Quiz Review

CMPUT 467: Machine Learning II

Ch 2: Probability Basics

- Expectations and variance
- Independence and conditional independence
- Joint probabilities, marginal and conditional probabilities
- Mixture distributions

Some questions (1)

- Assume $\mathbf{X}$ is a random vector of dimension d, with covariance $\mathbf{\Sigma}$
- **Question**: Does this mean $\mathbf{X}$ is a multivariate Gaussian? Why or why not?

Some questions (2)

- Assume $\mathbf{X}$ is a random vector of dimension d, with covariance $\mathbf{\Sigma}$
- **Question**: Does this mean $\mathbf{X}$ is a multivariate Gaussian? Why or why not?
- **Answer**: No, covariance is defined for any of the distributions we talk about. The variable $\mathbf{X}$ can even consist of both continuous and discrete random variables

Some questions (3)

- Assume $\mathbf{X}$ is a random vector of dimension d, with covariance $\mathbf{\Sigma}$
- **Follow-up question**: If X_1 is continuous and X_2 is discrete, then what is the formula for Cov(X_1, X_2) ?
- Recall: $\text{Cov}(X_1, X_2) = \mathbb{E}[(X_1 - \mathbb{E}[X_1])(X_2 - \mathbb{E}[X_2])]$

Some questions (4)

- Assume $\mathbf{X}$ is a random vector of dimension d, with covariance $\mathbf{\Sigma}$
- **Follow-up**: If X_1 is continuous and X_2 is discrete, then what is the formula for $\text{Cov}(X_1, X_2)$?
- **Answer**: Let μ_1 and μ_2 be the means for X_1 and X_2 respectively

$$\begin{aligned}\text{Cov}(X_1, X_2) &= \mathbb{E}[(X_1 - \mathbb{E}[X_1])(X_2 - \mathbb{E}[X_2])] \\ &= \int_{\mathcal{X}_1} \sum_{x_2 \in \mathcal{X}_2} p(x_1, x_2)(x_1 - \mu_1)(x_2 - \mu_2) dx_1 \\ &= \int_{\mathcal{X}_1} p(x_1) \sum_{x_2 \in \mathcal{X}_2} p(x_2 \,|\, x_1)(x_1 - \mu_1)(x_2 - \mu_2) dx_1\end{aligned}$$

-

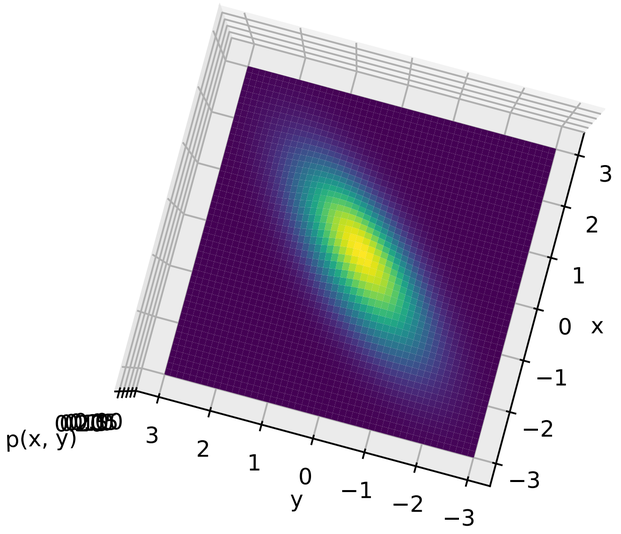
Some questions (5)

- Assume $\mathbf{X}$ is a random vector of dimension d, with covariance $\mathbf{\Sigma}$
- Now assume $\mathbf{X}$ is a multivariate Gaussian
- **Question**: If the first eigenvalue in $\mathbf{\Sigma}$ is very big (1000) and the second is very small (0.1), then what does this tell us about the shape of the Gaussian?

Some questions (5)

- Assume $\mathbf{X}$ is a random vector of dimension d, with covariance $\mathbf{\Sigma}$
- Now assume $\mathbf{X}$ is a multivariate Gaussian
- **Question**: If the first eigenvalue in $\mathbf{\Sigma}$ is very big (1000) and the second is very small (0.1), then what does this tell us about the shape of the Gaussian?
- **Answer**: The distribution is wide in one orientation and narrow in another

Example of eigenvalues



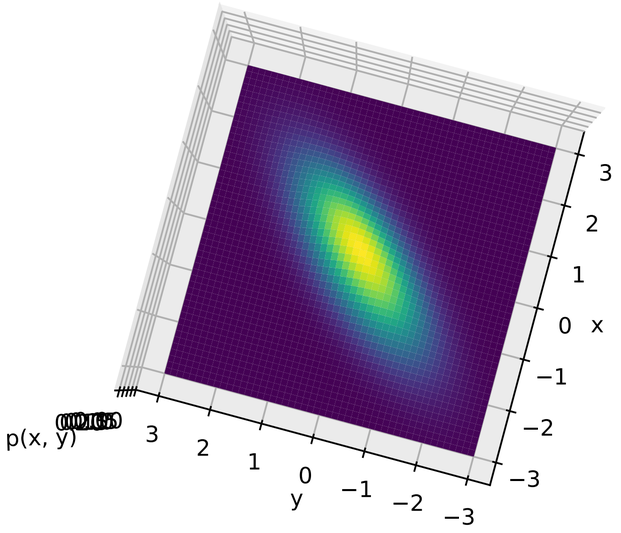
$$\mathbf{\Sigma} = \begin{bmatrix} 1.0 & 0.75 \\ 0.75 & 1.0 \end{bmatrix}$$

This $\mathbf{\Sigma}$ has singular values: $\sigma_1 = 1.75, \sigma_2 = 0.25$

These are also the eigenvalues for $\mathbf{\Sigma}$!

