Advanced Introduction to Machine Learning
— Spring Quarter, Week 5 —
https://canvas.uw.edu/courses/1372141

Prof. Jeff Bilmes

University of Washington, Seattle
Departments of: Electrical & Computer Engineering, Computer Science & Engineering
http://melodi.ee.washington.edu/~bilmes

April 27th/29nd, 2020
Announcements

- HW2 is posted. Due May 1st, 6:00pm via our assignment dropbox (https://canvas.uw.edu/courses/1372141/assignments).
- Virtual office hours this week, Thursday night at 10:00pm via zoom (same link as class).
Class Road Map

- W1(3/30, 4/1): What is ML, Probability, Coins, Gaussians and linear regression, Associative Memories, Supervised Learning
- W2(4/6, 4/8): More supervised, logistic regression, complexity and bias/variance tradeoff
- W6(5/4, 5/6): Curse of Dimensionality, Dimensionality Reduction, $k$-NN
- W7(5/11, 5/13): $k$-NN, LSH, DTs, Bootstrap/Bagging, Boosting & Random Forests, GBDTs
- W8(5/18, 5/20): Graphs; Graphical Models (Factorization, Inference, MRFs, BNs);
- W9(5/27, 6/1): Learning Paradigms; Clustering; EM Algorithm;
- W10(6/3, 6/8): Spectral Clustering, Graph SSL, Deep models, (SVMs, RL); The Future.

Last lecture is 6/8 since 5/25 is holiday (or we could just have lecture on 5/25).
1. Introduction
• What is ML
• What is AI
• Why are we so interested in these topics right now?

2. ML Paradigms/Concepts
• Overfitting/Underfitting, model complexity, bias/variance
• size of data, big data, sample complexity
• ERM, loss + regularization, loss functions, regularizers
• Supervised, unsupervised, and semi-supervised learning:
  • reinforcement learning, RL, multi-agent, planning/control
  • transfer and multi-task learning
  • federated and distributed learning
  • active learning, machine teaching
  • self-supervised, zero/one-shot, open-set learning

3. Dealing with Features
• dimensionality reduction, PCA, LDA, MDS, T-SNE, UMAP
• Locality sensitive hashing (LSH)
• feature selection
• feature engineering
• matrix factorization & feature engineering
• representation learning

4. Evaluation
• accuracy/error, precision/recall, ROC, likelihood/posterior, cost/utility, margin
• train/eval/test data splits
• n-fold cross validation
• method of the bootstrap

5. Optimization Methods
• Unconstrained Continuous Optimization: (stochastic) gradient descent (SGD), adaptive learning rates, conjugate gradient, 2nd order Newton
• Constrained Continuous Optimization: Frank-Wolf (conditional gradient descent), projected gradient, linear, quadratic, and convex programming
• Discrete optimization - greedy, beam search, branch-and-bound, submodular optimization.

6. Inference Methods
• probabilistic inference
• MLE, MAP
• belief propagation
• forward/backpropagation
• Monte Carlo methods

7. Models & Representation
• linear least squares, linear regression, logistic regression, sparsity, ridge, lasso
• generative vs. discriminative models
• Naive Bayes
• k-nearest neighbors
• clustering, k-means, k-mediods, EM & GMMs, single linkage
• decision trees and random forests
• support vector machines, kernel methods, max margin
• perceptron, neural networks, DNNs
• Gaussian processes
• Bayesian nonparametric methods
• ensemble methods
• the bootstrap, bagging, and boosting
• graphical models
• time-series, HMMs, DBNs, RNNs, LSTMs, Attention, Transformers
• structured prediction
• grammars (as in NLP)

8. Philosophy, Humanity, Spirituality
• artificial intelligence (AI)
• artificial general intelligence (AGI)
• artificial intelligence vs. science fiction

9. Applications
• computational biology
• social networks
• computer vision
• speech recognition
• natural language processing
• information retrieval
• collaborative filtering/matrix factorization

10. Programming
• python
• libraries (e.g., NumPy, SciPy, matplotlib, scikit-learn (sklearn), pytorch, CNTK, Theano, tensorflow, kers, H2O, etc.)
• HPC: C/C++, CUDA, vector processing

11. Background
• linear algebra
• multivariate calculus
• probability theory and statistics
• information theory
• mathematical (e.g., convex) optimization

12. Other Techniques
• compressed sensing
• submodularity, diversity/homogeneity modeling

---

Performance

- Excellent
- Good
- Poor

Input Layer 1
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Hidden Layer 4
Hidden Layer 5
Hidden Layer 6
Hidden Layer 7
Output

12. Determinant
Let A be an n x n matrix.

\[ \det(A) = \prod_{i=1}^{n} A_{ii} \quad \det(A^T) = \det(A) \quad \det(AB) = \det(A) \det(B) \]

\[ \det(I + uu^T) = 1 + u^T \]

---
Multiclass and Voronoi tessellations

- $\ell$ classes, $j \in \{1, 2, \ldots, \ell\}$, parameters $\theta^{(j)} \in \mathbb{R}^m$, $\ell$ linear discriminant functions $h_{\theta^{(j)}}(x) = \langle \theta^{(j)}, x \rangle$.
- Partition input space $x \in \mathbb{R}^m$ based on who wins

\[
\mathbb{R}^m = \mathbb{R}^m_1 \cup \mathbb{R}^m_2 \cup \cdots \cup \mathbb{R}^m_\ell
\]  

(5.19)

where $\mathbb{R}^m_i \subseteq \mathbb{R}^m$ and $\mathbb{R}^m_i \cap \mathbb{R}^m_j = \emptyset$ whenever $i \neq j$.
- Winning block, for all $i \in [\ell]$:

\[
\mathbb{R}^m_i \triangleq \{ x \in \mathbb{R}^m : h_{\theta^{(i)}}(x) > h_{\theta^{(j)}}(x), \forall j \neq i \}
\]  

(5.20)

- Alternatively,

\[
\mathbb{R}^m_i \triangleq \left\{ x \in \mathbb{R}^m : \langle \theta^{(i)} - \theta^{(j)}, x \rangle > 0, \forall j \neq i \right\}
\]  

(5.21)

consider when $\langle \cdot, \cdot \rangle$ is dot product.
- Alternatively, under softmax regression model,

\[
\mathbb{R}^m_i \triangleq \{ x \in \mathbb{R}^m : \Pr(C_i|x) > \Pr(C_j|x), \forall j \neq i \}
\]  

(5.22)
0/1-loss and Bayes error

- Let $L$ be 0/1, i.e., $L(y, y') = 1_{\{y \neq y'\}}$.
- Probability of error for a given $x$

$$\Pr(h_\theta(x) \neq Y) = \int p(y|x) 1_{\{h_\theta(x) \neq y\}} dy = 1 - p(h_\theta(x)|x) \quad (5.7)$$

- To minimize the probability of error, $h_\theta(x)$ should ideally choose a class having maximum posterior probability for the current $x$.
- Smallest probability of error is known is Bayes error for $x$

$$\text{Bayes error}(x) = \min_y (1 - p(y|x)) = 1 - \max_y p(y|x) \quad (5.8)$$

- Bayes classifier (or predictor) predicts using highest probability, assuming have access to $p(y|x)$. I.e.,

$$\text{BayesPredictor}(x) \triangleq \arg\max_y p(y|x) \quad (5.9)$$

- Bayes predictor has 'Bayes error' as its error. Irreducible error rate.
Bayes Error, overall difficulty of a problem

- For Bayes error, we often take the expected value w.r.t. $x$. This gives an indication overall of how difficult a classification problem is, since we can never do better than Bayes error.

- Bayes error, various equivalent forms (Exercise: show last equality):

  \[
  \text{Bayes Error} = \min_h \int p(x, y) \mathbf{1}_{h(x) \neq y} dy dx \quad (5.13)
  \]

  \[
  = \min_h \Pr(h(X) \neq Y) \quad (5.14)
  \]

  \[
  = E_p(x) \left[ \min(p(y|x), 1 - p(y|x)) \right] \quad (5.15)
  \]

  \[
  = \frac{1}{2} - \frac{1}{2} E_p(x) \left[ \left| 2p(y|x) - 1 \right| \right] \quad (5.16)
  \]

- For binary classification, if Bayes error is 1/2, prediction can never be better than flipping a coin (i.e., $x$ tells us nothing about $y$).

- Bayes error, property of the distribution $p(x, y)$; can be useful to decide between say $p(x, y)$ vs. $p(x', y)$ for two different feature sets $x$ and $x'$. 
ERM and complexity

- Empirical risk minimization on data $\mathcal{D}$:

$$
\hat{h} \in \arg\min_{h \in \mathcal{H}} \hat{\text{Risk}}(h) = \frac{1}{n} \sum_{i=1}^{n} 1\{h(x(i)) \neq y(i)\} 
$$

(5.13)

$\hat{h}$ is a random variable, computed based on random data set $\mathcal{D}$.

- How close is $\hat{h}$ to true risk, the one for the best hypothesis $h^*$?

$$
h^* \in \arg\min_{h \in \mathcal{H}} \text{Risk}(h) = \int p(x, y) 1\{h(x) \neq y\} dx dy 
$$

(5.14)

- We can sometimes bound the probability (e.g., PAC-style bounds)

$$
\Pr(\text{Risk}(\hat{h}) - \text{Risk}(h^*) > \epsilon) < \delta(n, \text{VC, } \epsilon) 
$$

(5.15)

where $\delta$ is function of sample size $n$, and complexity (ability, flexibility, expressiveness) of family of models $\mathcal{H}$ over which we’re learning. Bound gets worse with increasing complexity of $\mathcal{H}$ or decreasing $n$.

- Extensive mathematical theory behind this (e.g., Vapnik-Chervonenkis (VC) dimension, or Rademacher complexity).
Generative vs. Discriminative models

- Generative models, given data drawn from $p(x, y)$ model the entire distribution, often factored as $p(x|y)p(y)$ where $y$ is the class label (or prediction) random variable and $x$ is the feature vector random variable.
Generative vs. Discriminative models

- Generative models, given data drawn from $p(x, y)$ model the entire distribution, often factored as $p(x|y)p(y)$ where $y$ is the class label (or prediction) random variable and $x$ is the feature vector random variable.

- Discriminative model: When goal is classification/regression, can sometimes be simpler to just model $p(y|x)$ since that is sufficient to achieve Bayes error (i.e., $\arg\max_y p(y|x)$ for classification) or $E[Y|x]$ for regression. (e.g., support vector machines, max-margin learning, conditional likelihood, etc.)
Generative vs. Discriminative models

- Generative models, given data drawn from $p(x, y)$ model the entire distribution, often factored as $p(x|y)p(y)$ where $y$ is the class label (or prediction) random variable and $x$ is the feature vector random variable.

- Discriminative model: When goal is classification/regression, can sometimes be simpler to just model $p(y|x)$ since that is sufficient to achieve Bayes error (i.e., $\arg\max_y p(y|x)$ for classification) or $E[Y|x]$ for regression. (e.g., support vector machines, max-margin learning, conditional likelihood, etc.)

- Sometimes they are the same. Naïve Bayes class-conditional Gaussian generative model $p(x|y)$ leads to logistic regression.
Accuracy Objectives

Decreasing order of complexity of the task being learnt, while still solving a classification decision problem. From strongest to weakest goals.

- **generative modeling**, model $p(x, y)$ and compute $\text{argmax}_y p(x, y)$ for given $x$. Maximum likelihood principle.

- **discriminative modeling**, model only $p(y|x)$ and compute $\text{argmax}_y p(y|x)$. Maximum conditional likelihood principle.

