COMP 790-124: Goals for today

Set up our first model.

See a simple optimization algorithm derivation.

Get several algorithms to play with.

All in the context of linear regression.

Steps in applying machine learning techniques to a new problem

Model	Specify a model and generate some synthetic data
Objective	Obtain an objective for your model
Optimization	Derive an optimization algorithm, test on synthetic data
Application	Apply optimization algorithm to the real data

A little bit on notation

A nice convention for writing math:

1. scalars, plain lower case x, y, z
2. vectors, bold lower case $\mathbf{x}, \mathbf{y}, \mathbf{z}$
3. matrices, bold upper case $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$

So, if you had a vector $\mathbf{x}$ then its i^{th} entry would be x_i.

If you had a matrix $\mathbf{X}$, you might denote its i^{th} row as $\mathbf{x}_i$, and element of $\mathbf{X}$ in i^{th} row and j^{th} column would be $x_{i,j}$.

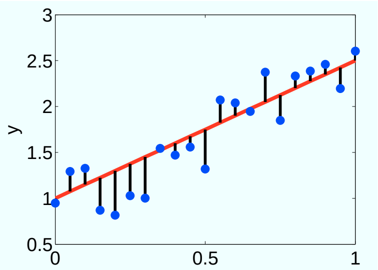
Model: Linear regression

We want to model

$$y = \beta_0 + \sum_{j=1}^{p} x_j \beta_j + \epsilon$$

where ϵ is Gaussian noise with mean 0 and variance σ^2

We assume that y can be explained as a linear combination of p features in x, and this linear combination was corrupted by Gaussian noise.



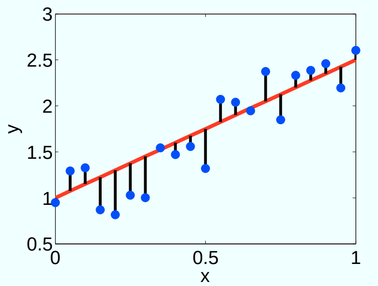
Model: Synthesizing data from the model

In developing and testing our learning procedures we frequently rely on *synthetic* data.

Our model: $y = \beta_0 + \sum_{j=1}^{p} x_j \beta_j + \epsilon$

Simple synthetic data generation:

```
beta0 = 1; beta = [1.5];sigma = sqrt(0.1);
x = [0:0.05:1]';
y = beta0 + x*beta; noise = sigma*randn(length(x),1);
y = y + noise;
```



Objective: Linear regression - likelihood function

We start by writing out a probability distribution

$$p(y|\mathbf{x}, \beta_0, \boldsymbol{\beta}, \sigma) = \frac{1}{\sqrt{(2\pi)\sigma^2}} \exp\left\{-\frac{(y - \beta_0 - \mathbf{x}'\boldsymbol{\beta})^2}{2\sigma^2}\right\}$$

Suppose we gathered n instances of x and y.

We denote the i^{th} instance as $(\mathbf{x}_i, y_i)$, then we can write down a likelihood function

$$L(\beta_0, \beta, \sigma) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \beta_0, \boldsymbol{\beta}, \sigma)$$

A likelihood function enables us to compare parameters in their suitability in explaining data. Always in the context of a model.

Objective: Log likelihood

Log likelihood is simpler to work with than plain likelihood

$$
\begin{aligned}
\log L(\beta_0, \boldsymbol{\beta}, \sigma) &= \sum_i \log p(y_i | \mathbf{x}_i, \beta_0, \boldsymbol{\beta}, \sigma) \\
&= -n/2 \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} \sum_i (y_i - \beta_0 - \mathbf{x}_i' \boldsymbol{\beta})^2
\end{aligned}
$$

Log is a monotone function; maximum of $\log(f(x))$ is achieved at the same point as maximum of $f(x)$.

Optimization: Maximum likelihood estimate

Finding a maximum likelihood estimate amounts to fitting a probabilistic model.

In the case of linear regression there are a couple of ways of doing this. We are going with an easy one: coordinate ascent.