This is not true in general. Why is is true for $\mathbf{\Sigma}$?

Example of eigenvalues



$$\boldsymbol{\Sigma} = \begin{bmatrix} 1.0 & 0.75 \\ 0.75 & 1.0 \end{bmatrix}$$

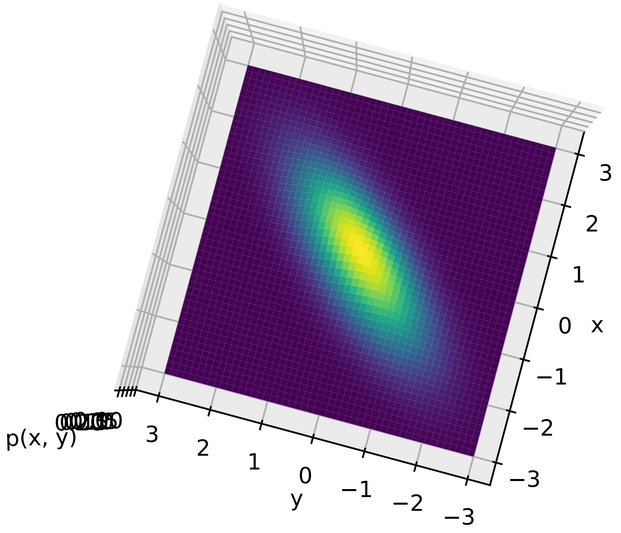
This $\boldsymbol{\Sigma}$ has singular values: $\sigma_1 = 1.75, \sigma_2 = 0.25$
These are also the eigenvalues for $\boldsymbol{\Sigma}$!

For a square, symmetric matrix, the eigenvalue decomposition is

$\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ for orthonormal $\mathbf{U}$, diagonal $\boldsymbol{\Lambda}$

We also know $\boldsymbol{\Sigma}$ is positive definite. What does this tell us about $\boldsymbol{\Lambda}$?

Example of eigenvalues



$$\mathbf{\Sigma} = \begin{bmatrix} 1.0 & 0.75 \\ 0.75 & 1.0 \end{bmatrix}$$

This $\mathbf{\Sigma}$ has singular values: $\sigma_1 = 1.75, \sigma_2 = 0.25$
These are also the eigenvalues for $\mathbf{\Sigma}$!

For a square, symmetric matrix, the eigenvalue decomposition is

$\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ for orthonormal $\mathbf{U}$, diagonal $\mathbf{\Lambda}$

$\mathbf{\Sigma}$ is positive definite, so $\mathbf{\Lambda}$ is a diagonal matrix with positive terms on the diagonal
Therefore, $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ is also a valid SVD

Mixture distributions

- Mixture distributions allow us to get more complex distributions by taking a convex combination of simpler distributions
- Gaussian mixture model (GMM) takes a mixture of m Gaussians, can get back a distribution with up to m modes
- **Question**: can we get even just a bit more complexity by mixing GMMs? What if we take one GMM $p(x) = \sum_{k=1}^{m} w_m p_k(x)$ and another $q(x) = \sum_{k=1}^{m} w_m q_k(x)$ and set the mixture to $\tilde{p}(x) = 0.5p(x) + 0.5q(x)$?

Mixture distributions

- Mixture distributions allow us to get more complex distributions by taking a convex combination of simpler distributions
- **Question**: can we get even just a bit more complexity by mixing GMMs?

  What if we take one GMM $p(x) = \sum_{k=1}^{m} w_m p_k(x)$ and another

  $q(x) = \sum_{k=1}^{m} a_m q_k(x)$ and set the mixture to $\tilde{p}(x) = 0.5p(x) + 0.5q(x)$?
- **Ans**: this is equivalent to taking a GMM with 2m modes

Ch 3: Revisiting Linear Regression

- Linear regression objective and closed-form matrix solution (OLS)
  - note you don’t need to remember formulas
- Understanding why small singular values can indicate we get overfitting
- The utility of l2 regularization for avoiding issues with small singular values
- The bias-variance trade-off, and relationship to the covariance matrix and the singular values of the data matrix

Linear Regression Objectives

- LR objective $\frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2$
- Ridge Regression objective $\frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$
- **Question**: How do we get the LR objective from the RR objective?