- **rank modeling**, rank function model $f(y|x)$ so that $f(\sigma_1|x) \geq f(\sigma_2|x) \geq \cdots \geq f(\sigma_\ell|x)$ whenever $p(\sigma_1|x) \geq p(\sigma_2|x) \geq \cdots \geq p(\sigma_\ell|x)$.

- **best decision modeling**, decision function $h_y(x) \geq h_{y'}(x)$ whenever $p(y|x) \geq p(y'|x)$ for all $y' \neq y$. Discriminant functions.

Weaker model often lives within stronger model (e.g., logistic or softmax uses linear discriminant functions) as we saw, thereby giving it improved interpretation (e.g., probabilities rather than just unnormalized scores).
Naïve Bayes classifier

- Recall, $x \in \mathbb{R}^m$, so $m$ feature values in $x = (x_1, x_2, \ldots, x_m)$ used to predict $y$ in joint model $p(x, y)$.
- Generative conditional distribution $p(x, y) = p(x|y)p(y)$. Conditioned on $y$, $p(x|y)$ can be an arbitrary distribution.
- Naïve Bayes classifiers make simplifying (factorization) assumption.

$$p(x|y) = \prod_{j=1}^{m} p(x_j|y)$$

(5.21)

or that $X_i \perp X_j | Y$, read “$X_i$ and $X_j$ are conditionally independent given $Y$, for all $i \neq j$. This does not mean that $X_i \perp X_j$.

- Estimating many 1-dimensional distributions $p(x_j|y)$ much easier than full joint $p(x|y)$ due to curse of dimensionality.
- Naïve Bayes classifiers make a greatly simplifying assumption but often works quite well.
Posterior From Binary Naïve Bayes Gaussian

- Binary case $y \in \{0, 1\}$, Naïve Bayes Gaussian classifier posterior probability:

  $$p(y = 1 | x) = \frac{p(x | y = 1) p(y = 1)}{p(x | y = 1) p(y = 1) + p(x | y = 0) p(y = 0)}$$  
  \( (5.23) \)

  $$= \frac{1}{1 + \frac{p(x | y = 0) p(y = 0)}{p(x | y = 1) p(y = 1)}}$$  
  \( (5.24) \)

  $$= \frac{1}{1 + \exp \left( - \log \frac{p(x | y = 1) p(y = 1)}{p(x | y = 0) p(y = 0)} \right)}$$  
  \( (5.25) \)

  $$= g \left( \sum_{i=1}^{m} \log \frac{p(x_i | y = 1)}{p(x_i | y = 0)} + \log \frac{p(y = 1)}{p(y = 0)} \right)$$  
  \( (5.26) \)

where $g(z) = 1/(1 + e^{-z})$ is the logistic function, $\log p(y = 1)/p(y = 0)$ is log odds-ratio.
Binary ($\ell = 2$) Naïve Bayes Gaussian

Since $p(x_i|y)$ is univariate Gaussian, $\log \frac{p(x_i|y=1)}{p(x_i|y=0)} + \log \frac{p(y=1)}{p(y=0)}$ takes one of the two following forms:

1. $b_i x_i + c_i$ if $\sigma_{1,i} = \sigma_{2,i}$ $\forall i$ (equal class variances). Hence, we have a linear model $p(y=1|x) = g(\theta^T x + \text{const})$ like we saw in Logistic regression!!

2. $a_i x_i^2 + b_i x_i + c_i$ otherwise, so we have a quadratic model followed by the logistic function. $p(y=1|x) = g(\sum_{i=1}^{m} a_i x_i^2 + b_i x_i + c_i)$. Unequal class variances introduce quadratic features. Alternatively, we could have just started with features like $x_i^2$, increasing $m$, without need for unequal class variances. Still no cross interaction terms of the from $x_i x_j$ due to Naïve Bayes assumption! But can introduce those $x_i x_j$ cross features as well, further increasing $m$, feature augmentation.

3. non-Naïve-Bayes with fewer features is identical to doing Naïve Bayes with more features. This helps to explain why Naïve Bayes often works so well, since one can remove unfounded assumptions simply by adding features.
Bottom line: Binary Naive Bayes Gaussian (equivalently, two-class Gaussian generative model classifiers) is equivalent to logistic regression when we form the posterior distribution $p(y|x)$ for classification. We can eliminate the Naive Bayes property in original feature space by adding more features (feature augmentation).

In general, feature engineering (producing and choosing right features) is an important part of any machine learning system.

Representation learning - the art and science of using machine learning to learn the features themselves, often using a deep neural network. Self-supervised learning can facilitate this (see final lecture).

Disentanglement - the art and science of taking a feature representation and producing new deduced features where separate phenomena in the signal have separate representation in the deduced features. A modern form of ICA (independent component analysis).
Norms and $p$-norms

For any real valued $p \geq 1$ and vector $x \in \mathbb{R}^m$, we have a norm:

$$\|x\|_p \triangleq \left( \sum_{j=1}^{m} |x_j|^p \right)^{1/p}$$

Properties of any “norm” $\|x\|$ for $x \in \mathbb{R}^m$ are:

1. $\|x\| \geq 0$ for any $x \in \mathbb{R}^m$.
2. $\|x\| = 0$ iff $x = 0$.
3. Absolutely homogeneous: $\|\alpha x\| = |\alpha|\|x\|$ for any $\alpha \in \mathbb{R}$.
4. Triangle inequality: For any $x, y \in \mathbb{R}^m$, we have $\|x + y\| \leq \|x\| + \|y\|$.

Euclidean norm: The 2-norm (where $p = 2$) is the standard Euclidean distance (from zero) that we use in everyday life.

$p$-norm is also defined and is a valid norm for $p \rightarrow \infty$ in which case

$$\|x\|_\infty = \max_{j=1}^{m} |x_j|$$

There are other norms besides $p$ norms.
The Lasso and Constrained Optimization

- Lasso adds penalty to learnt weights

\[
\tilde{\theta} \in \arg\min_{\theta} \frac{1}{2} \sum_{i=1}^{n} \frac{1}{2} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \|\theta\|_1
\]  

(5.25)

where \(\|\cdot\|_1\) is known as the 1-norm, defined as \(\|\theta\|_1 = \sum_{j=1}^{m} |\theta_j|\).

- Equivalent formulation:

\[
\text{minimize } \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2
\]

subject to \(\|\theta\|_1 < t\)  

(5.26)

(5.27)

- One-to-one correspondence between \(\lambda\) and \(t\) yielding identical solution (see Boyd and Vandenberghe’s book on Convex Analysis).

- No effect if \(t > \sum_{i=1}^{m} |\tilde{\theta}_i|\) where \(\tilde{\theta}\) is the least squares estimate.
Sparsity

- Consider linear model $h_\theta(x) = \langle \theta, x \rangle$, where $x \in \mathbb{R}^m$.
- $m$ might be very big since we don’t know what features to use.
- Useful: allow learning system to throw away needless features, throw away means the resulting learnt model has $\theta_j = 0$. Helps with:
  - reduce computation, memory, and resources (e.g., sensor requirements)
  - improve prediction (reduce noise variables). Trade off a bit of bias increase in order to reduce variance (to improve overall accuracy).
  - Helps ease interpretation (what from $x$ is critical to predict $y$). Remove dirty variables (data hygiene).
  - Reduce redundant cancellation, i.e., if typically $\theta_i x_i \approx -\theta_j x_j$, no effect on prediction even if both $\theta_i$ and $\theta_j$ are not zero.
- Effective new model has $m' \ll m$ features.
Feature selection in ML

- Feature Selection: a branch of feature engineering.
- Many applications start with massive feature sets, more than is necessary.
  - Example: spam detection, natural language processing (NLP) and n-gram features.
  - Cancer detection in computational biology, k-mer motifs, gene expression via DNA microarrays, mRNA abundance, biopsies.
- Important in particular when $m > n$ but potentially useful for any $n, m$ when features are redundant/superfluous, irrelevant, or expensive.
- Overall goal, want least costly but still informative feature set.
- Sparsity (via shrinkage methods) is one way to do this, but other discrete optimization methods too (e.g., the greedy procedure).
- Many are implemented in python’s sklearn: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selection
Greedy Forward Selection Heuristic

- Let \( x \in \mathbb{R}^m \). Let the features be indexed integer set \( U = \{1, 2, \ldots, m\} \), and for \( A \subseteq U \) let \( x_A = (x_{a_1}, x_{a_2}, \ldots, x_{a_k}) \) be the features indexed by subset \( A \), where \( |A| = k \). Similarly \( \theta_A \) is so defined.
- Let \( h_{\theta_A}(x_A) \) be the best learnt model using only features \( A \) giving parameters \( \theta_A \) and accuracy \( (A) \approx 1 - \Pr(h_{\theta_A}(X_A) \neq Y) \).
- To find best set \( (\max_{A \subseteq U: |A| = k} \text{accuracy}(A)) \), need to consider \( \binom{m}{k} = O((me/k)^k) \) subsets by Stirling’s formula, exponential in \( m \).
- Greedy forward selection to select \( k \) features, achieve sparsity, \( O(mk) \).

**Algorithm 1:** Greedy Forward Selection

1. \( A \leftarrow \emptyset \);
2. for \( i = 1, \ldots, k \) do
3. \( \quad \text{Choose } v' \in \text{argmax}_{v \in U \setminus A} \text{accuracy}(A \cup \{v\}) \); 
4. \( \quad A \leftarrow A \cup \{v'\} ; \)

- Better than \( \binom{n}{k} \), still need to retrain model each time.
- One solution model accuracy (e.g., submodularity, later). Another solution, Lasso regression (next).
Forward vs. Backwards Selection

- Rather than start from the empty set and add one at a time, we can:
  - Start with all features and greedily remove the feature that hurts us the least. **backwards selection**
- This is also only a heuristic (might not work well).
- Backwards selection, variance is an issue when using many features.
- Bidirectional greedy: pick a random order, consider both adding starting at empty set, and removing starting at all features, based on order, and randomly choose based on distribution proportional to both:
  1. **accuracy gain** $\text{accuracy}(A \cup \{v\}) - \text{accuracy}(A)$ and
  2. **accuracy loss** $\text{accuracy}(S) - \text{accuracy}(S \setminus \{v\})$. 
Ridge regression adds sum-of-squares penalty to learnt coefficients. penalty to learnt weights

\[ \tilde{\theta} \in \arg\min_{\theta} J_{\text{ridge}}(\theta) = \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \|\theta\|_2^2 \]  

(5.1)

where \( \| \cdot \|_2 \) is known as the 2-norm, defined as \( \|\theta\|_2 = \sqrt{\sum_{j=1}^{m} \theta_j^2} \).
Ridge regression adds sum-of-squares penalty to learnt coefficients. penalty to learnt weights

$$\tilde{\theta} \in \arg\min_{\theta} J_{\text{ridge}}(\theta) = \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \|\theta\|_2^2$$  \hspace{1cm} (5.1)

where \(\| \cdot \|_2\) is known as the 2-norm, defined as \(\|\theta\|_2 = \sqrt{\sum_{j=1}^{m} \theta_j^2}\).

