lambda \|w\|^2 + \sum_{j=1}^{M} (y(x_j;w) - t_j)^2 \]

- Tries to avoid overfitting by trading 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:
\[ \lambda f(\|w\|) + \sum_{i=1}^{M} (y(x^{(i)}; w) - t_i)^2 \]

• Lasso: use the 1-norm instead 2-norm for f:
\[ \lambda \|w\|_1 + \sum_{i=1}^{M} (y(x^{(i)}; w) - t_i)^2 \]

Norm Balls

q-norm balls for different values of q
(green contour = points equidistant from origin for norm q)
Regularization and Norm Balls

• $L_2$ ball ($f = 2$-norm)
  – Smooth
  – Chance of hitting 0 values is vanishingly small
• $L_1$ ball ($f = 1$-norm)
  – Pointy
  – Chance of hitting all non-0 values vanishingly small

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

Blue = contours of equal error on training set

What $L_1$ Regularization Does

• Trades performance on training data against $L_1$ norm of the weights
• When $\lambda$ is large, many weights will be zero
• Tends to favor sparser solutions
Implementing LASSO

• Several different approaches are possible:
  – Minimize weighted sum of training error & 1-norm on weights
  – Minimize training error w/strict bound on 1-norm of weights

• Both easily be implemented as a convex program
  (can be solved somewhat efficiently by gradient descent)
• Solve incrementally using an algorithm called LARS

Tangent: Working with 1-norm

• Suppose you want to minimize the 1-norm of a vector \( \mathbf{x} \) 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 proportional to $e^{\|x\|}$
- Lasso = assumption of Laplace (double exponential) distribution:

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

Bayesian vs. Non-Bayesian Regularization

- Is there really a difference?
- Bayesian view is arguably more general:
  - Can always find a Bayesian interpretation of anything by exponentiating and calling it a prior
  - Not all all priors can be interpreted as “complexity penalties” in an optimization framework

- Hyperpriors (on other parameters such as $\lambda$)?
  - Priors on priors
  - Actually makes sense if # of parameters is decreasing at each level of the hierarchy
  - Actually works!
More thoughts on Bayesian approaches

• Priors open 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: Not all Bayesian priors reduce to easy optimization problems with closed form solutions

Which is better L₁ or L₂?

• No clear winner
• L₂:
  – Easier to implement
  – Sometimes gives better performance on test data
• L₁:
  – More expensive (no direct solution)
  – Gives more understandable (interpretable) 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 some cases is has been shown that $L_1$ is a reasonable approximation to $L_0$

## Regularization vs. Cross Validation

- Why is regularization better than cross validation as a way to avoid overfitting?
  - Not clear how to use cross validation efficiently when evaluating many potential feature sets
  - Exponentially many possible subsets of a set of candidate features
- But how do we we pick regularization parameter $\lambda$?
- May use cross validation, but observe that this is now a search a one-dimensional search, i.e., find the $\lambda$ at which overfitting starts
Other ways to get a sparse, good feature set

• 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, but forward selection provably finds the “right” features under certain assumptions

Feature Selection vs. Deep Learning?

• Q: Is feature selection still an issue with deep learning?
• A: Yes, but it is more subtle:
  – Early layers of deep networks view as discovering complex features from raw inputs
  – Deep learning is surprisingly resistant to overfitting in some cases
  – Overfitting is still possible
  – Still useful to use cross validation to avoid overly complicated networks
  – Regularization is still used sometimes to produce more robust solutions
Conclusions

• Regularization trades training set performance against solution complexity to avoid overfitting
• Can reduce the need for cross validation, but
  – Regularization parameters still must be chosen, possibly by cross-validation
  – 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