Linear Regression Objectives

- LR objective $\frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2$
- Ridge Regression objective $\frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$
- **Question**: How do we get the LR objective from the RR objective?
- **Answer**: Set $\lambda = 0$ (regularization weight is zero, so no regularizer)

Linear Regression Solution

- The closed form solution satisfies $\mathbf{A}\mathbf{w} = \mathbf{b}$ for $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$ and $\mathbf{b} = \mathbf{X}^\top\mathbf{y}$
- **Question**: Our goal is to minimize $\frac{1}{2}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2$. Why can't we just use $\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$? This would be great because then we would have $\mathbf{X}\mathbf{w} = \mathbf{y}$

Linear Regression Solution

- The closed form solution satisfies $\mathbf{A}\mathbf{w} = \mathbf{b}$ for $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$ and $\mathbf{b} = \mathbf{X}^\top\mathbf{y}$
- **Question**: Our goal is to minimize $\frac{1}{2}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2$. Why can't we just use $\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$? Then we would have $\mathbf{X}\mathbf{w} = \mathbf{y}$
- **Answer**: $\mathbf{X}$ is typically not a square matrix and so cannot be inverted (inverse only exists for square matrices)
- Instead, use the pseudo-inverse $\mathbf{X}^\dagger \in \mathbb{R}^{d \times n}$

Pseudoinverse

- For thin SVD $\mathbf{X} = \mathbf{U}_d \mathbf{\Sigma}_d \mathbf{V}^\top$ and full rank $\mathbf{X}$, we have $\mathbf{X}^\dagger = \mathbf{V}\mathbf{\Sigma}_d^{-1}\mathbf{U}_d$
  - When $\mathbf{X}$ is full rank, we have $\mathbf{X}^\dagger = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$
  - Even if not full rank, $\mathbf{X}^\dagger$ is defined $\mathbf{X}^\dagger = \mathbf{V}\mathbf{\Sigma}_d^\dagger\mathbf{U}_d$ where the pseudo-inverse of the singular value matrix is the inverse of non-zero values and 0 else
- The pseudo-inverse $\mathbf{X}^\dagger \in \mathbb{R}^{d \times n}$ is the closest we get to an inverse and $\mathbf{w} = \mathbf{X}^\dagger\mathbf{y}$ (here $\mathbf{X}^\dagger\mathbf{X} = \mathbf{I} \in \mathbb{R}^{d \times d}$ if $\mathbf{X}$ full rank, but $\mathbf{X}\mathbf{X}^\dagger \neq \mathbf{I} \in \mathbb{R}^{n \times n}$)
- Notice $\mathbf{X}\mathbf{w} = \mathbf{X}\mathbf{X}^\dagger\mathbf{y} \neq \mathbf{y}$, but in some sense closest approximation

Linear Regression Solution and Overfitting

- The closed-form solution satisfies $\mathbf{A}\mathbf{w} = \mathbf{b}$ for $\mathbf{A} = \mathbf{X}^\top\mathbf{X}$ and $\mathbf{b} = \mathbf{X}^\top\mathbf{y}$
- If $\mathbf{A}$ is low-rank ($\mathbf{X}$ has a zero singular values), then there are infinitely many solutions for $\mathbf{w}$
  - This linear system is under-constrained
- More likely, $\mathbf{A}$ is nearly low-rank; equivalently $\mathbf{X}$ is nearly low-rank
- Typical reason: insufficient data
- In d dimensions, the observed data **looks** like it lies in a lower-dimensional space, because it takes many points to start covering the actual region spanned by the data

LR and Overfitting

- We know that $\mathbf{X}$ can have small singular values if
  - input features are highly correlated (or linearly dependent)
  - OR we have insufficient data
- **Question**: If the true y is only a function of the first two features of $\mathbf{x}$, then does that imply that $\mathbf{X}$ will be low rank?

LR and Overfitting

- We know that $\mathbf{X}$ can have small singular values if
  - input features are highly correlated (or linearly dependent)
  - OR we have insufficient data
- **Question**: If the true y is only a function of the first three features of $\mathbf{x}$, then does that imply that $\mathbf{X}$ will be low rank?
- **Answer**: likely not. They are different random variables. This functional relationship is about how the RVs $\mathbf{x}$ and y are related. It does not necessarily imply anything about the relationships between RVs within $\mathbf{x}$

  Can you think of an example where this might happen?

LR and Overfitting

- If the true y is only a function of the first three features of $\mathbf{x}$, then does that imply that $\mathbf{X}$ will be low rank?
- **Answer**: likely not. They are different random variables. The functional relationship is about how the RVs $\mathbf{x}$ and y are related. It does not necessarily imply anything about the relationships between RVs within $\mathbf{x}$
- **Exception**: y might only be a function of the first three features because the rest are all perfectly redundant. Then both y is only related to the first three features AND $\mathbf{X}$ is low rank. But there is no reason to believe this is the reason for the relationship, without further info

The LR solution, with and without regularization

- $\mathbf{w}_{\text{MLE}} = \sum_{j=1}^{\text{rank}(X)} \frac{\mathbf{u}_j^\top \mathbf{y}}{\sigma_j} \mathbf{v_j}$ for $\mathbf{w}_{\text{MLE}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$
- $\mathbf{w}_{\text{MAP}} = \sum_{j=1}^{\text{rank}(X)} \frac{\sigma_j \mathbf{u}_j^\top \mathbf{y}}{\sigma_j^2 + \lambda} \mathbf{v_j}$ for $\mathbf{w}_{\text{MAP}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$
- If λ reasonably big (say 10^-3), then we avoid dividing by a very small singular value
- **Question**: Why do we subscript these with MLE and MAP?

