CPSC 340: Machine Learning and Data Mining

Gradient Descent

BONUS SLIDES

Beyond Gradient Descent

- There are many variations on gradient descent.
  - Methods employing a “line search” to choose the step-size.
  - “Conjugate” gradient and “accelerated” gradient methods.
  - Newton’s method (which uses second derivatives).
  - Quasi-Newton and Hessian-free Newton methods.
  - Stochastic gradient (later in course).

- This course focuses on gradient descent and stochastic gradient:
  - They’re simple and give reasonable solutions to most ML problems.
  - But the above can be faster for some applications.

Why use the negative gradient direction?

- For a twice-differentiable 'f', multivariable Taylor expansion gives:

$$f(w^{t+1}) = f(w^t) + \nabla f(w^t)^T (w^{t+1} - w^t) + \frac{1}{2}(w^{t+1} - w^t)\underbrace{\nabla^2 f(v)}_{\text{for some 'v' between } w^{t+1} \text{ and } w^t.}(w^{t+1} - w^t)$$

- If gradient can't change arbitrarily quickly, Hessian is bounded and:

$$f(w^{t+1}) = f(w^t) + \nabla f(w^t)^T (w^{t+1} - w^t) + \underbrace{O(\|w^{t+1} - w^t\|^2)}_{\text{becomes negigible as } w^{t+1} \text{ gets close to } w^t}$$

  - But which choice of w^{t+1} decreases 'f' the most?
    - As $||w^{t+1}-w^t||$ gets close to zero, the value of w^{t+1} minimizing $f(w^{t+1})$ in this formula converges to $(w^{t+1} - w^t) = - \alpha^t \nabla f(w^t)$ for some scalar α^t.
    - So if we're moving a small amount, the optimal w^{t+1} is: $w^{t+1} = w^t - \alpha_t \nabla f(w^t)$ for some scalar α_t.

Normalized Steps

Question from class: "can we use $w^{t+1} = w^t - \frac{1}{\|\nabla f(w^t)\|} \nabla f(w^t)$"

This will work for a while, but notice that

$$\|w^{t+1} - w^t\| = \left\| \frac{1}{\|\nabla f(w^t)\|} \nabla f(w^t) \right\|$$

$$= \frac{1}{\|\nabla f(w^t)\|} \|\nabla f(w^t)\|$$

$$= 1$$

So the algorithm <u>never converges</u>

Log-Sum-Exp for Brittle Regression

- To use log-sum-exp for brittle regression:

$$\|Xw - y\|_\infty = \max_i \{|w^T x_i - y_i|\}$$

$$= \max_i \{\max\{w^T x_i - y_i, y_i - w^T x_i\}\} \quad \text{since } |z| = \max\{z, -z\}$$

$$= \log\left(\sum_{i=1}^n \exp(w^T x_i - y_i) + \sum_{i=1}^n \exp(y_i - w^T x_i)\right)$$

using log-sum-exp to approximate "max" over $2n$ terms.

Log-Sum-Exp Numerical Trick

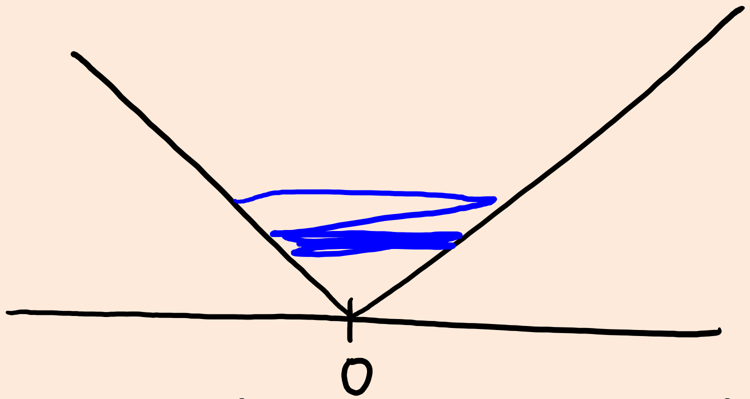
- Numerical problem with log-sum-exp is that $\exp(z_i)$ might overflow.
  - For example, $\exp(100)$ has more than 40 digits.
- Implementation 'trick': Let $\beta = \max_i \{z_i\}$