Lasso linear regression

$$\tilde{\theta} \in \arg\min_{\theta} J_{\text{lasso}}(\theta) = \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \|\theta\|_1$$  \hspace{1cm} (5.2)

where \(\| \cdot \|_1\) is known as the 1-norm, defined as \(\|\theta\|_1 = \sum_{j=1}^{m} |\theta_j|\).
We have (consistent w. picture) that if $0 < r < p$ then $k x_k r < k x_k p$, thus:

$$\{x \in \mathbb{R}^m : \|x\|_\infty = 1 \} \subset \{x \in \mathbb{R}^m : \|x\|_2 = 1 \} \subset \{x \in \mathbb{R}^m : \|x\|_1 = 1 \}$$

Note, $k x_k r$ is known as the infinity-norm (or max-norm).
We have (consistent w. picture) that if $0 < r < p$ then $\|x\|_r \geq \|x\|_p$, thus:

$$\{ x \in \mathbb{R}^m : \|x\|_1 \leq 1 \} \subset \{ x \in \mathbb{R}^m : \|x\|_2 \leq 1 \} \subset \{ x \in \mathbb{R}^m : \|x\|_\infty \leq 1 \}$$

Note, $\|x\|_\infty = \max_i x_i$ is known as the infinity-norm (or max-norm).
More $p$-norm unit ball contours (in 2D)

Right: various $p$-norm balls $p \geq 1$ and analogous $p < 1$.
For $p = 0$, we get the L0 ball, which counts the number of non-zero entries and is most sparse encouraging. The L1 ball is the convex envelope of the L0 ball, so it is arguably the best combination of easy & appropriate.
Linear Regression, Ridge, Lasso

\[ \tilde{\theta} \in \mathbb{R}^m : \|\theta\|_1 \leq t \]

\[ \{ \theta : \sum_{i=1}^{n} (y(i) - \theta^T x(i))^2 = c \} \]

\[ \tilde{\theta} \in \mathbb{R}^m : \|\theta\|_2 \leq t \]

\[ \{ \theta : \sum_{i=1}^{n} (y(i) - \theta^T x(i))^2 = c \} \]

\( \tilde{\theta} \) is the linear least squares solution, with quadratic contours of equal error. For given \( \lambda \), solution trades off regularizer cost w. error cost. From Hastie et. al.
Other norms as regularizers

Using $p$-norms with $p < 1$ encourages sparsity even further (but harder to optimize).

\[ \| \theta \|_p = \left( \sum_{i=1}^{n} \left| \theta_i \right|^p \right)^{1/p} \]
Lasso: How to solve

- Closed form solution for ridge, no closed form for lasso.
Lasso: How to solve

- Closed form solution for ridge, no closed form for lasso.
- We used gradient descent (or SGD) for ridge. Lasso regression is not differentiable because $|\theta|$ is not differentiable at $\theta = 0$.

\[
\frac{\partial}{\partial \theta} |\theta| (\theta) = \begin{cases} 
1 & \text{if } \theta > 0 \\
-1 & \text{if } \theta < 0 \\
?? & \text{if } \theta = 0
\end{cases}
\]  

(5.3)
Lasso: How to solve

- Closed form solution for ridge, no closed form for lasso.
- We used gradient descent (or SGD) for ridge. Lasso regression is not differentiable be since $|\theta|$ is not differentiable at $\theta = 0$.

\[
\frac{\partial}{\partial \theta} |\theta|(\theta) = \begin{cases} 
1 & \text{if } \theta > 0 \\
-1 & \text{if } \theta < 0 \\
?? & \text{if } \theta = 0 
\end{cases} 
\]  

(5.3)

- How to solve? Many possible ways.
Lasso: How to solve

- Closed form solution for ridge, no closed form for lasso.
- We used gradient descent (or SGD) for ridge. Lasso regression is not differentiable because $|\theta|$ is not differentiable at $\theta = 0$.

\[
\frac{\partial}{\partial \theta} |\theta|(\theta) = \begin{cases} 
1 & \text{if } \theta > 0 \\
-1 & \text{if } \theta < 0 \\
?? & \text{if } \theta = 0 
\end{cases} 
\] (5.3)

- How to solve? Many possible ways.
- Simple and effective: coordinate descent, where we find a subgradient of objective $J_{\text{lasso}}(\theta)$ w.r.t. one element of $\theta$ at a time and solve it.
Lasso: How to solve

- Closed form solution for ridge, no closed form for lasso.
- We used gradient descent (or SGD) for ridge. Lasso regression is not differentiable be since $|\theta|$ is not differentiable at $\theta = 0$.

\[
\frac{\partial |\theta|}{\partial \theta}(\theta) = \begin{cases} 
1 & \text{if } \theta > 0 \\
-1 & \text{if } \theta < 0 \\
?? & \text{if } \theta = 0 
\end{cases} \tag{5.3}
\]

- How to solve? Many possible ways.
- Simple and effective: coordinate descent, where we find a subgradient of objective $J_{\text{lasso}}(\theta)$ w.r.t. one element of $\theta$ at a time and solve it.
- Use subgradient descent (see https://see.stanford.edu/materials/lsocoee364b/01-subgradients_notes.pdf for full details).
Lasso: How to solve

- Closed form solution for ridge, no closed form for lasso.
- We used gradient descent (or SGD) for ridge. Lasso regression is not
  differentiable because $|\theta|$ is not differentiable at $\theta = 0$.

$$\frac{\partial}{\partial \theta} |\theta|(\theta) = \begin{cases} 
1 & \text{if } \theta > 0 \\
-1 & \text{if } \theta < 0 \\
?? & \text{if } \theta = 0
\end{cases}$$  \hspace{1cm} (5.3)

- How to solve? Many possible ways.
- Simple and effective: coordinate descent, where we find a subgradient
  of objective $J_{\text{lasso}}(\theta)$ w.r.t. one element of $\theta$ at a time and solve it.
- Use subgradient descent (see https://see.stanford.edu/materials/lsocoee364b/01-subgradients_notes.pdf for full
details).
- Results in simple solution, soft thresholding
Coordinate Descent

- Isolate one coordinate $k$ fix the remainder $j \neq k$

\[
\sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \|\theta\|_1 = \sum_{i=1}^{n} \left( y^{(i)} - \sum_{j=1}^{m} \theta_j x^{(i)}_j \right)^2 + \lambda \|\theta\|_1
\]  

(5.4)

\[
= \sum_{i=1}^{n} \left( y^{(i)} - \theta_k x^{(i)}_k - \sum_{j \neq k} \theta_j x^{(i)}_j \right)^2 + \lambda \sum_{j \neq k} |\theta_j| + \lambda |\theta_k|
\]  

(5.5)

\[
= \sum_{i=1}^{n} (r^{(i)}_k - \theta_k x^{(i)}_k)^2 + \lambda \sum_{j \neq k} |\theta_j| + \lambda |\theta_k|
\]  

(5.6)

\[
= J_{\lambda, \text{lasso}}(\theta_k; \theta_1, \theta_2, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_m)
\]  

(5.7)
Coordinate Descent

- Isolate one coordinate $k$; fix the remainder $j \neq k$

\[
\sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \|\theta\|_1 = \sum_{i=1}^{n} \left( y^{(i)} - \sum_{j=1}^{m} \theta_j x_j^{(i)} \right)^2 + \lambda \|\theta\|_1
\]

(5.4)

\[
= \sum_{i=1}^{n} \left( y^{(i)} - \theta_k x_k^{(i)} - \sum_{j \neq k} \theta_j x_j^{(i)} \right)^2 + \lambda \sum_{j \neq k} |\theta_j| + \lambda |\theta_k|
\]

(5.5)

\[
= \sum_{i=1}^{n} (r_k^{(i)} - \theta_k x_k^{(i)})^2 + \lambda \sum_{j \neq k} |\theta_j| + \lambda |\theta_k|
\]

(5.6)

\[
= J_{\lambda,\text{lasso}}(\theta_k; \theta_1, \theta_2, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_m)
\]

(5.7)

- We then optimize only $\theta_k$ holding the remainder fixed, and then choose a different $k'$ for the next round.
### Coordinate Descent

- General algorithm, one round of coordinate descent to be repeated.

**Algorithm 2: Coordinate Descent for Lasso Solution**

1. \( \theta \leftarrow 0 \)
2. **for** \( k = 1, \cdots, m \) **do**
3. \( r_k^{(i)} \leftarrow y(i) - \sum_{j \neq k} x_j^{(i)} \theta_j \)
4. \( \theta_k \leftarrow \arg\min_{\theta_k} \sum_{i=1}^{n} (r_k^{(i)} - x_k^{(i)} \theta_k)^2 + \lambda |\theta_k| \)

- We repeat the above several times until convergence.
- This doesn’t help lack of differentiability. How do we perform the minimization?
Convexity, Subgradients, and Subdifferential

- **Convexity**: A general function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^m$ and $\lambda \in [0, 1]$:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \quad (5.8)$$
Convexity, Subgradients, and Subdifferential

- Convexity: A general function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^m$ and $\lambda \in [0, 1]$:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$  \hspace{1cm} (5.8)

- Convex functions have no local minima other than the global minima
Convexity, Subgradients, and Subdifferential

- **Convexity:** A general function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^m$ and $\lambda \in [0, 1]$:
  \[
  f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)
  \] (5.8)

- Convex functions have no local minima other than the global minima.

- A vector $g \in \mathbb{R}^m$ is a **subgradient** of $f$ at $x$ if for all $z$ we have $f(z) \geq f(x) + g^T(z - x)$.

Convex functions have no local minima other than the global minima.
Convexity, Subgradients, and Subdifferential

- Convexity: A general function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^m$ and $\lambda \in [0, 1]$:

  $$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \quad (5.8)$$

- Convex functions have no local minima other than the global minima.
- A vector $g \in \mathbb{R}^m$ is a subgradient of $f$ at $x$ if for all $z$ we have $f(z) \geq f(x) + g^T(z - x)$. Valid when $f$ is differentiable (e.g., at $x_1$) and when $f$ is not differentiable (e.g., at $x_2$).
Convexity, Subgradients, and Subdifferential

- **Convexity**: A general function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^m$ and $\lambda \in [0, 1]$:
  \[ f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \]  
  \[ (5.8) \]

- Convex functions have no local minima other than the global minima.
- A vector $g \in \mathbb{R}^m$ is a subgradient of $f$ at $x$ if for all $z$ we have $f(z) \geq f(x) + g^T(z - x)$. Valid when $f$ is differentiable (e.g., at $x_1$) and when $f$ is not differentiable (e.g., at $x_2$).
- Set of all subgradients of $f$ at $x$ known as the subdifferential at $x$
  \[ \partial f(x) = \{ g \in \mathbb{R}^m : \forall z, f(z) \geq f(x) + g^T(z - x) \} \]  
  \[ (5.9) \]
Convexity, Subgradients, and Subdifferential

- Convexity: A general function \( f : \mathbb{R}^m \rightarrow \mathbb{R} \) is convex if for all \( x, y \in \mathbb{R}^m \) and \( \lambda \in [0, 1] \):

\[
    f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)
\]  

(5.8)

- Convex functions have no local minima other than the global minima.

A vector \( g \in \mathbb{R}^m \) is a subgradient of \( f \) at \( x \) if for all \( z \) we have \( f(z) \geq f(x) + g^\top (z - x) \). Valid when \( f \) is differentiable (e.g., at \( x_1 \)) and when \( f \) is not differentiable (e.g., at \( x_2 \)).