Optimization: Coordinate ascent

The idea is that we update each of our parameters, in turn, so as to maximize likelihood.

We stop when the log likelihood stops changing.

Optimization: Deriving a coordinate ascent algorithm

Compute the partial derivative of log-likelihood with respect to each parameter and equate it to 0, solve for updates.

$$\frac{\partial \log L(\beta_0, \beta_1, \ldots, \beta_p)}{\partial \beta_i} = 0$$

In our case

$$\begin{aligned}
\beta_0 &= \frac{1}{n} \sum_i (y_i - \mathbf{x}_i' \boldsymbol{\beta}) \\
\beta_j &= \frac{1}{\sum_i x_{i,j}^2} \sum_i \underbrace{\left(y_i - \beta_0 - \sum_{k \neq j} x_{i,k} \beta_k\right)}_{y_i^{(-j)}} x_{i,j}
\end{aligned}$$

Optimization: Coordinate ascent with normalized predictors

If we assume that each feature is normalized

$$\sum_i x_{i,k}^2 = 1 \quad \text{and} \quad \sum_i x_{i,k} = 0$$

then we can simplify[1]

$$\begin{aligned} \beta_0 &= \frac{1}{n} \sum_i y_i \\ \beta_j &= \sum_i y_i^{(-j)} x_{i,j} \end{aligned}$$

So we really only have to iterate through updating $\beta_1, \beta_2, \ldots \beta_k, \beta_1, \ldots$ until the likelihood no longer improves.

[1] reminder: $y_i^{(-j)} = y_i - \beta_0 - \sum_{k \neq j} x_{i,k} \beta_k$

Intermezzo - catching bugs early

Newbie mistake: Write the whole code at once and run it without any sanity checks/tests!

Intermezzo - catching bugs early

We have synthetic data in hand:

- Any amount of it (too much might make your code run slowly)
- You have the ground truth parameters used to generate it

Check your optimization algorithm:

- If you are using gradients, check them numerically (compare to finite differences)
- If you are using a different step to optimize: What are the guarantees? Do they hold as you run on the synth data?
- Check updates one at a time:
  - Set all variables/parameters except one to ground truth, update that one variable and see if the objective is improving.
- Run updates on all parameters from some sensible initialization: Do you recover the ground truth?
  - Is the algorithm converging?
  - Are you getting stuck? Is your objective multimodal?
  - How does the objective of the solution you converge to compare to the ground truth's objective?

Quick look at some simple code

Stripped down of various checks so we can see structure.

Questions?

Difficulties

What if we have two copies of the same predictor?

This happens, for example, with single nucleotide polymorphism data (SNP is a variable position in a genome)

If two variable positions are sufficiently close in the genome these two positions might be inherited together. Example:

Patient	Pos 1	Pos 2
1	A/A	G/G
2	A/A	G/G
3	C/C	A/A
4	C/C	A/A
5	A/A	G/G
6	C/C	A/A
7	C/C	A/A

encode as

Patient	Pos 1	Pos 2
1	1	1
2	1	1
3	0	0
4	0	0
5	1	1
6	0	0
7	0	0

Anything we can say using SNP at position 1 as a predictor we can say by using SNP at position 2 as a predictor.

Difficulties

Nothing in the objective function we chose (log likelihood for linear regression) tells us how to resolve these ties.

Things can get even uglier with multicolinear predictors (ex. $x_3 = x_1 + 0.5x_2 - 0.1x_4$) because we can not catch these relationships as easily as we can catch copies of the same predictor.

More difficulties

Having more predictors than data instances ($p > n$) *guarantees* multicolinearity.

It is clear that we have to break ties between all of these solutions that have equal likelihood.

This is a common issue: an ill-posed problem

Objective: Adding Regularization/prior

Standard solution to ill-posedness of problems is to use some sort of regularization.

Sum of squares of parameters - Tikhonov regularization

Change our objective

$$\underbrace{-n/2\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_i (y_i - \beta_0 - \mathbf{x}_i'\boldsymbol{\beta})^2}_{\text{log likelihood}} - \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{\textit{regularization}}$$

Model: Regularization/prior

Feels like a trick: how would we come up with Tikhonov regularization?