$$
\begin{aligned}
\log\left(\sum_i \exp(z_i)\right) &= \log\left(\sum_i \exp(z_i - \beta + \beta)\right) \\
&= \log\left(\sum_i \exp(z_i - \beta)\exp(\beta)\right) \\
&= \log\left(\exp(\beta)\sum_i \exp(z_i - \beta)\right) \\
&= \log(\exp(\beta)) + \log\left(\sum_i \exp(z_i - \beta)\right) \\
&= \beta + \log\left(\sum_i \exp(z_i - \beta)\right)
\end{aligned}
$$

≤ 1 so no overflow

Gradient Descent for Non-Smooth?

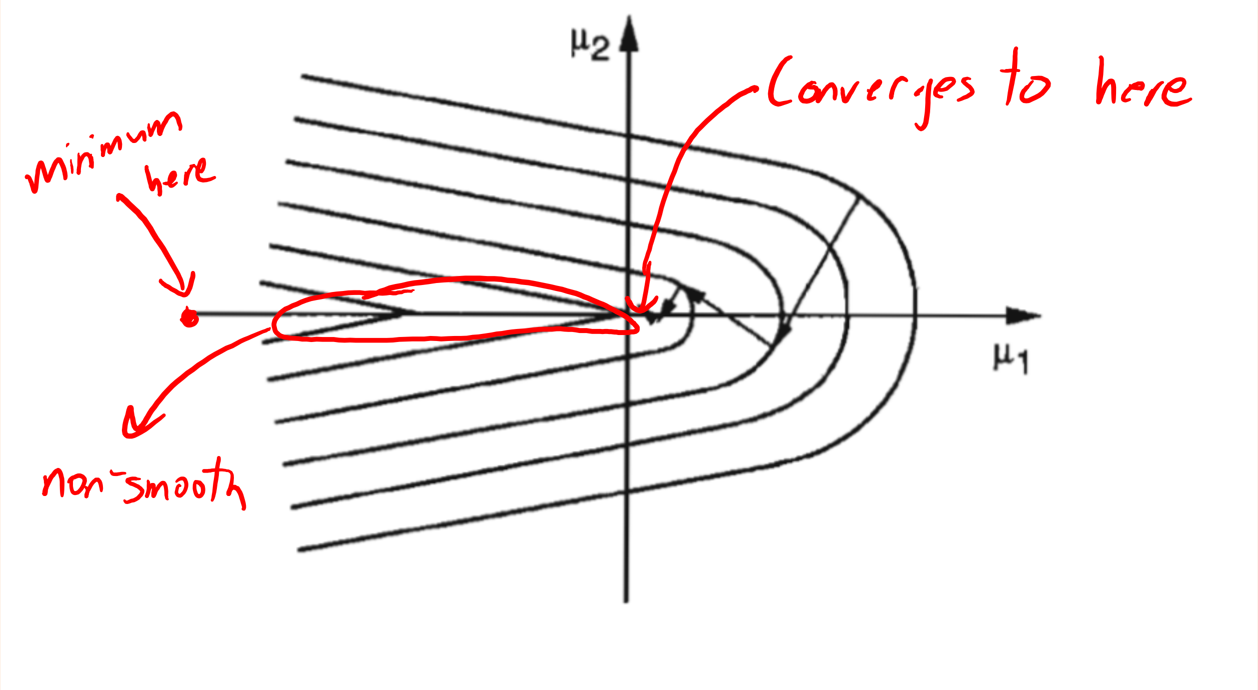
- "You are unlikely to land on a non-smooth point, so gradient descent should work for non-smooth problems?"
  - Consider just trying to minimize the absolute value function:



  - Norm(gradient) is constant when not at 0, so unless you are lucky enough to hit exactly 0, you will just bounce back and forth forever.
  - We didn't have this problem for smooth functions, since the gradient gets smaller as you approach a minimizer.
  - You could fix this problem by making the step-size slowly go to zero, but you need to do this carefully to make it work, and the algorithm gets much slower.