- Set of all subgradients of \( f \) at \( x \) known as the subdifferential at \( x \)

\[
    \partial f(x) = \{ g \in \mathbb{R}^m : \forall z, f(z) \geq f(x) + g^\top (z - x) \}
\]  

(5.9)

- If \( f \) is differentiable at \( x \) then \( \partial f(x) = \{ \nabla f(x) \} \).
Convexity, Subgradients, and Lasso

Fact: a convex function is minimized at $x^*$ if the vector 0 is a subgradient at $x^*$. In other words,

$$0 \in \partial f(x^*) \iff f(x^*) \leq f(x) \forall x$$  \hspace{1cm} (5.10)
Fact: a convex function is minimized at $x^*$ if the vector 0 is a subgradient at $x^*$. In other words,

$$0 \in \partial f(x^*) \text{ iff } f(x^*) \leq f(x) \forall x$$

(5.10)

That is, a minimum means zero is a member of the subdifferential.
Fact: a convex function is minimized at $x^*$ if the vector 0 is a subgradient at $x^*$. In other words,

$$0 \in \partial f(x^*) \iff f(x^*) \leq f(x) \forall x$$

That is, a minimum means zero is a member of the subdifferential.

When $f$ is everywhere differentiable, this is same as finding $x^*$ that satisfies $\nabla_x f(x^*) = 0$. 

Convexity, Subgradients, and Lasso
Fact: a convex function is minimized at $x^*$ if the vector 0 is a subgradient at $x^*$. In other words,

$$0 \in \partial f(x^*) \text{ iff } f(x^*) \leq f(x) \forall x$$

(5.10)

That is, a minimum means zero is a member of the subdifferential.

When $f$ is everywhere differentiable, this is same as finding $x^*$ that satisfies $\nabla_x f(x^*) = 0$.

Fact: The lasso coordinate objective $\sum_{i=1}^{n} (r_k^{(i)} - x_k^{(i)} \theta_k)^2 + \lambda |\theta_k|$ is convex in the parameter $\theta_k$. 
Convexity, Subgradients, and Lasso

- Fact: a convex function is minimized at $x^*$ if the vector 0 is a subgradient at $x^*$. In other words,

$$0 \in \partial f(x^*) \iff f(x^*) \leq f(x) \forall x$$  \hspace{1cm} (5.10)

- That is, a minimum means zero is a member of the subdifferential.

- When $f$ is everywhere differentiable, this is same as finding $x^*$ that satisfies $\nabla_x f(x^*) = 0$.

- Fact: The lasso coordinate objective $\sum_{i=1}^n (r_k^{(i)} - x_k^{(i)} \theta_k)^2 + \lambda \lvert \theta_k \rvert$ is convex in the parameter $\theta_k$.

- Hence, we find a value of $\theta_k$ so that zero is a subgradient (i.e., is a member of the subdifferential).
Fact: a convex function is minimized at $x^*$ if the vector 0 is a subgradient at $x^*$. In other words,

$$0 \in \partial f(x^*) \iff f(x^*) \leq f(x) \forall x$$

(5.10)

That is, a minimum means zero is a member of the subdifferential.

When $f$ is everywhere differentiable, this is same as finding $x^*$ that satisfies $\nabla_x f(x^*) = 0$.

Fact: The lasso coordinate objective $\sum_{i=1}^{n} (r^{(i)}_k - x^{(i)}_k \theta_k)^2 + \lambda |\theta_k|$ is convex in the parameter $\theta_k$.

Hence, we find a value of $\theta_k$ so that zero is a subgradient (i.e., is a member of the subdifferential).

With L1 objective, things are thus almost as easy as with L2.
Define $a_k = 2 \sum_{i=1}^{n} (x_k^{(i)})^2$ and $c_k = 2 \sum_{i=1}^{n} r_k^{(i)} x_k^{(i)}$. Note $a_k > 0$. Find $\theta_k$ making the following contain zero:
Lasso’s Subdifferential

- Define $a_k = 2 \sum_{i=1}^{n} (x_k^{(i)})^2$ and $c_k = 2 \sum_{i=1}^{n} r_k^{(i)} x_k^{(i)}$. Note $a_k > 0$.

Find $\theta_k$ making the following contain zero:

$$
\partial f = \partial \left( \sum_{i=1}^{n} (r_k^{(i)} - x_k^{(i)} \theta_k)^2 + \lambda |\theta_k| \right)
$$

$$
= \begin{cases}
    \{ a_k \theta_k - c_k - \lambda \} & \text{if } \theta_k < 0 \\
    [-c_k - \lambda, -c_k + \lambda] & \text{if } \theta_k = 0 \\
    \{ a_k \theta_k - c_k + \lambda \} & \text{if } \theta_k > 0
\end{cases}
$$
Lasso’s Subdifferential

- Define $a_k = 2 \sum_{i=1}^{n} (x_k^{(i)})^2$ and $c_k = 2 \sum_{i=1}^{n} r_k^{(i)} x_k^{(i)}$. Note $a_k > 0$.

- Find $\theta_k$ making the following contain zero:

$$\partial f = \partial \left( \sum_{i=1}^{n} (r_k^{(i)} - x_k^{(i)} \theta_k)^2 + \lambda |\theta_k| \right)$$

$$= \begin{cases} 
    \{ a_k \theta_k - c_k - \lambda \} & \text{if } \theta_k < 0 \\
    [-c_k - \lambda, -c_k + \lambda] & \text{if } \theta_k = 0 \\
    \{ a_k \theta_k - c_k + \lambda \} & \text{if } \theta_k > 0
\end{cases}$$

- The value of $\theta_k$ such that $0 \in \partial f$ depends on $c_k$, three cases $c_k < -\lambda$, $|c_k| \leq \lambda$, and $c_k > \lambda$ on right.
Lasso’s Subdifferential

- Define $a_k = 2 \sum_{i=1}^{n} (x_k^{(i)})^2$ and $c_k = 2 \sum_{i=1}^{n} r_k^{(i)} x_k^{(i)}$. Note $a_k > 0$.

  Find $\theta_k$ making the following contain zero:

  $$\partial_f = \partial \left( \sum_{i=1}^{n} (r_k^{(i)} - x_k^{(i)} \theta_k)^2 + \lambda |\theta_k| \right)$$

  $$= \begin{cases} 
  \{ a_k \theta_k - c_k - \lambda \} & \text{if } \theta_k < 0 \\
  [-c_k - \lambda, -c_k + \lambda] & \text{if } \theta_k = 0 \\
  \{ a_k \theta_k - c_k + \lambda \} & \text{if } \theta_k > 0 
  \end{cases}$$

- The value of $\theta_k$ such that $0 \in \partial_f$ depends on $c_k$, three cases $c_k < -\lambda$, $|c_k| \leq \lambda$, and $c_k > \lambda$ on right.

- If $c_k < -\lambda$, or $c_k > \lambda$, differentiable at point when $0 \in \partial_f = \{ \nabla f \}$, analytic solution giving minimum, (pink/orange line zero, x-axis).
Lasso’s Subdifferential

- Define $a_k = 2 \sum_{i=1}^{n} (x_k^{(i)})^2$ and $c_k = 2 \sum_{i=1}^{n} r_k^{(i)} x_k^{(i)}$. Note $a_k > 0$.
- Find $\theta_k$ making the following contain zero:

$$\partial f = \partial \left( \sum_{i=1}^{n} (r_k^{(i)} - x_k^{(i)} \theta_k)^2 + \lambda |\theta_k| \right)$$

$$= \begin{cases} 
    \{a_k \theta_k - c_k - \lambda\} & \text{if } \theta_k < 0 \\
    [-c_k - \lambda, -c_k + \lambda] & \text{if } \theta_k = 0 \\
    \{a_k \theta_k - c_k + \lambda\} & \text{if } \theta_k > 0 
\end{cases}$$

- The value of $\theta_k$ such that $0 \in \partial f$ depends on $c_k$, three cases $c_k < -\lambda$, $|c_k| \leq \lambda$, and $c_k > \lambda$ on right.
- If $c_k < -\lambda$, or $c_k > \lambda$, differentiable at point when $0 \in \partial f = \{\nabla f\}$, analytic solution giving minimum, (pink/orange line zero, x-axis).
- If $|c_k| \leq \lambda$, $0 \in [-c_k - \lambda, -c_k + \lambda]$, hence $\theta_k = 0$ is the minimum.
This gives us the following solution (note again $a_k > 0$):

$$
\theta_k = \begin{cases} 
\frac{(c_k + \lambda)}{a_k} & \text{if } c_k < -\lambda \quad (\text{making } \theta_k < 0) \\
0 & \text{if } |c_k| \leq \lambda \quad (\text{making } \theta_k = 0) \\
\frac{(c_k - \lambda)}{a_k} & \text{if } c_k > \lambda \quad (\text{making } \theta_k > 0)
\end{cases}
$$

$$
\theta_k = \text{sign}(c_k) \max(0, |c_k| - \lambda)/a_k
$$

(5.11)

(5.12)
This gives us the following solution (note again $a_k > 0$):

$$
\theta_k = \begin{cases} 
(c_k + \lambda)/a_k & \text{if } c_k < -\lambda \quad (\text{making } \theta_k < 0) \\
0 & \text{if } |c_k| \leq \lambda \quad (\text{making } \theta_k = 0) \\
(c_k - \lambda)/a_k & \text{if } c_k > \lambda \quad (\text{making } \theta_k > 0)
\end{cases}
$$

$$
= \text{sign}(c_k) \max(0, |c_k| - \lambda)/a_k 
$$

Note that without any Lasso term, the solution for $\theta_k$ would be

$$
\theta_k = c_k/a_k = \text{sign}(c_k)|c_k|/a_k = \text{sign}(c_k) \max(0, |c_k|)/a_k.
$$
This gives us the following solution (note again $a_k > 0$):

$$
\theta_k = \begin{cases} 
(c_k + \lambda)/a_k & \text{if } c_k < -\lambda \quad (\text{making } \theta_k < 0) \\
0 & \text{if } |c_k| \leq \lambda \quad (\text{making } \theta_k = 0) \\
(c_k - \lambda)/a_k & \text{if } c_k > \lambda \quad (\text{making } \theta_k > 0)
\end{cases}
$$

$$
\theta_k = \text{sign}(c_k) \max(0, |c_k| - \lambda)/a_k
$$

Note that without any Lasso term, the solution for $\theta_k$ would be

$$
\theta_k = c_k/a_k = \text{sign}(c_k)|c_k|/a_k = \text{sign}(c_k) \max(0, |c_k|)/a_k.
$$

The above has a nice intuitive: we solve as normal (using gradients, $c_k/a_k$, and $\lambda$) as long as $c_k$ is not too small.
Soft Thresholding

This gives us the following solution (note again $a_k > 0$):

$$\theta_k = \begin{cases} 
\frac{(c_k + \lambda_k)}{a_k} & \text{if } c_k < -\lambda \quad \text{(making } \theta_k < 0) \\
0 & \text{if } |c_k| \leq \lambda \quad \text{(making } \theta_k = 0) \\
\frac{(c_k - \lambda_k)}{a_k} & \text{if } c_k > \lambda \quad \text{(making } \theta_k > 0) 
\end{cases}$$

$$= \text{sign}(c_k) \max(0, |c_k| - \lambda)/a_k \quad (5.11)$$

Note that without any Lasso term, the solution for $\theta_k$ would be

$$\theta_k = c_k/a_k = \text{sign}(c_k)|c_k|/a_k = \text{sign}(c_k) \max(0, |c_k|)/a_k.$$

The above has a nice intuitive: we solve as normal (using gradients, $c_k/a_k$, and $\lambda$) as long as $c_k$ is not too small.

If $c_k$ does get to small (in the middle term), then we just truncate $\theta_k$ to zero, and when we truncate depends on $\lambda$ and $c_k$. 