Model: Regularization/prior

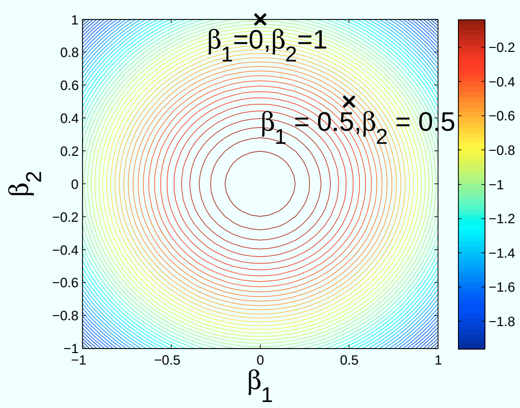
A second look at our model:

$$
\begin{aligned}
p(y|\mathbf{x}, \beta_0, \boldsymbol{\beta}, \sigma) &= \frac{1}{\sqrt{(2\pi)\sigma^2}} \exp\left\{-\frac{(y - \beta_0 - \mathbf{x}'\boldsymbol{\beta})^2}{2\sigma^2}\right\} \\
p(\beta_j) &= \frac{1}{\sqrt{(2\pi)\psi^2}} \exp\left\{-\frac{\beta_j^2}{2\psi^2}\right\} \qquad (j > 0)
\end{aligned}
$$

We have added a prior distribution on feature weights $\boldsymbol{\beta}$. A Gaussian distribution with mean 0 and variance ψ^2

Model: Regularization/prior

What does this prior distribution do? Consider just 2 predictors, like the SNPs we had before. These are the level curves of our prior:



A priori, before we see any data, we are saying that we prefer an even split of predictor weights over just choosing one of them.

Objective: Likelihood $\times$ prior $\propto$ posterior

Recall Bayes' theorem (here applied to $\boldsymbol{\beta}$ and Data):

$$p(\boldsymbol{\beta}|\mathrm{Data}) = \frac{\mathrm{p}(\mathrm{Data}|\boldsymbol{\beta})\mathrm{p}(\boldsymbol{\beta})}{\mathrm{p}(\mathrm{Data})}$$

Product of likelihood function and a prior is *proportional to* ($\propto$) posterior distribution

$$p(\boldsymbol{\beta}|\mathrm{Data}) \propto \underbrace{p(Data|\boldsymbol{\beta})}_{\text{likelihood}} \underbrace{p(\boldsymbol{\beta})}_{\text{prior}}$$

Finding β that maximizes $p(\boldsymbol{\beta}|\mathrm{Data})$ is called MAP (Maximum a posteriori) estimation.

For the purposes of MAP estimation of $p(\mathrm{Data})$ is a constant and we can ignore it. However, it is very important in model selection. We will come back to it.

Objective: Likelihood $\times$ prior $\propto$ posterior

In our case

$$L(\beta_0, \boldsymbol{\beta}, \sigma) = \prod_{i=1}^{n} p(y_i | \mathbf{x}_i, \beta_0, \boldsymbol{\beta}, \sigma)$$

and a prior

$$p(\boldsymbol{\beta}) = \prod_{j} \frac{1}{\sqrt{(2\pi)\psi^2}} \exp\left\{ -\frac{\beta_j^2}{2\psi^2} \right\}$$

yield the posterior distribution over $\boldsymbol{\beta}$

$$\begin{aligned} p(\boldsymbol{\beta}|y) \quad \propto \quad & \prod_{i} \frac{1}{\sqrt{(2\pi)\sigma^2}} \exp\left\{ -\frac{(y_i - \beta_0 - \mathbf{x}_i' \boldsymbol{\beta})^2}{2\sigma^2} \right\} \times \\ & \prod_{j} \frac{1}{\sqrt{(2\pi)\psi^2}} \exp\left\{ -\frac{\beta_j^2}{2\psi^2} \right\} \end{aligned}$$