Bias and variance

- $\mathbf{w}_{\text{MLE}}$ is unbiased and potentially high-variance, $\sigma^2 \mathbb{E}\left[\sum_{j=1}^{d} \sigma_j^{-2}\right]$
- $\mathbf{w}_{\text{MAP}}$ is biased and lower variance, $\sigma^2 \mathbb{E}\left[\sum_{j=1}^{d} \frac{\sigma_j^2}{(\sigma_j^2 + \lambda)^2}\right]$
- **Question**: when do we expect $\mathbf{w}_{\text{MAP}}$ to be better than $\mathbf{w}_{\text{MLE}}$?

Bias and variance

- $\mathbf{w}_{\text{MLE}}$ is unbiased and potentially high-variance, $\sigma^2 \mathbb{E}\left[\sum_{j=1}^{d} \sigma_j^{-1}\right]$
- $\mathbf{w}_{\text{MAP}}$ is biased and lower variance, $\sigma^2 \mathbb{E}\left[\sum_{j=1}^{d} \frac{\sigma_j^2}{(\sigma_j^2 + \lambda)^2}\right]$
- **Exercise**: show that the variance for $\mathbf{w}_{\text{MAP}}$ always lower than $\mathbf{w}_{\text{MLE}}$

Ch. 4: Optimization

- Second-order multivariate gradient descent
- Understanding why the Hessian in the second-order update accounts for differences in curvature in different dimensions
- Understanding the importance of an adaptive vector stepsize
- The mini-batch stochastic gradient descent (SGD) update rule
- Understanding why SGD is a more computationally efficient update than GD
- Understanding the momentum update

All the Updates

- Assumes we have $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$
- Second-order GD: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{H}^{-1}_{c(w_t)} \nabla c(\mathbf{w}_t)$
- First-order GD with vector stepsizes: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \boldsymbol{\eta}_t \cdot \nabla c(\mathbf{w}_t)$
  - element-wise product with stepsize
- Mini-batch SGD with vector stepsizes, using a mini-batch $\mathscr{B}$ of indices: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \boldsymbol{\eta}_t \cdot \frac{1}{b}\sum_{i \in \mathscr{B}} \nabla c_i(\mathbf{w}_t)$

Some optimization questions

- We use $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$ to grab b components of n for SGD
- But when we did LR we used $c(\mathbf{w}) = \sum_{i=1}^{n} c_i(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$. Is this mismatch a problem?
- How do we write the Ridge Regression loss as $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$?

Some optimization questions

- We use $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$. But when we did LR we used $c(\mathbf{w}) = \sum_{i=1}^{n} c_i(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$. Is this mismatch a problem?
- **Answer**: the constant 1/n in front does not change the solution. For OLS, it is really not necessary to include 1/n. When talking about GD and SGD, its useful to think of c as an expectation over losses per sample

What is the OLS solution for the normalized objective?

- $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w}) = \frac{1}{2n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ gives gradients
- $\frac{1}{n}\mathbf{X}^\top\mathbf{X}\mathbf{w} = \frac{1}{n}\mathbf{X}^\top\mathbf{y}$ and so $\mathbf{w} = \left(\frac{1}{n}\mathbf{X}^\top\mathbf{X}\right)^{-1}\frac{1}{n}\mathbf{X}^\top\mathbf{y}$
- Notice that the 1/n comes outside the inverse and becomes n
- $\mathbf{w} = \left(\frac{1}{n}\mathbf{X}^\top\mathbf{X}\right)^{-1}\frac{1}{n}\mathbf{X}^\top\mathbf{y} = n\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\frac{1}{n}\mathbf{X}^\top\mathbf{y} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y}$

Some optimization questions

- We use $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$. But when we did LR we used $c(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$. Is this mismatch a problem?
- How do we write the Ridge Regression loss as $c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w})$?

A normalized RR objective

- $\frac{1}{2}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\frac{\lambda}{2}\|\mathbf{w}\|_2^2$. What is the normalized c?
- $c(\mathbf{w})=\frac{1}{n}\sum_{i=1}^{n}c_i(\mathbf{w})=\frac{1}{2n}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\frac{\lambda}{2n}\|\mathbf{w}\|_2^2=\frac{1}{n}\left(\frac{1}{2}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$
- Therefore must have $c_i(\mathbf{w})=\frac{1}{2}(\mathbf{x}_i^\top\mathbf{w}-y_i)^2+\frac{\lambda}{2}\|\mathbf{w}\|_2^2$
- Makes very clear that regularizer has a diminishing role with increasing n

A normalized RR objective

- $\frac{1}{2}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\frac{\lambda}{2}\|\mathbf{w}\|_2^2$. What is the normalized c?
- $c(\mathbf{w})=\frac{1}{n}\sum_{i=1}^{n}c_i(\mathbf{w})=\frac{1}{2n}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\frac{\lambda}{2n}\|\mathbf{w}\|_2^2=\frac{1}{n}\left(\frac{1}{2}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$
- Therefore must have $c_i(\mathbf{w})=\frac{1}{2}(\mathbf{x}_i^\top\mathbf{w}-y_i)^2+\frac{\lambda}{2}\|\mathbf{w}\|_2^2$
- Makes very clear that regularizer has a diminishing role with increasing n
- **Question**: What is the mini-batch SGD update for RR?