Gradient Descent for Non-Smooth?

- Counter-example from Bertsekas' "Nonlinear Programming" where gradient descent for a non-smooth convex problem does not converge to a minimum.

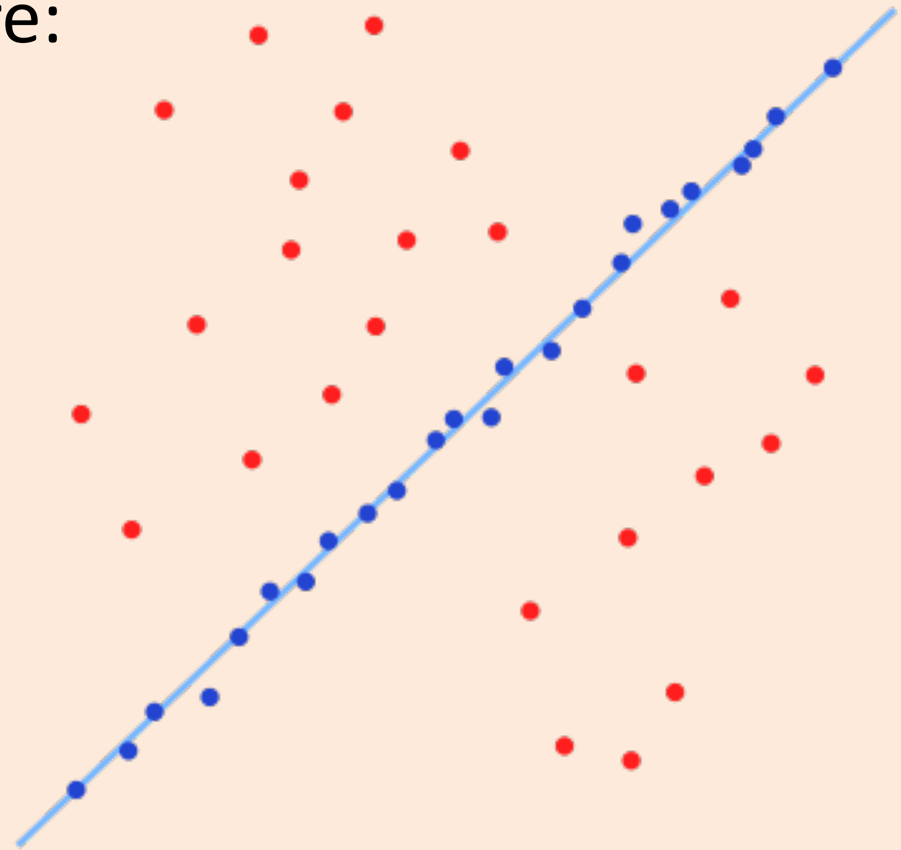


(even though function is convex)

Figure 6.3.8. Contours and steepest ascent path for the function of Exercise 6.3.8.

Random Sample Consensus (RANSAC)