Objective: Regularized log likelihood and log posterior

Regularized log likelihood after eliminating all terms that do not depend on β

$$-\frac{1}{2\sigma^2}\sum_i (y_i - \beta_0 - \mathbf{x}_i'\boldsymbol{\beta})^2 - \lambda \sum_{j=1}^{p} \beta_j^2$$

Log of posterior $p(\boldsymbol{\beta}|y)$ after eliminating all terms that do not depend on β

$$-\frac{1}{2\sigma^2}\sum_i (y_i - \beta_0 - \mathbf{x}_i'\boldsymbol{\beta})^2 - \frac{1}{2\psi^2} \sum_{j=1}^{p} \beta_j^2$$

So if we set $\lambda = \frac{1}{2\psi^2}$ this the same objective with respect to $\boldsymbol{\beta}$.

Objective: Ridge regression

The cost we introduced is referred to as ridge regression cost and $\sum \beta_j^2$ is called ridge term.

We can state the ridge regression objective as

$$LL_{\mathrm{ridge}}(\beta) = -\frac{1}{2}\sum_i (y_i - \beta_0 - \mathbf{x}_i' \boldsymbol{\beta})^2 - \frac{\alpha}{2}\sum_{j=1}^{p} \beta_j^2$$

You will frequently see it with a sign flip and optimization presented as a minimization. We'll stick with likelihood and maximization.

Optimization: Back to optimization

Take derivatives of the ridge regression objective and set them to zero, this gives us[2]

$$
\begin{aligned}
\beta_0 &= \frac{1}{n} \sum_i (y_i - \mathbf{x}_i' \boldsymbol{\beta}) \\
\beta_j &= \frac{1}{1+\alpha} \sum_i y_i^{(-j)} x_{i,j}
\end{aligned}
$$

Stating the obvious: setting $\alpha = 0$ recovers the regression update.

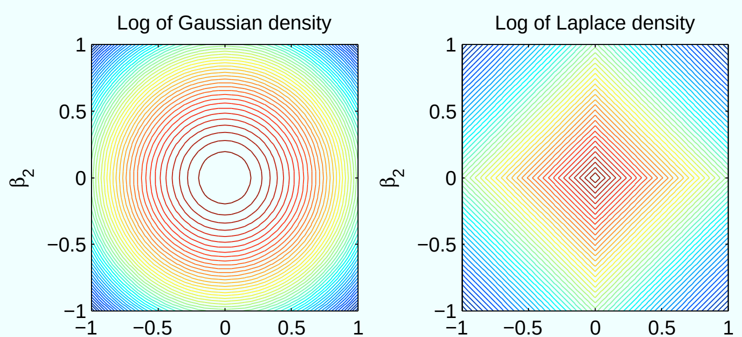
[2] reminder: $y_i^{(-j)} = y_i - \beta_0 - \sum_{k \neq j} x_{i,k} \beta_k$

More priors

Putting a Gaussian prior on β gave us an interesting and useful effect: we can deal with $p > n$ situations and weights get split across similar predictors.

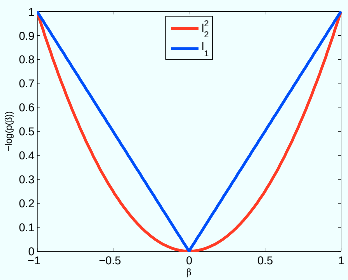
You can now play and put other priors on β, but one choice gets quite a bit of attention.

$$p(\beta_j) \propto \exp\left\{-\frac{1}{s}|\beta_j|\right\}$$



Encouraging sparsity - Penalty view

Ridge regression alleviates some of the problems of $p > n$ but does not drive feature weights to 0.



Red line corresponds to Gaussian prior and blue to Laplace prior.