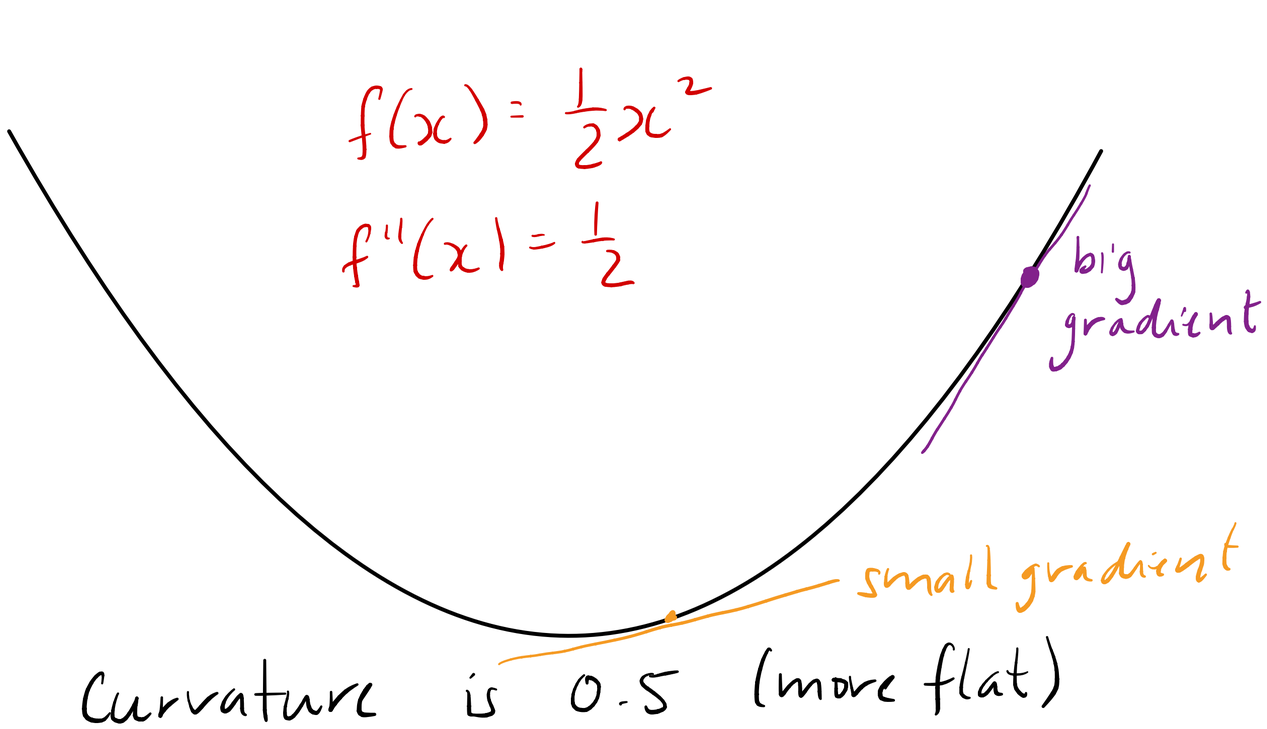
Mini-batch SGD for RR

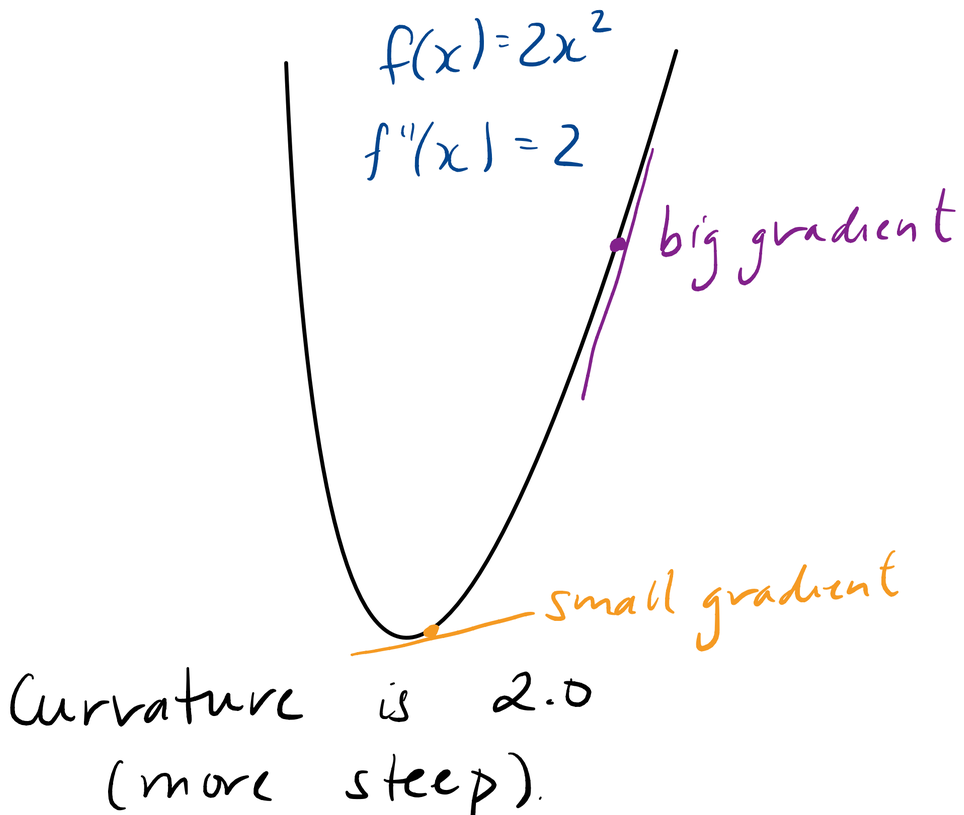
- Therefore must have $c_i(\mathbf{w}) = \frac{1}{2}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$
- **Question**: What is the mini-batch SGD update for RR?
- $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \boldsymbol{\eta}_t \cdot \frac{1}{b} \sum_{i \in \mathcal{B}} \nabla c_i(\mathbf{w}_t)$
- where $\nabla c_i(\mathbf{w}) = (\mathbf{x}_i^\top \mathbf{w} - y_i)\mathbf{x}_i + \lambda \mathbf{w}$

The Hessian and curvature

- The Hessian and second-derivative have a clear correspondence using the directional derivative
- The curvature (second-derivative) is about the shape of the bowl (wide flat bowl, or steep bowl)
- The gradient is at a specific point in that bowl, and can be big or small

Visualizing the difference





Second-order stepsize is always 2 here, for both gradients

Second-order stepsize is alway

The Hessian has two uses

- The Hessian also helps us know: are we in a local-min, local-max or potentially a saddlepoint?
- But this question only uses the sign of the eigenvalues of the Hessian. The magnitudes give additional information (about curvature)
  - Signs tell us type of bowl (convex or concave)
  - Magnitudes tells us the shape of the bowl
- We care more about Hessian approximations to approximate curvature