Simple solution: $X$ has orthonormal columns

Let $\tilde{\theta}$ be the linear least squares solution. When the design matrix $X$ has orthonormal columns (a very special case), then solutions are simple modifications of $\tilde{\theta}$.
Simple solution: $X$ has orthonormal columns

- Let $\tilde{\theta}$ be the linear least squares solution. When the design matrix $X$ has orthonormal columns (a very special case), then solutions are simple modifications of $\tilde{\theta}$.
- Ridge solution becomes $\tilde{\theta}_j/(1 + \lambda)$. shrinkage.
Simple solution: \( X \) has orthonormal columns

- Let \( \tilde{\theta} \) be the linear least squares solution. When the design matrix \( X \) has orthonormal columns (a very special case), then solutions are simple modifications of \( \tilde{\theta} \).
- Ridge solution becomes \( \tilde{\theta}_j / (1 + \lambda) \). \( \text{shrinkage} \).
- Lasso solution becomes \( \text{sign}(\tilde{\theta}_j) \max(0, (|\tilde{\theta}_j| - \lambda)) \). \( \text{soft thresholding} \).
Simple solution: $X$ has orthonormal columns

- Let $\hat{\theta}$ be the linear least squares solution. When the design matrix $X$ has orthonormal columns (a very special case), then solutions are simple modifications of $\hat{\theta}$.

- Ridge solution becomes $\tilde{\theta}_j / (1 + \lambda)$. shrinkage.

- Lasso solution becomes $\text{sign}(\tilde{\theta}_j) \max(0, (|\tilde{\theta}_j| - \lambda))$. soft thresholding

- Best subset of size $k$: $\tilde{\theta}_{\sigma_j} 1\{j \leq k\}$ where $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ is a permutation of $\{1, 2, \ldots, m\}$ such that $\tilde{\theta}_{\sigma_1} \geq \tilde{\theta}_{\sigma_2} \geq \cdots \geq \tilde{\theta}_{\sigma_m}$. hard thresholding
Shrinkage relative to linear regression

Ridge

Lasso

Best Subset

(0,0)

(0,0)

(0,0)
Other regularizers

\[ \tilde{\theta} \in \arg \min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda R(\theta) \]  (5.13)
Other regularizers

- Other regularizers

\[ \tilde{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda R(\theta) \]  

(5.13)

- We’ve seen ridge \( R(\theta) = ||\theta||_2^2 \) and lasso \( R(\theta) = ||\theta||_1 \).
Other regularizers

Other regularizers

\[ \tilde{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda R(\theta) \]

We’ve seen ridge \( R(\theta) = \|\theta\|_2^2 \) and lasso \( R(\theta) = \|\theta\|_1 \).

\( p\)-"norm" regularizers \( R(\theta) = \|\theta\|_p^p \) for \( 0 < p < 1 \), not a norm.
Other regularizers

- Other regularizers

\[
\hat{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda R(\theta)
\]  

(5.13)

- We’ve seen ridge \( R(\theta) = \|\theta\|_2^2 \) and lasso \( R(\theta) = \|\theta\|_1 \).
- \( p-"\text{norm}"\) regularizers \( R(\theta) = \|\theta\|_p^p \) for \( 0 < p < 1 \), not a norm.
- Mixed norms, **elastic-net** \( R(\theta) = \alpha \|\theta\|_1 + (1 - \alpha) \|\theta\|_2 \) for \( 0 \leq \alpha \leq 1 \).
Other regularizers

- Other regularizers

\[ \tilde{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda R(\theta) \quad (5.13) \]

- We’ve seen ridge \( R(\theta) = \|\theta\|_2^2 \) and lasso \( R(\theta) = \|\theta\|_1 \).
- \( p\)-”norm” regularizers \( R(\theta) = \|\theta\|_p^p \) for \( 0 < p < 1 \), not a norm.
- Mixed norms, elastic-net \( R(\theta) = \alpha \|\theta\|_1 + (1 - \alpha) \|\theta\|_2 \) for \( 0 \leq \alpha \leq 1 \).
- Grouped Lasso

\[ \tilde{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \sum_{j=1}^{g} \|\theta_{A_g}\|_2 \quad (5.14) \]

where \( A_1, A_2, \ldots, A_g \) set of groups of feature indices, and the regularizer is sum of (not-squared) 2-norms over groups.
Other regularizers

- We’ve seen ridge $R(\theta) = \|\theta\|_2^2$ and lasso $R(\theta) = \|\theta\|_1$.
- $p$-"norm" regularizers $R(\theta) = \|\theta\|_p^p$ for $0 < p < 1$, not a norm.
- Mixed norms, elastic-net $R(\theta) = \alpha\|\theta\|_1 + (1 - \alpha)\|\theta\|_2$ for $0 \leq \alpha \leq 1$.
- Grouped Lasso

$$\tilde{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \sum_{j=1}^{g} \|\theta_{A_g}\|_2$$

where $A_1, A_2, \ldots, A_g$ set of groups of feature indices, and the regularizer is sum of (not-squared) 2-norms over groups.
- Total Variation: $R(\theta) = \sum_{j=1}^{m-1} |\theta_j - \theta_{j+1}|$. 
Other regularizers

\[ \tilde{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \mathcal{R}(\theta) \]  

(5.13)

- We’ve seen ridge \( R(\theta) = \|\theta\|_2^2 \) and lasso \( R(\theta) = \|\theta\|_1 \).
- \( p-”norm” \) regularizers \( R(\theta) = \|\theta\|_p^p \) for \( 0 < p < 1 \), not a norm.
- Mixed norms, elastic-net \( R(\theta) = \alpha \|\theta\|_1 + (1 - \alpha) \|\theta\|_2 \) for \( 0 \leq \alpha \leq 1 \).
- Grouped Lasso

\[ \tilde{\theta} \in \arg\min_{\theta} \sum_{i=1}^{n} (y^{(i)} - \theta^T x^{(i)})^2 + \lambda \sum_{j=1}^{g} \|\theta_{A_g}\|_2 \]  

(5.14)

where \( A_1, A_2, \ldots, A_g \) set of groups of feature indices, and the regularizer is sum of (not-squared) 2-norms over groups.

- Total Variation: \( \mathcal{R}(\theta) = \sum_{j=1}^{m-1} |\theta_j - \theta_{j+1}| \).

- Structured regularizers, contours can be other convex polyhedra (besides just octahedron in 3D)
As the dimensions grow

- $d^{th}$ order linear model, with $x \in \mathbb{R}^m$

$$f_\theta(x) = \theta_0 + \sum_{i=1}^{m} \theta_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{m} \theta_{i,j} x_i x_j + \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{m} \theta_{i,j,k} x_i x_j x_k$$

$$+ \cdots + \sum_{i_1 \leq 1}^{m} \sum_{i_2 \leq 1}^{m} \cdots \sum_{i_d \leq 1}^{m} \prod_{\ell=1}^{d} x_{i_\ell}$$

number of parameters to learn grows as $O(m^d)$ exponentially in the order.
Geometric Tessellation. Inner and Surface Volume

- Tessellation regions grow exponentially with dimension

\[ r=5, m = 1 \]
\[ r=4, m = 2 \]
\[ r=3, m = 3 \]

With \( r \) distinct values (edge length \( r \)), we have \( r^m \) combinations in \( m \) dimensions. Surface volume is \( 2^m r^m \) while inner volume is \( r^m \). Ratio of surface area to volume is \( \frac{2^m r^m}{r^m} = 2^m \frac{r}{r} \).

A lot more inner substance, relative to surface, as \( r \) grows. \( 2^m \frac{r}{r} \to 0 \) as \( r \to 1 \).

A lot more surface, relative to inner volume, as \( m \) grows. \( 2^m \frac{r}{r} \to 1 \) as \( m \to 1 \).
Tessellation regions grow exponentially with dimension.

With $r$ distinct values (edge length $r$), we have $r^m$ combinations in $m$ dimensions.
Geometric Tessellation. Inner and Surface Volume

- Tessellation regions grow exponentially with dimension

- With $r$ distinct values (edge length $r$), we have $r^m$ combinations in $m$ dimensions.

- Surface volume is $2mr^{m-1}$ while inner volume is $r^m$. Ratio of surface area to volume is $2mr^{m-1}/r^m = 2m/r$
Geometric Tessellation. Inner and Surface Volume

- Tessellation regions grow exponentially with dimension.

- With \( r \) distinct values (edge length \( r \)), we have \( r^m \) combinations in \( m \) dimensions.

- Surface volume is \( 2mr^{m-1} \) while inner volume is \( r^m \). Ratio of surface area to volume is \( 2mr^{m-1}/r^m = 2m/r \)

- A lot more inner substance, relative to surface, as \( r \) grows. \( 2m/r \to 0 \) as \( r \to \infty \).
Geometric Tessellation. Inner and Surface Volume

- Tessellation regions grow exponentially with dimension.

- With $r$ distinct values (edge length $r$), we have $r^m$ combinations in $m$ dimensions.

- Surface volume is $2mr^{m-1}$ while inner volume is $r^m$. Ratio of surface area to volume is $2mr^{m-1}/r^m = 2m/r$.

- A lot more inner substance, relative to surface, as $r$ grows. $2m/r \to 0$ as $r \to \infty$.

- A lot more surface, relative to inner volume, as $m$ grows. $2m/r \to \infty$ as $m \to \infty$. 
Data in high dimensions

From Hastie et. al, 2009

- Suppose data $\mathcal{D} = \{x^{(i)}\}_{i=1}^n$ is distributed uniformly at random within $m$-dimensional unit box so $x^{(i)} \in [0, 1]^m$ (thus, $0 \leq x_j^{(i)} \leq 1$).
Data in high dimensions

From Hastie et. al, 2009

- Suppose data \( \mathcal{D} = \{x^{(i)}\}_{i=1}^n \) is distributed uniformly at random within \( m \)-dimensional unit box so \( x^{(i)} \in [0, 1]^m \) (thus, \( 0 \leq x_j^{(i)} \leq 1 \)).
- If we take an arbitrary data point and a fraction \( 0 < r < 1 \) of the closest data points, will carve out \( m \)-dimensional volume, with expected edge length \( r^{1/m} \), so \( e_m(r) = r^{1/m} \).
Data in high dimensions

From Hastie et. al, 2009

- Suppose data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{n}$ is distributed uniformly at random within $m$-dimensional unit box so $x^{(i)} \in [0, 1]^m$ (thus, $0 \leq x_j^{(i)} \leq 1$).

- If we take an arbitrary data point and a fraction $0 < r < 1$ of the closest data points, will carve out $m$-dimensional volume, with expected edge length $r^{1/m}$, so $e_m(r) = r^{1/m}$.

- One percent of the data in ten dimensions has expected edge length $e_{10}(0.01) = 0.63$. Ten percent of the data: $e_{10}(0.1) = 0.8$. 

![Image of high-dimensional data distribution]
Data in high dimensions

From Hastie et. al, 2009

• Suppose data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{n}$ is distributed uniformly at random within $m$-dimensional unit box so $x^{(i)} \in [0, 1]^m$ (thus, $0 \leq x_j^{(i)} \leq 1$).

• If we take an arbitrary data point and a fraction $0 < r < 1$ of the closest data points, will carve out $m$-dimensional volume, with expected edge length $r^{1/m}$, so $e_m(r) = r^{1/m}$.

• One percent of the data in ten dimensions has expected edge length $e_{10}(0.01) = 0.63$. Ten percent of the data: $e_{10}(0.1) = 0.8$. 0.01 percent of the data in 100 dimensions, $e_{100}(0.0001) = 0.91$. 
Data in high dimensions

From Hastie et. al, 2009

- Suppose data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{n}$ is distributed uniformly at random within $m$-dimensional unit box so $x^{(i)} \in [0, 1]^m$ (thus, $0 \leq x_j^{(i)} \leq 1$).

