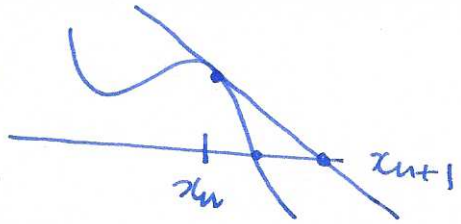
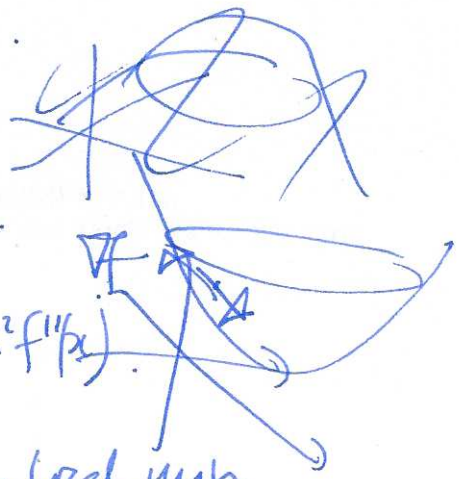
XGBoost

- gradient descent: 1-d version Newton's method: looking for zeros.



high dimension version: $x_{n+1} = x_n - \nabla f$.



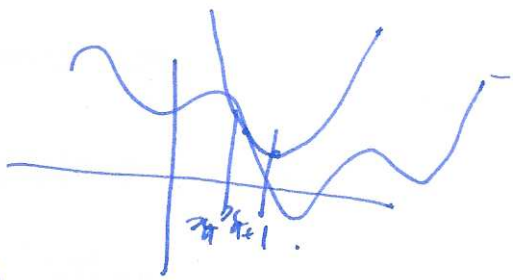
- linear approx: $f(x+t) \simeq f(x) + tf'(x)$.
- quadratic approx: $f(x+t) \simeq f(x) + tf'(x) + \frac{1}{2}t^2 f''(x)$.

note: if $f''(x) > 0$ then quadratic expression has a local min and you can find it!

$$q(x) = f(x) + tf'(x) + \frac{1}{2}t^2 f''(x).$$

$$q'(x) = f'(x) + tf''(x) = 0 \leftarrow \text{solve for critical pt}$$

$$t = -\frac{f'(x)}{f''(x)}.$$



so set $x_{n+1} = x_n + t = x_n - \frac{f'(x_n)}{f''(x_n)}$.

note if original function actually quadratic this gives the correct answer in 1 step!

high dimensional version: $f'(x) \leftrightarrow \nabla f$ - vector of partial derivatives $\left[\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right]^T$.

$f''(x) = \nabla^2 f(x) = Hf(x) =$ nxn matrix of 2nd partial derivatives $\left[\frac{\partial^2 f_i}{\partial x_i \partial x_j}\right]$.

$$x_{n+1} = x_n - \left[\frac{\partial f}{\partial x_i \partial x_j}\right]^{-1} \left[\frac{\partial f}{\partial x_i}\right] = (H_f(a))^{-1} \cdot \nabla f.$$

problem · need to compute inverse of matrix, Gaussian elimination methods are $O(n^3)$.

· if H very small det, H^{-1} may be very large, i.e. local min may be far away, so common modification is to fix $\epsilon > 0$ and $0 < \epsilon < 1$

use: $x_{n+1} = x_n - \epsilon H_f^{-1}(x_n) \nabla f(x_n)$.

Fact: if f is strongly convex and H is Lipschitz and x_0 is close enough to x_{min}, get quadratic convergence.

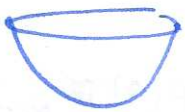
note instead of computing $H_f^{-1}(x_n)$ can solve $[H_f(x_n)]\, h = -f'(x_n)$ ($n \times n$ matrix, n-vector)

(then $h = -H_f^{-1}(x_n) \nabla f$ if H_f^{-1} invertible).