Understanding the Hessian

- Imagine we are at point $\mathbf{w}_t$ and we want to step in direction $\mathbf{u}$
- Easier to reason about curvature in the direction of $\mathbf{u}$ (essentially on a line)
- Define local function $g(\tau) = c(\mathbf{w}_t + \tau\mathbf{u})$, how functions changes as move τ in the direction of $\mathbf{u}$
- $g'(\tau)$ and $g''(\tau)$ tell us steepness of change and curvature at points along that direction and $g'(0)$ and $g''(0)$ tell us steepness/curvature at this current point
- Notice $g''(0) = \mathbf{u}^\top \mathbf{H}_{c(w_t)} \mathbf{u}$

Understanding the Hessian

- Imagine we are at point $\mathbf{w}_t$ and we want to step in direction $\mathbf{u}$
- Notice $g''(0) = \mathbf{u}^\top \mathbf{H}_{c(w_t)} \mathbf{u}$
- If $\mathbf{u}$ is an eigenvector of $\mathbf{H}_{c(w_t)}$ with eigenvalue λ, then
  $g''(0) = \mathbf{u}^\top \mathbf{H}_{c(w_t)} \mathbf{u} = \mathbf{u}^\top (\lambda) \mathbf{u} = \lambda \mathbf{u}^\top \mathbf{u} = \lambda$
- If $\mathbf{u} = \alpha_1 \mathbf{u}_1 + \alpha_2 \mathbf{u}_2$ for eigenvectors $\mathbf{u}_1, \mathbf{u}_2$ then
  $g''(0) = \mathbf{u}^\top \mathbf{H}_{c(w_t)} \mathbf{u} = \mathbf{u}^\top (\lambda_1 \alpha \mathbf{u}_1 + \lambda_2 \beta \mathbf{u}_2) = \ldots = \alpha^2 \lambda_1 + \beta^2 \lambda_2$

Question: can the eigenvalues be both positive and negative?