- If we take an arbitrary data point and a fraction $0 < r < 1$ of the closest data points, will carve out $m$-dimensional volume, with expected edge length $r^{1/m}$, so $e_m(r) = r^{1/m}$.

- One percent of the data in ten dimensions has expected edge length $e_{10}(0.01) = 0.63$. Ten percent of the data: $e_{10}(0.1) = 0.8$. 0.01 percent of the data in 100 dimensions, $e_{100}(0.0001) = 0.91$.

- Hence, since max edge length is 1, small fraction $r$ of data takes up most of the edge length range.
Data in high dimensions

From Hastie et. al, 2009

- Suppose data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{n}$ is distributed uniformly at random within $m$-dimensional unit box so $x^{(i)} \in [0, 1]^m$ (thus, $0 \leq x_j^{(i)} \leq 1$).
- If we take an arbitrary data point and a fraction $0 < r < 1$ of the closest data points, will carve out $m$-dimensional volume, with expected edge length $r^{1/m}$, so $e_m(r) = r^{1/m}$.
- One percent of the data in ten dimensions has expected edge length $e_{10}(0.01) = 0.63$. Ten percent of the data: $e_{10}(0.1) = 0.8$. 0.01 percent of the data in 100 dimensions, $e_{100}(0.0001) = 0.91$.
- Hence, since max edge length is 1, small fraction $r$ of data takes up most of the edge length range.
- In fact, $\lim_{m \to \infty} e_m(\epsilon) = 1$ for any $\epsilon > 0$. 

\[ e_m(r) = r^{1/m} \]

\[ e_{10}(0.01) = 0.63 \]

\[ e_{10}(0.1) = 0.8 \]

\[ e_{100}(0.0001) = 0.91 \]
Typical McIntosh apple (meaning the fruit, not the phone), 2.75 inches in diameter (radius $r = 3.5$ cm), $.3$ mm ($\epsilon = .03$ cm) thin peel. Sphere volume $\frac{4}{3}\pi r^3$, apple is about $180$ cm$^3$, while peel volume is $\frac{4}{3}\pi r^3 - \frac{4}{3}\pi (r - \epsilon)^3 \approx 4.5$ cm$^3$, relative volume about $4.5/180 = 2.5\%$. 
Typical McIntosh apple (meaning the fruit, not the phone), 2.75 inches in diameter (radius $r = 3.5\text{cm}$), .3mm ($\epsilon = .03\text{cm}$) thin peel. Sphere volume $\frac{4}{3}\pi r^3$, apple is about $180\ \text{cm}^3$, while peel volume is $\frac{4}{3}\pi r^3 - \frac{4}{3}\pi (r - \epsilon)^3 \approx 4.5\ \text{cm}^3$, relative volume about $4.5/180 = 2.5\%$.

Volume of sphere in $m$ dimensions $V_m(r) = \bar{V}_m r^m$ for an appropriate $m$-dimensional value (on HW3).
Typical McIntosh apple (meaning the fruit, not the phone), 2.75 inches in diameter (radius $r = 3.5 \text{cm}$), .3mm ($\epsilon = .03 \text{cm}$) thin peel. Sphere volume $\frac{4}{3}\pi r^3$, apple is about 180 cm$^3$, while peel volume is $\frac{4}{3}\pi r^3 - \frac{4}{3}\pi (r - \epsilon)^3 \approx 4.5 \text{ cm}^3$, relative volume about $4.5/180 = 2.5\%$.

Volume of sphere in $m$ dimensions $V_m(r) = \bar{V}_m r^m$ for an appropriate $m$-dimensional value (on HW3).

Fraction of radius 1 apple that is peel (with peel thickness $\epsilon$) is

$$\frac{V_m(1) - V_m(1 - \epsilon)}{V_m(1)} = 1 - (1 - \epsilon)^m \approx 0.5\% \quad \text{when } m = 3. \quad (5.16)$$
Typical McIntosh apple (meaning the fruit, not the phone), 2.75 inches in diameter (radius $r = 3.5\, \text{cm}$), .3mm ($\epsilon = .03\, \text{cm}$) thin peel. Sphere volume $\frac{4}{3}\pi r^3$, apple is about 180 cm$^3$, while peel volume is $\frac{4}{3}\pi r^3 - \frac{4}{3}\pi (r - \epsilon)^3 \approx 4.5 \, \text{cm}^3$, relative volume about $4.5/180 = 2.5\%$.

Volume of sphere in $m$ dimensions $V_m(r) = \bar{V}_m r^m$ for an appropriate $m$-dimensional value (on HW3).

Fraction of radius 1 apple that is peel (with peel thickness $\epsilon$) is

$$\frac{V_m(1) - V_m(1 - \epsilon)}{V_m(1)} = 1 - (1 - \epsilon)^m$$

(5.16)

As dimensionality $m$ gets larger, apple becomes all peel and no core.
Uniformly at random unit vectors on unit sphere

- Let $S_{m-1}(r)$ indicate surface “area” of $m$-dimensional sphere radius $r$. 
Uniformly at random unit vectors on unit sphere

- Let $S_{m-1}(r)$ indicate surface “area” of $m$-dimensional sphere radius $r$.
- Define uniform distribution $U_{S_{m-1}(1)}$ on $S_{m-1}(1)$ and independently draw two random vectors $x, y \sim U_{S_{m-1}(1)}$, hence $\|x\|_2 = \|y\|_2 = 1$.

It can be shown that if $x, y$ are independent random vectors uniformly distributed on the $m$-dimensional sphere, then:

$$
\Pr(\|x \cdot y\| < \varepsilon) > \frac{1}{2} e^{-m\varepsilon^2/2}
$$

which means that uniformly at random vectors in high dimensional space are almost always nearly orthogonal!

One of the reasons why high dimensional random projections preserve information — if two random high dimensional vectors are almost orthogonal, then projections onto them will also be almost orthogonal.
Let \( S_{m-1}(r) \) indicate surface “area” of \( m \)-dimensional sphere radius \( r \).

Define uniform distribution \( U_{S_{m-1}(1)} \) on \( S_{m-1}(1) \) and independently draw two random vectors \( x, y \sim U_{S_{m-1}(1)} \), hence \( \|x\|_2 = \|y\|_2 = 1 \)

Two vectors are orthogonal if \( \langle x, y \rangle = 0 \) and are nearly so if \( |\langle x, y \rangle| < \epsilon \) for small \( \epsilon \).
Let $S_{m-1}(r)$ indicate surface “area” of $m$-dimensional sphere radius $r$.

Define uniform distribution $U_{S_{m-1}(1)}$ on $S_{m-1}(1)$ and independently draw two random vectors $x, y \sim U_{S_{m-1}(1)}$, hence $\|x\|_2 = \|y\|_2 = 1$

Two vectors are orthogonal if $\langle x, y \rangle = 0$ and are nearly so if $|\langle x, y \rangle| < \epsilon$ for small $\epsilon$.

It can be shown that if $x, y$ are independent random vectors uniformly distributed on the $m$-dimensional sphere, then:

$$\Pr(|\langle x, y \rangle| < \epsilon) > 1 - e^{-m\epsilon^2/2}$$

which means that uniformly at random vectors in high dimensional space are almost always nearly orthogonal!
Uniformly at random unit vectors on unit sphere

- Let $S_{m-1}(r)$ indicate surface “area” of $m$-dimensional sphere radius $r$.
- Define uniform distribution $U_{S_{m-1}(1)}$ on $S_{m-1}(1)$ and independently draw two random vectors $x, y \sim U_{S_{m-1}(1)}$, hence $\|x\|_2 = \|y\|_2 = 1$.
- Two vectors are orthogonal if $\langle x, y \rangle = 0$ and are nearly so if $|\langle x, y \rangle| < \epsilon$ for small $\epsilon$.
- It can be shown that if $x, y$ are independent random vectors uniformly distributed on the $m$-dimensional sphere, then:

$$\Pr(|\langle x, y \rangle| < \epsilon) > 1 - e^{-m\epsilon^2/2}$$

(5.17)

which means that uniformly at random vectors in high dimensional space are almost always nearly orthogonal!

- One of the reasons why high dimensional random projections preserve information — if two random high dimensional vectors are almost orthogonal, then projections onto them will also be almost orthogonal.
Consider a cube in $m$-dimensions, $[0, 1]^m$ having unit volume. The largest sphere contained by this cube has diameter 1 (the inner circle to the right). The smallest sphere containing this cube has diameter $\sqrt{m}$ (outer circle).
Consider a cube in $m$-dimensions, $[0, 1]^m$ having unit volume. The largest sphere contained by this cube has diameter 1 (the inner circle to the right). The smallest sphere containing this cube has diameter $\sqrt{m}$ (outer circle).

- Fraction of inner spherical volume (unit sphere radius $1/2$, $V_m(r)$ for radius $r = 1/2$) to unit cube (which has volume 1 for any $m$)

$$\frac{\text{Inner spherical vol.}}{\text{unit cube vol.}} = \frac{V_m(r)}{1} = \frac{\pi^{m/2}}{\Gamma\left(\frac{m}{2} + 1\right)} r^m$$

(5.18)
Inner/Outer Spherical Bounds on Cubic Boxes

Consider a cube in $m$-dimensions, $[0, 1]^m$ having unit volume. The largest sphere contained by this cube has diameter 1 (the inner circle to the right). The smallest sphere containing this cube has diameter $\sqrt{m}$ (outer circle).

Fraction of inner spherical volume (unit sphere radius $1/2$, $V_m(r)$ for radius $r = 1/2$) to unit cube (which has volume 1 for any $m$)

$$\frac{\text{Inner spherical vol.}}{\text{unit cube vol.}} = \frac{V_m(r)}{1} = \frac{\pi^{m/2}}{\Gamma\left(\frac{m}{2} + 1\right)} r^m$$ (5.18)

We have $\lim_{m \to \infty} V_m(1/2)/1 = 0$. Hence, number of volumes of unit spheres that can fit in a cube goes to infinity. More and more unit spheres can be packed into a unit cube as the dimensionality gets higher.
Using 2D to represent High-D as if 2D was High-D

- Relationship between unit-radius \((r = 1)\) sphere and unit volume (side length = 1) cube as the dimensions grow if they were to be also true in 2D.

Illustration of the relationship between the sphere and the cube in 2, 4, and \(m\)-dimensions (from Blum, Hopcroft, & Kannan, 2016).
Using 2D to represent High-D as if 2D was High-D

- Relationship between unit-radius \( r = 1 \) sphere and unit volume (side length = 1) cube as the dimensions grow if they were to be also true in 2D.
- Note that for cube, as \( m \) gets higher, distance from center to face stays at 1/2 but distance from center to vertex grows as \( \sqrt{m}/2 \).

Illustration of the relationship between the sphere and the cube in 2, 4, and \( m \)-dimensions (from Blum, Hopcroft, & Kannan, 2016).
Using 2D to represent High-D as if 2D was High-D

- Relationship between unit-radius \( (r = 1) \) sphere and unit volume (side length = 1) cube as the dimensions grow if they were to be also true in 2D.
- Note that for cube, as \( m \) gets higher, distance from center to face stays at \( 1/2 \) but distance from center to vertex grows as \( \sqrt{m}/2 \).
- For \( m = 4 \) vertex distance is \( \sqrt{m}/2 = 1 \).

Illustration of the relationship between the sphere and the cube in 2, 4, and \( m \)-dimensions (from Blum, Hopcroft, & Kannan, 2016).
The Blessing of Dimensionality

- More dimensions can help to distinguish categories, make pattern recognition easier (and even possible). Recall Voronoi tessellation...
The Blessing of Dimensionality

- More dimensions can help to distinguish categories, make pattern recognition easier (and even possible). Recall Voronoi tessellation.