Objective: Lasso = linear regression + ℓ_1 penalty

Or alternatively, linear regression with Laplace prior on β.

$$LL_{\mathrm{lasso}}(\boldsymbol{\beta}) = -1/2 \sum_i (y_i - \beta_0 - \mathbf{x}_i' \boldsymbol{\beta})^2 - \lambda \sum_{j=1}^{p} |\beta_j|$$

Optimization: Coordinate ascent for lasso

Not as easy as previous ones: $|\cdot|$ does not have a derivative at 0, so we split cases

Suppose optimal $\beta_j > 0$ then we can write a partial derivative of objective down

$$\frac{\partial LL_{\text{lasso}}}{\partial \beta_j} = \sum_i (y - \beta_0 - \mathbf{x}'\boldsymbol{\beta})(x_{i,j}) - \lambda$$

and equating this to 0 gives us[3]

$$\beta_j = \sum_i y_i^{(-j)} x_{i,j} - \lambda$$

as long as this does not invalidate our assumption that $\beta_j > 0$ we can accept this update.

---

[3]reminder: $y_i^{(-j)} = y_i - \beta_0 - \sum_{k \neq j} x_{i,k}\beta_k$

Optimization: Coordinate ascent for lasso

Suppose optimal $\beta_j < 0$ then we can write a partial derivative of objective down

$$\frac{\partial LL_{\text{lasso}}}{\partial \beta_j} = \sum_i (y - \beta_0 - \mathbf{x}'\boldsymbol{\beta})(x_{i,j}) + \lambda$$

and equating this to 0 gives us[4]

$$\beta_j = \sum_i y_i^{(-j)} x_{i,j} + \lambda$$

as long as this does not invalidate our assumption that $\beta_j < 0$ we can accept this update.

[4]reminder: $y_i^{(-j)} = y_i - \beta_0 - \sum_{k \neq j} x_{i,k}\beta_k$

Optimization: Coordinate ascent for lasso

The only times we invalidate our assumptions are the ones where

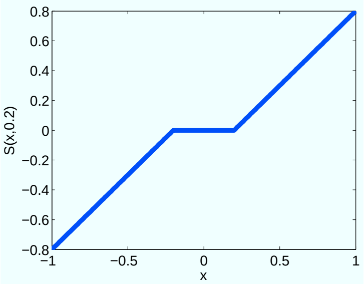
$$\left| \sum_i y_i^{(-j)} x_{i,j} \right| < \lambda$$

What do we do in this case? Set $\beta_j = 0$. Check that is indeed optimal.

Optimization: Shrink and threshold

This reasoning leads to the definition of a shrinkage and threshold operator

$$S(x, \lambda) = \mathrm{sign}\,(x)\max(|x| - \lambda, 0)$$



Optimization: Lasso updates

Now we can write down the lasso updates

$$\beta_0 = \frac{1}{n}\sum_i (y_i - \mathbf{x}_i'\boldsymbol{\beta})$$

$$\beta_j = S\left(\sum_i y_i^{(-j)} x_{i,j}, \lambda\right)$$

where $S(x, \lambda) = \mathrm{sign}(x)\max(|x| - \lambda, 0)$ and $y_i^{(-j)} = y_i - \beta_0 - \sum_{k \neq j} x_{i,k}\beta_k$

Objectives you can optimize now

Linear Regression using update

$$\beta_j = \sum_i y_i^{(-j)} x_{i,j}$$

Ridge Regression using update

$$\beta_j = \frac{1}{1+\alpha} \sum_i y_i^{(-j)} x_{i,j}$$

Lasso using update

$$\beta_j = S\left(\sum_i y_i^{(-j)} x_{i,j}, \lambda\right)$$

I have not told you how to pick λ,α etc. But you can think about cross validation and how you would use it.

Next time we will expand this to elastic net (Ridge + Lasso) talk about how to cross-validate and see some applications.

We covered

- Several models: Linear Regression, Ridge Regression, Lasso
- Linked these models to their objectives
- Presented optimization algorithms for these objectives
- For more on what we covered today see [2], `http://www.jstatsoft.org/v33/i01/paper`
- Also look at Least Angle Regression (LARS) [1], `http://projecteuclid.org/euclid.aos/1083178935`

- Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani.
  Least angle regression.
  Annals of Statistics, 32:407–499, 2004.
- Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani.
  Regularization paths for generalized linear models via coordinate descent.
  Journal of Statistical Software, 33(1):1–22, 2 2010.