Optimization

Nicole Beckage

Quality control

Brakes' Bearing Works manufactures bearing shafts whose diameters are normally distributed with parameters \( \mu = 1, \sigma = 2 \). The buyer's specifications require these diameters to be \( 1.000 \pm 0.002 \) cm. What fraction of the manufacturer's shafts are likely to be rejected? If the manufacturer improves her quality control, she can reduce the value of \( \sigma \). What value of \( \sigma \) will ensure that no more than 1 percent of her shafts are likely to be rejected?

Appeal to CDF

Use the statistical distribution

What does the standard deviation need to be?

- So we know what the standard deviation needs to be to get the error rate down to 1/100 shafts rejected.
  - Std <= 0.00116
- What happens if the engineers come back with a cost estimate of how much it will take to decrease variance of the shafts?
  - Maybe 1/100 rejection rate is not the most profitable or optima,
Optimization with constraints

- We want to find a way of making the standard deviation small
- Without costing the company too much
- We need to find the optimal tradeoff between cost and efficiency
  - Cost here being decreasing the std
  - Efficiency being how many shafts are thrown away

What is Optimization?

- Find the variables that minimize (or maximize) a function subject to constraints.
- Minimize = -1 * maximize

Example:
- Stock market. Minimize variance of return subject to getting at least $50.
- Weather: Maximize the probability of seeing an eclipse considering time and the cost of gas.

Why do we care

- Optimization is what most machine learning problems are
- Linear regression for example is just solving $\min_{\beta} \sum (y - \beta x)^2$
- SVM classifier is also a type of minimization problem
- As is neural networks, k means, graphical models etc.
- MLE is a type of optimization problem too.

- Nearly every ML problem can be framed as an optimization problem.

Maximum likelihood estimation:

$\max_{\theta} \sum \log p_{\theta}(x_i)$

Collaborative filtering:

$\min_{w} \sum \log (1 + \exp(w^T x_i - w^T z_j))$

$k$-means:

$\min_{\mu_1, \ldots, \mu_k} \sum_{i=1}^{k} \sum_{j \in S_i} |x_i - \mu_j|^2$

Optimization is hard.

- We need to find the global minimum.
- It’s impossible to know for certain we’ve found the global minimum
- Unless we check everywhere.
- (goes back to the issue of inference)
- We can’t check everywhere in the space because that’s computationally unrealistic.
- It’s even harder in discrete spaces...
So what can we do?

- We’ve looked at optimization in a few settings...
  - Linear regression: apply a bunch of math and find the best solution
  - Regularized regression: explore the space by trying out a few different values
  - Neural networks: make decisions based on a subset of the data and repeat until parameter changes is small.

Analytic solutions

- Wear the hat of a statistician rather than a data scientist
  - If we’re working in the domain where distributional assumptions are true (e.g. normal distribution, finite variance, etc.)
  - We’ll write down a function to capture how different our model prediction is from the true outcome
  - And solve for the MLE of the error function.

- Linear regression all over again.

Optimization algorithms

- What happens when the math is too complex?
  - We have some stupid algorithms we can try that actually do pretty well.

  Most common one is gradient descent.
  - Find the lowest point by following gradients to minimum values

Gradient Descent: The delta learning rule

- \[ \Delta w_j = \varepsilon x_j(y_j - y) \]
  - We want to find the parameters that minimize the difference between our predicted and actual output
  - We also want to adjust our parameters in a direction that will get us closer to the true output
  - First derivative tells us slope to min
  - We want to adjust our parameters proportional to the gradient.

Deriving the delta rule

Define error as RSS

\[ E = \sum_{i=1}^{n}(y^{(i)} - y^{(i)})^2 \]

Differentiate error with respect to weights (chain rule)

\[ \frac{\partial E}{\partial w_j} = \sum_{i=1}^{n} \frac{\partial E^{(i)}}{\partial w_j} \frac{\partial y^{(i)}}{\partial w_j} \]

\[ = -2 \sum_{i=1}^{n} x_j^{(i)} (y^{(i)} - y^{(i)}) \]

Change weights proportional to the error derivatives summed over all training cases

\[ \Delta w_j = -\varepsilon \frac{\partial E}{\partial w_j} \]

\[ = \sum_{i=1}^{n} e_i x_j^{(i)} (y^{(i)} - y^{(i)}) \]
Using the delta rule

- The delta rule gives us a means to design a class of algorithms.
- Algorithm for Neural network is called back propagation, in matlab it’s called fminsearch, etc.
- We move along the gradient until we’ve found a minimum.
- We do this multiple times with multiple initial conditions to deal with local minima.

Gradient Decent

- Adjust weights to move in direction of the minimum.

\[
\Delta w_{ji} = \epsilon x_i (y - \hat{y}) f'(h_j)
\]

The role of \( \epsilon \)
- How big of steps do we make along the gradient

Small \( \epsilon \)

Big \( \epsilon \)

Online vs batch

Batch learning: steepest descent on the error surface
- Travels perpendicular to the contour lines

Online: gradient of descent based on error of single example
- Zig-zags in direction of steepest descent.

Why learning can be slow

- Direction of steepest decent is perpendicular to the minimum.
- Learning rate too small
  - Takes forever to move toward minimum
- Learning rate too large
  - High probability of moving away from the minimum.
Do we have guarantees?

- In a lot of cases I have said that any solution we find to the delta rule is the best solution we can find.
- Implies that we've found a point in the error hyperplane where the derivatives are zero.
- How can we know that a solution is the best solution?
- In general, we can’t
- In specific (and surprisingly common) conditions we can.

Convexity

- We have a space such that if a straight line is drawn between any two points in that space, every point on the line is also in the space.

Convex optimization problems

- An optimization problem is convex if its objective (loss, error) function is a convex function.
- The constraints are convex and
- The equality constraints are affine.

What does convexity buy us?

- Tons!
- You want to find a way of making your error function convex.
- Lots of math but the takeaway is
  - In convex functions we know that any local minima is the global minima
  - This implies if we find one point where the gradient is zero, we have found the best parameters for our model.