- Support vector machines (SVMs) can find and exploit data patterns extant only in extremely high (or even infinite) dimensional space!!!
The Blessing of Dimensionality

- More dimensions can help to distinguish categories, make pattern recognition easier (and even possible). Recall Voronoi tessellation.

- Support vector machines (SVMs) can find and exploit data patterns extant only in extremely high (or even infinite) dimensional space!!!

- For still more blessings, see “High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality”, David L. Donoho.
We’ve already seen that feature selection is one strategy to reduce feature dimensionality.
We’ve already seen that **feature selection** is one strategy to reduce feature dimensionality.

Feature selection, each feature is either selected or not, all or nothing.

\[ x_{12} = f(x_1, x_2) \]
Feature Selection vs. Dimensionality Reduction

- We’ve already seen that **feature selection** is one strategy to reduce feature dimensionality.
- Feature selection, each feature is either selected or not, all or nothing.
- Other **dimensionality reduction** strategies take the input $x \in \mathbb{R}^m$ and encode each $x$ into $e(x) \in \mathbb{R}^{m'}$, a lower dimensional space, $m' < m$. 
Feature Selection vs. Dimensionality Reduction

- We’ve already seen that **feature selection** is one strategy to reduce feature dimensionality.
- Feature selection, each feature is either selected or not, all or nothing.
- Other **dimensionality reduction** strategies take the input $x \in \mathbb{R}^m$ and encode each $x$ into $\epsilon(x) \in \mathbb{R}^{m'}$, a lower dimensional space, $m' < m$.

Core advantage: If $U = \{1, 2, \ldots, m\}$, there may be no subset $A \subseteq U$ such that $x_A$ will work well. There might be a combination $\epsilon(x)$ that works well. On the right, simple linear combination $a_1 x_1 + a_2 x_2$ works quite well.
PCA is a linear encoding $\phi(x) = x^T W$ for $m \times m$ matrix $W$. 

Let $X$ be the $n \times m$ design matrix, thus $X^0 = XW$ is the encoded $n \times m$ design matrix. Any linear transformation encoder can be so written. PCA has orthonormal columns $W$ (so $W(:,i)W(:,j) = \delta_{ij}$) ordered decreasing by ability to explain variance in original space. Thus, taking first $m_0 \leq m$ columns of $W$ giving $m \times m_0$ matrix $W_k$, yields $XW_k$ a dimensionality reduction. When $m_0 < m$ then the projection is onto a subspace that maximizes the variance over all possible subspaces of dimension $m_0$. When $m_0 = m$ then we order the columns decreasing by variance explanation (so taking any length $m_0$ prefix of columns gives the answer).
Principle Component Analysis (PCA)

- PCA is a linear encoding $e(x) = x^TW$ for $m \times m$ matrix $W$.
- Let $X$ be the $n \times m$ design matrix, thus $X' = XW$ is the encoded $n \times m$ design matrix.
Principle Component Analysis (PCA)

- PCA is a linear encoding $e(x) = x^T W$ for $m \times m$ matrix $W$.
- Let $X$ be the $n \times m$ design matrix, thus $X' = XW$ is the encoded $n \times m$ design matrix.
- Any linear transformation encoder can be so written. PCA has orthonormal columns $W$ (so $\langle W(:,i), W(:,j) \rangle = 1_{\{i=j\}}$) ordered decreasing by ability to explain variance in original space.
Principle Component Analysis (PCA)

- PCA is a linear encoding $\mathbf{e}(x) = x^\top \mathbf{W}$ for $m \times m$ matrix $\mathbf{W}$.
- Let $\mathbf{X}$ be the $n \times m$ design matrix, thus $\mathbf{X}' = \mathbf{X}\mathbf{W}$ is the encoded $n \times m$ design matrix.
- Any linear transformation encoder can be so written. PCA has orthonormal columns $\mathbf{W}$ (so $\langle \mathbf{W}(:,i), \mathbf{W}(:,j) \rangle = 1_{\{i=j\}}$) ordered decreasing by ability to explain variance in original space.
- Thus, taking first $m' \leq m$ columns of $\mathbf{W}$ giving $m \times m'$ matrix $\mathbf{W}_{m'}$ yields $\mathbf{X}\mathbf{W}_{m'}$ a dimensionality reduction.
PCA is a linear encoding $\mathbf{e}(x) = x^T \mathbf{W}$ for $m \times m$ matrix $\mathbf{W}$.

Let $X$ be the $n \times m$ design matrix, thus $X' = X \mathbf{W}$ is the encoded $n \times m$ design matrix.

Any linear transformation encoder can be so written. PCA has orthonormal columns $\mathbf{W}$ (so $\langle \mathbf{W}( :, i), \mathbf{W}( :, j) \rangle = 1_{\{i=j\}}$) ordered decreasing by ability to explain variance in original space.

Thus, taking first $m' \leq m$ columns of $\mathbf{W}$ giving $m \times m'$ matrix $\mathbf{W}_k$, yields $X \mathbf{W}_k$ a dimensionality reduction.

When $m' < m$ then the projection is onto a subspace that maximizes the variance over all possible subspaces of dimension $m'$. When $m' = m$ then we order the columns decreasing by variance explanation (so taking any length $m'$ prefix of columns gives the answer).
Maximum Variance: 1D projection

- Consider $m' = 1$ to start, vector $w^{(1)} \in \mathbb{R}^m$ is the projection (i.e., $\langle x, w^{(1)} \rangle$ projects to 1D).
Maximum Variance: 1D projection

- Consider $m' = 1$ to start, vector $w^{(1)} \in \mathbb{R}^m$ is the projection (i.e., $\langle x, w^{(1)} \rangle$ projects to 1D).

- Assume unit vectors, $\|w^{(1)}\|_2 = \sqrt{\langle w^{(1)}, w^{(1)} \rangle} = 1$. 

\[
\text{Mean of data } \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)} \text{ so mean of projected data } \langle \bar{x}, w^{(1)} \rangle.
\]

\[
\text{Variance of projected data } \frac{1}{n} \sum_{i=1}^{n} (\langle x^{(i)}, w^{(1)} \rangle - \langle \bar{x}, w^{(1)} \rangle)^2 = w^{(1)} | S w^{(1)} | (5.19)
\]

where 
\[
S = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \bar{x})(x^{(i)} - \bar{x})^{\top} (5.20)
\]
Maximum Variance: 1D projection

- Consider $m' = 1$ to start, vector $w^{(1)} \in \mathbb{R}^m$ is the projection (i.e., $\langle x, w^{(1)} \rangle$ projects to 1D).

- Assume unit vectors, $\|w^{(1)}\|_2 = \sqrt{\langle w^{(1)}, w^{(1)} \rangle} = 1$.

- Mean of data $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}$ so mean of projected data $\langle w^{(1)}, \bar{x} \rangle$. 
Maximum Variance: 1D projection

- Consider $m' = 1$ to start, vector $w^{(1)} \in \mathbb{R}^m$ is the projection (i.e., $\langle x, w^{(1)} \rangle$ projects to 1D).
- Assume unit vectors, $\|w^{(1)}\|_2 = \sqrt{\langle w^{(1)}, w^{(1)} \rangle} = 1$.
- Mean of data $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}$ so mean of projected data $\langle w^{(1)}, \bar{x} \rangle$.
- Variance of projected data

$$\frac{1}{n} \sum_{i=1}^{n} \left( \langle w^{(1)}, x^{(i)} \rangle - \langle w^{(1)}, \bar{x} \rangle \right)^2 = w^{(1)^T} S w^{(1)}$$

where

$$S = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \bar{x})(x^{(i)} - \bar{x})^T$$

(5.19)

(5.20)
Goal: Maximize projected variance \( w^{(1)\,T} S w^{(1)} = \langle w^{(1)}, S w^{(1)} \rangle \) subject to \( \| w^{(1)} \|_2 = 1 \).
Goal: Maximize projected variance $w^{(1)\top}S w^{(1)} = \langle w^{(1)}, Sw^{(1)} \rangle$ subject to $\|w^{(1)}\|_2 = 1$.

Constrained optimization, Lagrange multipliers, yielding stationary point $Sw^{(1)} = \lambda_1 w^{(1)}$, so $w^{(1)}$ is an eigenvector of $S$ with eigenvalue $\lambda_1$. 

Maximum Variance: 1D projection
Maximum Variance: 1D projection

- **Goal:** Maximize projected variance $w^{(1)^T}S w^{(1)} = \langle w^{(1)}, Sw^{(1)} \rangle$ subject to $\|w^{(1)}\|_2 = 1$.

- **Constrained optimization, Lagrange multipliers,** yielding stationary point $Sw^{(1)} = \lambda_1 w^{(1)}$, so $w^{(1)}$ is an eigenvector of $S$ with eigenvalue $\lambda_1$.

- **Pre-multiply by $w^{(1)}$ and use $\|w^{(1)}\|_2 = 1$ yields**

  $$\langle w^{(1)}, Sw^{(1)} \rangle = \lambda_1 = \text{the extreme of the variance.} \quad (5.21)$$
Maximum Variance: 1D projection

- Goal: Maximize projected variance $w^{(1)^T}S w^{(1)} = \langle w^{(1)}, S w^{(1)} \rangle$ subject to $\|w^{(1)}\|_2 = 1$.

- Constrained optimization, Lagrange multipliers, yielding stationary point $S w^{(1)} = \lambda_1 w^{(1)}$, so $w^{(1)}$ is an eigenvector of $S$ with eigenvalue $\lambda_1$.

- Pre-multiply by $w^{(1)}$ and use $\|w^{(1)}\|_2 = 1$ yields

$$\langle w^{(1)}, S w^{(1)} \rangle = \lambda_1$$

(5.21)

- Largest Eigenvalue of $S$ is the greatest variance, and projection direction is achieved using corresponding eigenvector $w^{(1)}$. 
Goal: Maximize projected variance $w^{(1)^T}S w^{(1)} = \langle w^{(1)}, S w^{(1)} \rangle$ subject to $\|w^{(1)}\|_2 = 1$.

Constrained optimization, Lagrange multipliers, yielding stationary point $S w^{(1)} = \lambda_1 w^{(1)}$, so $w^{(1)}$ is an eigenvector of $S$ with eigenvalue $\lambda_1$.

Pre-multiply by $w^{(1)}$ and use $\|w^{(1)}\|_2 = 1$ yields

$$\langle w^{(1)}, S w^{(1)} \rangle = \lambda_1 \tag{5.21}$$

Largest Eigenvalue of $S$ is the greatest variance, and projection direction is achieved using corresponding eigenvector $w^{(1)}$.

This is the first principle component.
Subsequent components

- To produce $k^{th}$ component given first $k - 1$ components, first form

$$X_k = X - \sum_{i=1}^{k-1} (Xw(i))w(i)^T$$

and then proceed with maximize projected variance $\langle w(k), S_kw(k) \rangle$

subject to $\|w(k)\|_2 = 1$ where $S_k$ is computed via $X_k$. 

$$X_k = X - \sum_{i=1}^{k-1} (Xw(i))w(i)^T$$

(5.22)
Subsequent components

To produce $k^{th}$ component given first $k - 1$ components, first form

$$X_k = X - \sum_{i=1}^{k-1} X w^{(i)} w^{(i)\top}$$ (5.22)

and then proceed with maximize projected variance $\langle w^{(k)}, S_k w^{(k)} \rangle$
subject to $\|w^{(k)}\|_2 = 1$ where $S_k$ is computed via $X_k$.