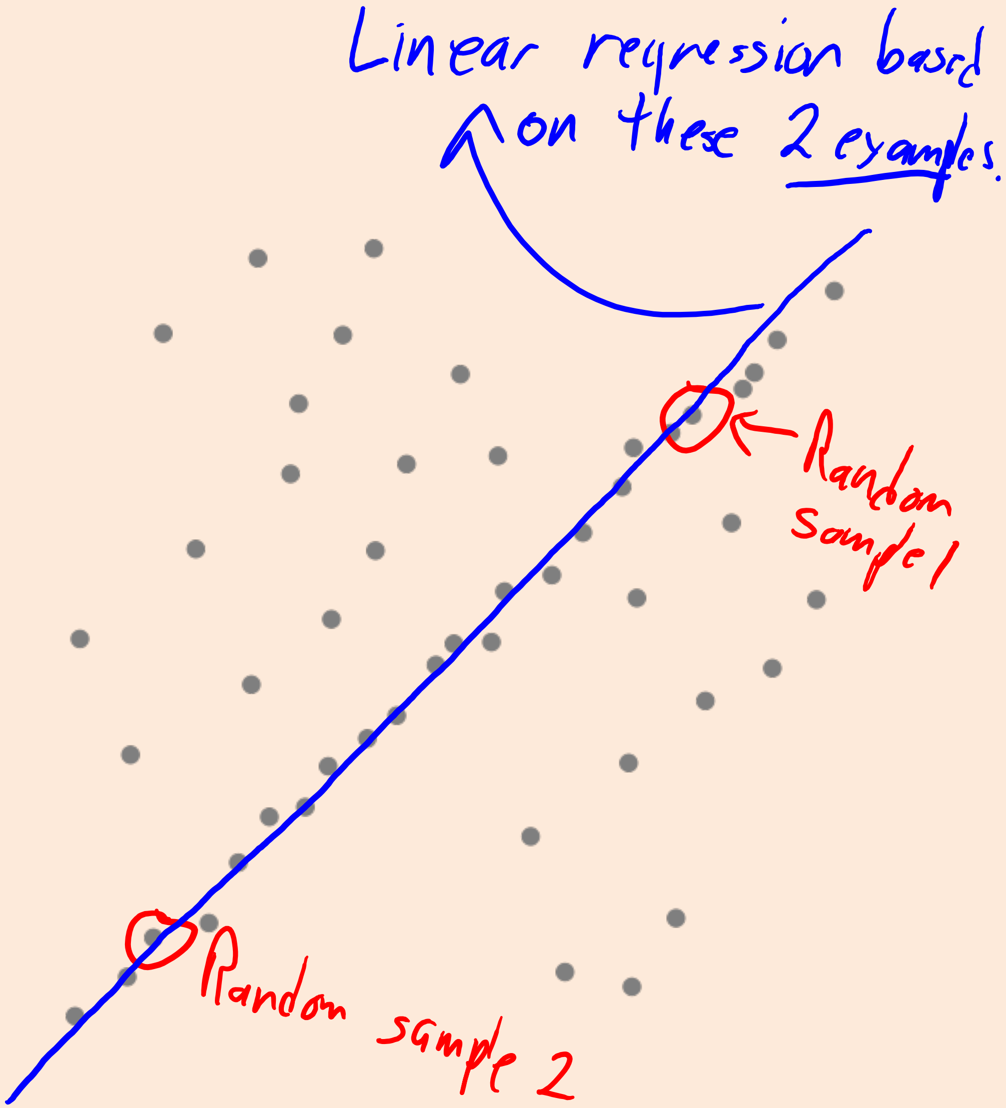
- In computer vision, a widely-used generic framework for robust fitting is random sample consensus (RANSAC).
- This is designed for the scenario where:
  - You have a large number of outliers.
  - Majority of points are "inliers":
    it's really easy to get low error on them.



https://en.wikipedia.org/wiki/Random_sample_consensus

Random Sample Consensus (RANSAC)

- RANSAC:
  - Sample a small number of training examples.
    - Minimum number needed to fit the model.
    - For linear regression with 1 feature, just 2 examples.
  - Fit the model based on the samples.
    - Fit a line to these 2 points.
    - With 'd' features, you'll need 'd' examples.
  - Test how many points are fit well based on the model.
  - Repeat until we find a model that fits at least the expected number of "inliers".
- You might then re-fit based on the estimated "inliers".



https://en.wikipedia.org/wiki/Random_sample_consensus