Momentum

- Replaces update with an exponential average of gradients
- $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \boldsymbol{\eta}_t \cdot \mathbf{g}_t$ becomes $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \boldsymbol{\eta}_t \cdot \mathbf{m}_{t+1}$ for either
- $\mathbf{m}_{t+1} = \mathbf{g}_t + \beta \mathbf{m}_t$ or normalized $\mathbf{m}_{t+1} = (1-\beta)\mathbf{g}_t + \beta \mathbf{m}_t$
- Smooths descent direction

Normalizing the momentum

- Equivalent to use $\mathbf{m}_{t+1} = \mathbf{g}_t + \beta \mathbf{m}_t$ or normalized $\mathbf{m}_{t+1} = (1-\beta)\mathbf{g}_t + \beta \mathbf{m}_t$
- To get the normalized one, equivalent to use $\mathbf{m}_{t+1} = \mathbf{g}_t + \beta \mathbf{m}_t$ and then normalize $(1-\beta)\mathbf{m}_{t+1}$; the normalization absorbed into the stepsize η
- Notice $\mathbf{m}_{t+1} = \mathbf{g}_t + \beta \mathbf{m}_t = \mathbf{g}_t + \beta \mathbf{g}_{t-1} + \beta^2 \mathbf{m}_{t-1} = \ldots = \sum_{i=0}^{t} \beta^i \mathbf{g}_{t-i}$
- $\mathbf{m}_{t+1} = (1-\beta)\mathbf{g}_t + \beta \mathbf{m}_t = (1-\beta)\mathbf{g}_t + \beta(1-\beta)\mathbf{g}_{t-1} + \beta^2 \mathbf{m}_{t-1} = \ldots = (1-\beta)\sum_{i=0}^{t} \beta^i \mathbf{g}_{t-i}$

Momentum vs RMSProp

- RMSProp slows down descent if several big gradients in a row
- Momentum seems to accelerate if so. What’s the deal?

Momentum vs RMSProp

- RMSProp slows down descent if several big gradients in a row
- Momentum seems to accelerate if so. What's the deal?
- **Answer**: we should think of momentum actually more as dampening.
- It takes an average of gradient, so it doesn't really accumulate large values (as long as we normalize, or make the stepsize out in front a bit smaller)
- But it nicely avoids oscillating when gradients change signs
- RMSProp does not as effectively prevent oscillation, since it just uses magnitude not sign

Convergence rates

- Typically opt for SGD if d and n are larger
  - Note: SGD means mini-batch SGD, using one sample per update is just a specific instance of mini-batch SGD with a batch size of b=1
- Our very approximate, big-O reasoning
  - Worth using 2nd order GD over GD if $d\epsilon \log(1/\epsilon) < 1$
  - Worth using SGD over GD if $b < \epsilon n$

Convergence rates

- Typically opt for SGD if d and n are larger
  - Note: SGD means mini-batch SGD, using one sample per update is just a specific instance of mini-batch SGD with a batch size of b=1
- Our very approximate, big-O reasoning
  - Worth using 2nd order GD over GD if $d\epsilon \log(1/\epsilon) < 1$
  - Worth using SGD over GD if $b < \epsilon n$

Convergence rates

- Typically opt for SGD if d and n are larger
- Our very approximate, big-O reasoning
  - Worth using 2nd order GD over GD if $d\epsilon \log(1/\epsilon) < 1$
  - Worth using SGD over GD if $b < \epsilon n$
- How does Adagrad or Adam change this?
  - Typically changes the constants in the bounds, not the rates
  - Very much matters practically, but not at this higher-level big-O view

Ch. 5: Generalized Linear Models

- Understand the purpose of the generalization from linear regression to GLMs
- Understand that the exponential family distribution underlies GLMs
- Know that linear regression, Poisson regression, logistic regression and multinomial logistic regression are examples of GLMs
- Know the distributions and transfers that correspond to each of these four GLMs
  - e.g., Poisson regression has a Poisson distribution p(y | x) with transfer exp

Generalized Linear Models (GLMs)

- Generalizes linear regression and p(y | **x**) a Gaussian: allows p(y |**x**) to be any natural exponential family distribution with natural parameter $\theta(\mathbf{x})$
- In GLMs, we learn the natural parameter $\theta(\mathbf{x}) = \mathbf{x}\mathbf{w}$
- Then $\mathbb{E}[Y \mid \mathbf{x}] = g(\theta(\mathbf{x}))$ for transfer function g
  - e.g., Gaussian with fixed (unknown) variance has g = identity
  - e.g., Bernoulli has $g = \sigma$ (i.e., $\sigma(\theta(\mathbf{x})) = \mathbb{E}[Y \mid \mathbf{x}]$)
  - e.g., Poisson p(y | **x**) has $g = \exp$
  - e.g., Multinomial (categorial) p(y | **x**) for multi-class has g = softmax

Exponential Family Distribution

- Generalize from $p(y \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}^\top \mathbf{w}, \sigma^2)$ to a wider set of distributions
- $p(y \mid \mathbf{x}) = \exp(\theta(\mathbf{x})y - a(\theta(\mathbf{x})) + b(y))$ for $\theta(\mathbf{x}) = \mathbf{x}\mathbf{w}$
- for log-partition function $a : \mathbb{R} \to \mathbb{R}$ where the transfer $g = a'$ is the derivative of a
- More generally, y can also be multivariate giving. Let $\mathbf{y}$ be a row vector.
- $p(\mathbf{y} \mid \mathbf{x}) = \exp(\langle \theta(\mathbf{x}), \mathbf{y} \rangle - a(\theta(\mathbf{x})) + b(y))$ for $\theta(\mathbf{x}) = \mathbf{x}\mathbf{W}$
- and where the log-partition function a inputs vectors instead of scalars