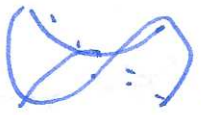
(this)

advantages: still $O(n^3)$, will still get (family) of solutions if $\det(H) = 0$

can use special methods assuming H is positive definite - if special methods don't work and H not +ve definite have problems anyway...



f quad +ve def



f quad saddle - critical point not min!

· problem: if $\det(H)$ close to zero then method is probably unstable.
— usual convergence problems for Newton's method / Gradient descent.

initial data: $X = \{(x_i, y_i)\}_{i=1}^{n}$ $x_i \in \mathbb{R}^d$

- loss function $L(y, F(x))$ (e.g. mean sum of squares, mean sum of the difference)

predicting quantitative data

e.g. $L(\theta) = \frac{1}{n}\sum_i (y_i - \hat{f}(x_i))^2$

- regularization function / shrinkage function / pruning function $\Omega(\theta)$

Logistic loss

$L(\theta) = \sum_i \left(y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i}) \right)$

↑ predicting probabilistic qualitative data.

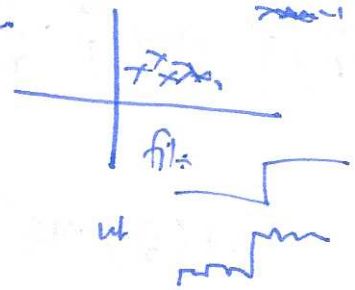
- actually minimize

$obj(\theta) = L(\theta) + \Omega(\theta)$

↑ error term

↑ model complexity term. ((e.g. # of ...))

Example

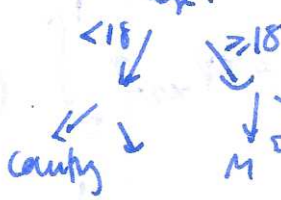


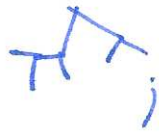
- recall decision trees.

e.g. use. age, sex, family to predict height.

The 24, M, Japan.

7, M,



many trees: (eg. random forest.  ;  ; ...).

take average...

$\Omega(\theta) = \sum_T f_t(x)$ where $f_t(x) =$

$\Omega(\theta)$ could be · # of trees

· # of trees + # leaves

— need to choose trees.
— need to combine trees.

problem: each tree has a certain number of parameters (where to choose each branch, leaf value)

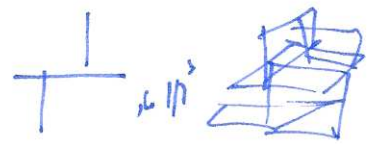
....

- usually can't do just solve to find optimal parameters
  - so need to use gradient descent / Newton Raphson / Hessian method...

simplification: just add one tree at a time!

- single tree t is: $f_t(x) = w_{q(x)}$, $w \in \mathbb{R}^T$

  $q : \mathbb{R}^d \to \{1, 2, \ldots, T\}$

- w vector of values on each leaf 
- q function sends data points $\mathbb{R}^d \to$ leaves, e.g. in $\mathbb{R}^2$ 

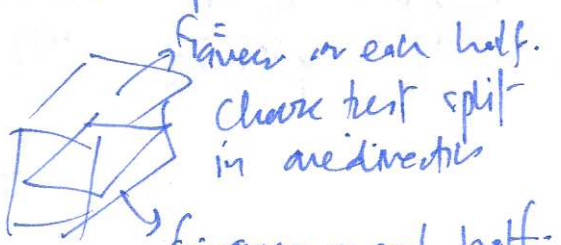
$$\omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$$

γ, λ tuning parameters we need to pick, 0.3?

T = # leaves.

So as you add leaves, if improvement $< \gamma$ then better off not adding that leaf.

- build first trees:



f_0 = constant function = average

$L(f_1) < L(f_0)$.

↑ but only add if at least γ better!

- add second tree: we have the function from the first tree $f_1 : \mathbb{R}^d \to \mathbb{R}$.

  replace data by error in prediction / residuals: $\{(x_i, y_i)\}_{i=1}^n$

  ↓

  $\{(x_i, y_i - f_1(y_i))\}_{i=1}^n$.

now use $f_1 + \alpha f_2$.

↑ learning parameter (0.3?)

keep going....