Exponential Family Distribution

- Generalize from $p(y \mid \mathbf{x}) = \mathcal{N}(\mathbf{x}^\top \mathbf{w}, \sigma^2)$ to a wider set of distributions
- More generally, y can also be multivariate giving. Let $\mathbf{y}$ be a row vector.
- $p(\mathbf{y} \mid \mathbf{x}) = \exp(\langle \theta(\mathbf{x}), \mathbf{y} \rangle - a(\theta(\mathbf{x})) + b(y))$ for $\theta(\mathbf{x}) = \mathbf{x}\mathbf{W}$
- and where the log-partition function a inputs vectors instead of scalars
- Using $g = \nabla a$ and $\theta(\mathbf{x}) = \mathbf{x}\mathbf{W}$ with log-likelihood results in a convex optimization (univariate or multivariate)
- **Question**: why is it useful that this is a convex optimization?

Proximal operators

- Optimize with smooth $c(\mathbf{w})$ and non-smooth $r(\mathbf{w})$
- To solve $\min_{\mathbf{w}\in\mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$ we break it into two steps
- GD Step: $\tilde{\mathbf{w}}_{t+1} = \mathbf{w}_t - \eta \nabla c(\mathbf{w}_t)$
- Projection step: $\mathbf{w}_{t+1} = \text{prox}_{\eta r}(\tilde{\mathbf{w}}_{t+1})$ where the proximal operator is
- $\text{prox}_{\eta r}(\mathbf{u}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w} - \mathbf{u}\|_2^2 + \eta r(\mathbf{w})$

Why so many subscripts in prox?

- $\text{prox}_{\eta r}(\mathbf{u}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}-\mathbf{u}\|_2^2 + \eta r(\mathbf{w})$
- Really, we mean $\text{prox}_{f}(\mathbf{u}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}-\mathbf{u}\|_2^2 + f(\mathbf{w})$ for function f
- In our updates, we have functions like $\lambda\ell_1(\mathbf{w})$ and then have the stepsize out front too, so write
- $\text{prox}_{\eta\lambda\ell_1}(\mathbf{u}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}-\mathbf{u}\|_2^2 + \eta\lambda\ell_1(\mathbf{w})$

Why does our project involve the stepsize?

- The GD step changes $\mathbf{w}$ by stepsize η amount, so we need to apply the projection that amount also to ensure they eventually balance each other out
- Example: $\text{prox}_{\eta\lambda\ell_1}(u) = \begin{cases} u - \eta\lambda & \text{if } u > \eta\lambda \\ u + \eta\lambda & \text{if } u < -\eta\lambda \\ 0 & \text{else} \end{cases}$
- At convergence, for larger $w_{t,j}$, GD step has does $\tilde{w}_{t+1,j} = w_{t,j} + \eta\lambda$ and the proximal update returns $w_{t+1,j} = \tilde{w}_{t+1,j} - \eta\lambda = w_{t,j}$ (no change, done opt)
- And for $w_{t,j}$ more negative, GD step has does $\tilde{w}_{t+1,j} = w_{t,j} - \eta\lambda$ and the proximal update returns $w_{t+1,j} = \tilde{w}_{t+1,j} + \eta\lambda = w_{t,j}$ (no change, done opt)

Why is it useful to break up into two steps?

- Solving $\text{prox}_{\eta r}(\mathbf{u}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}-\mathbf{u}\|_2^2 + \eta r(\mathbf{w})$ with simple loss $\|\mathbf{w}-\mathbf{u}\|_2^2$ is likely a lot simpler than solving $\min_{\mathbf{w}\in\mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$ for arbitrary $c(\mathbf{w})$
- Example, for $r(w) = \ell_1(w)$, can reason about balancing error $(w-u)^2$ versus error $\ell - 1(w) = |w|$.
  - e.g., $u = 0.1$, then at $w = 0$, $(w-u)^2 = 0.01$. If increase w to 0.1, then $(w-u)^2 = 0$ but $|w| = 0.1$. Worse error! In fact, w = 0 is optimal because these small differences to u are squared

Why is it useful to break up into two steps?

- Solving $\text{prox}_{\eta r}(\mathbf{u}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{w}-\mathbf{u}\|_2^2 + \eta r(\mathbf{w})$ with simple loss $\|\mathbf{w}-\mathbf{u}\|_2^2$ is likely a lot simpler than solving $\min_{\mathbf{w}\in\mathbb{R}^d} c(\mathbf{w}) + r(\mathbf{w})$ for arbitrary $c(\mathbf{w})$
- Example, for $r(w) = \ell_1(w)$, can reason about balancing error $(w-u)^2$ versus error $\ell - 1(w) = |w|$.
- For $c(w)$ the cross-entropy loss, how do we do this reasoning?

Cross-validation

- When have lots and lots of data, might just do train-validation-test split
  - Train models on training data, use validation to select hyperparameters
  - Example: might train with $\lambda = 0$, $\lambda = 0.01$ and $\lambda = 0.1$ and pick the best using the validation set (instead of doing internal CV)
  - Once select λ, then retrain on train+validation, and evaluation on test before deploying (instead of doing external CV)
- When we have less data, want to use all the data for training, validation & test