Overall, the collection $\{w^{(i)}, \lambda_i\}$ are the $m$ eigenvector/eigenvalue pairs of the matrix $S$. 

END
PCA: Minimizing reconstruction error $m' < m$

- Let $u_i, i \in [m]$ be a set of orthonormal vectors in $\mathbb{R}^m$. 

$$\text{Let } u_i, i \in [m] \text{ be a set of orthonormal vectors in } \mathbb{R}^m.$$
PCA: Minimizing reconstruction error $m' < m$

- Let $u_i, i \in [m]$ be a set of orthonormal vectors in $\mathbb{R}^m$.
- Basis decomposition: any sample $x^{(i)}$ can be written as $x^{(i)} = \sum_{j=1}^{m} \alpha_{i,j} u_j = \sum_{j=1}^{m} \langle x^{(i)}, u_j \rangle u_j$. 

Suppose we decide to use only $m' < m$ dimensions, so $\tilde{x}^{(i)} = \sum_{j=1}^{m'} \langle x^{(i)}, u_j \rangle u_j$.

Reconstruction error $J_{m^0} = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \tilde{x}^{(i)})^2$ (5.23)

To minimize, can be shown that we should choose $u_i$ to be the eigenvector of $S$ corresponding to the $i$th largest eigenvalue.

Let $W_{m^0}$ be the matrix of column eigenvectors of $S$ sorted decreasing by eigenvalue, and $W(:,1:m^0)$ the $m \times m^0$ matrix of the first $m^0$ columns.

Then $XW_{m^0}$ projects down to the first $m^0$ principle components, and is also widely known as the Karhunen-Loève transform (KL T).
PCA: Minimizing reconstruction error $m' < m$

- Let $u_i, i \in [m]$ be a set of orthonormal vectors in $\mathbb{R}^m$.
- Basis decomposition: any sample $x^{(i)}$ can be written as $x^{(i)} = \sum_{j=1}^{m} \alpha_{i,j} u_j = \sum_{j=1}^{m} \langle x^{(i)}, u_j \rangle u_j$.
- Suppose we decide to use only $m' < m$ dimensions, so $\tilde{x}^{(i)} = \sum_{j=1}^{m'} \langle x^{(i)}, u_j \rangle u_j$. 
PCA: Minimizing reconstruction error $m' < m$

- Let $u_i, i \in [m]$ be a set of orthonormal vectors in $\mathbb{R}^m$.
- Basis decomposition: any sample $x^{(i)}$ can be written as
  \[ x^{(i)} = \sum_{j=1}^{m} \alpha_{i,j} u_j = \sum_{j=1}^{m} \langle x^{(i)}, u_j \rangle u_j. \]
- Suppose we decide to use only $m' < m$ dimensions, so
  \[ \tilde{x}^{(i)} = \sum_{j=1}^{m'} \langle x^{(i)}, u_j \rangle u_j. \]
- Reconstruction error

\[
J_{m'} = \frac{1}{n} \sum_{i=1}^{n} \| x^{(i)} - \tilde{x}^{(i)} \|_2^2 \tag{5.23}
\]
PCA: Minimizing reconstruction error $m' < m$

- Let $u_i, i \in [m]$ be a set of orthonormal vectors in $\mathbb{R}^m$.
- Basis decomposition: any sample $x^{(i)}$ can be written as
  $$x^{(i)} = \sum_{j=1}^{m} \alpha_{i,j} u_j = \sum_{j=1}^{m} \langle x^{(i)}, u_j \rangle u_j.$$  
- Suppose we decide to use only $m' < m$ dimensions, so
  $$\tilde{x}^{(i)} = \sum_{j=1}^{m'} \langle x^{(i)}, u_j \rangle u_j.$$  
- Reconstruction error
  $$J_{m'} = \frac{1}{n} \sum_{i=1}^{n} \| x^{(i)} - \tilde{x}^{(i)} \|_2^2 \tag{5.23}$$

- To minimize, can be shown that we should choose $u_i$ to be the eigenvector of $S$ corresponding to the $i^{th}$ largest eigenvalue.
PCA: Minimizing reconstruction error $m' < m$

- Let $u_i, i \in [m]$ be a set of orthonormal vectors in $\mathbb{R}^m$.
- Basis decomposition: any sample $x^{(i)}$ can be written as
  \[ x^{(i)} = \sum_{j=1}^{m} \alpha_{i,j} u_j = \sum_{j=1}^{m} \langle x^{(i)}, u_j \rangle u_j. \]
- Suppose we decide to use only $m' < m$ dimensions, so
  \[ \tilde{x}^{(i)} = \sum_{j=1}^{m'} \langle x^{(i)}, u_j \rangle u_j. \]
- Reconstruction error
  \[ J_{m'} = \frac{1}{n} \sum_{i=1}^{n} \| x^{(i)} - \tilde{x}^{(i)} \|_2^2 \]
  \[ (5.23) \]
  
  - To minimize, can be shown that we should choose $u_i$ to be the eigenvector of $S$ corresponding to the $i^{th}$ largest eigenvalue.
  - Let $W_{m'}$ be the matrix of column eigenvectors of $S$ sorted decreasing by eigenvalue, and $W(:, 1 : m')$ the $m \times m'$ matrix of the first $m'$ columns.
PCA: Minimizing reconstruction error $m' < m$

- Let $u_i, i \in [m]$ be a set of orthonormal vectors in $\mathbb{R}^m$.
- Basis decomposition: any sample $x^{(i)}$ can be written as
  \[ x^{(i)} = \sum_{j=1}^{m} \alpha_{i,j} u_j = \sum_{j=1}^{m} \langle x^{(i)}, u_j \rangle u_j. \]
- Suppose we decide to use only $m' < m$ dimensions, so
  \[ \tilde{x}^{(i)} = \sum_{j=1}^{m'} \langle x^{(i)}, u_j \rangle u_j. \]
- Reconstruction error
  \[
  J_{m'} = \frac{1}{n} \sum_{i=1}^{n} \|x^{(i)} - \tilde{x}^{(i)}\|^2
  \]  
  (5.23)
- To minimize, can be shown that we should choose $u_i$ to be the eigenvector of $S$ corresponding to the $i^{th}$ largest eigenvalue.
- Let $W_{m'}$ be the matrix of column eigenvectors of $S$ sorted decreasing by eigenvalue, and $W(:,1:m')$ the $m \times m'$ matrix of the first $m'$ columns.
- Then $XW_{m'}$ projects down to the first $m'$ principle components, and is also widely known as the Karhunen-Loève transform (KLT)
Example: 2D and two principle directions
PCA and discriminability

PCA projects down to the directions of greatest variance. Are these also the greatest for classification?
PCA and discriminability

- PCA projects down to the directions of greatest variance. Are these also the greatest for classification?
- For classification, we would want the samples in different classes to be as separate as possible. I.e., make it as easy as possible to discern the difference (i.e., discriminate) between the different classes.
PCA and discriminability

- PCA projects down to the directions of greatest variance. Are these also the greatest for classification?
- For classification, we would want the samples in different classes to be as separate as possible. I.e., make it as easy as possible to discern the difference (i.e., discriminate) between the different classes.
- Any projection that blends together samples from different classes on top of each other would be undesirable.
Example: PCA on 2-class data
Example: Data normalization

Left: original data. Center: mean subtraction and variance normalization in each dimension. Right: decorrelated data (i.e., diagonal covariance).
From Bishop 2006

Projecting to greatest variability leads to poor separation, while exists another 1D projection that retains good separability.
PCA and lack of class separability

From Bishop 2006

Left. 2 dimensions of original data. Right: PCA Projection on 2 dimensions of greatest variability — bad for two reasons: (I) spreads individual classes (e.g., red); and (II) blends distinct classes (blue and green), both of which makes classification no easier.
PCA vs. LDA

From Bishop 2006

**PCA:**
component axes that maximize the variance

**LDA:**
maximizing the component axes for class-separation

PCA vs. Linear Discriminant Analysis (LDA).
Axis parallel projection vs. LDA

From Hastie et. al., 2009

Axis parallel (left) vs. LDA (right).
Linear Discriminant Analysis (LDA), 2 classes

- Consider class-conditional Gaussian data, so \( p(x|y) = \mathcal{N}(x|\mu_y, C_y) \) for mean vectors \( \{\mu_y\}_y \) and covariance matrices \( \{C_y\}_y, x \in \mathbb{R}^m. \)

\[
p(x|y) = \frac{1}{2\pi C_y|^{m/2}} \exp \left( -\frac{1}{2} (x - \mu_y)^T C_y^{-1} (x - \mu_y) \right) \tag{5.24}
\]
Linear Discriminant Analysis (LDA), 2 classes

- Consider class-conditional Gaussian data, so \( p(x|y) = \mathcal{N}(x|\mu_y, C_y) \) for mean vectors \( \{\mu_y\}_y \) and covariance matrices \( \{C_y\}_y, \ x \in \mathbb{R}^m \).

\[
p(x|y) = \frac{1}{\sqrt{2\pi|C_y|^{m/2}}} \exp \left( -\frac{1}{2}(x - \mu_y)^\top C_y^{-1}(x - \mu_y) \right) \quad (5.24)
\]

- Two class case \( y \in \{0, 1\} \) with equal covariances \( C_0 = C_1 = C \) (homoscedasticity property) and priors \( p(y = 0) = p(y = 1) \), consider log posterior odds ratio:

\[
\log \frac{p(y = 1|x)}{p(y = 0|x)} = -\frac{1}{2}(x - \mu_1)^\top C_1^{-1}(x - \mu_1) + \frac{1}{2}(x - \mu_0)^\top C_0^{-1}(x - \mu_0) \\
= \left( C^{-1}\mu_1 - C^{-1}\mu_0 \right)^\top x + \mu_0^\top C^{-1}\mu_0 - \mu_1^\top C^{-1}\mu_1 \\
= \theta^\top x + c \quad (5.25)
\]
Linear Discriminant Analysis (LDA), 2 classes

Consider class-conditional Gaussian data, so $p(x|y) = \mathcal{N}(x|\mu_y, C_y)$ for mean vectors $\{\mu_y\}_y$ and covariance matrices $\{C_y\}_y$, $x \in \mathbb{R}^m$.

$$p(x|y) = \frac{1}{2\pi C_y^{m/2}} \exp\left(-\frac{1}{2}(x - \mu_y)^T C_y^{-1} (x - \mu_y)\right)$$  \hspace{1cm} (5.24)

Two class case $y \in \{0, 1\}$ with equal covariances $C_0 = C_1 = C$ (homoskedasticity property) and priors $p(y = 0) = p(y = 1)$, consider log posterior odds ratio:

$$\log \frac{p(y = 1|x)}{p(y = 0|x)} = -\frac{1}{2}(x - \mu_1)^T C_1^{-1} (x - \mu_1) + \frac{1}{2}(x - \mu_0)^T C_0^{-1} (x - \mu_0)$$  \hspace{1cm} (5.25)

$$= (C^{-1}\mu_1 - C^{-1}\mu_0)^T x + \mu_0^T C^{-1}\mu_0 - \mu_1^T C^{-1}\mu_1$$  \hspace{1cm} (5.26)

$$= \theta^T x + c$$  \hspace{1cm} (5.27)

$\theta$ is a projection ($m \times 1$ matrix, linear transformation) down to the 1 dimension that is sufficient for prediction w/o loss.
Thus, decision boundary between the two classes is linear.
Thus, decision boundary between the two classes is linear.

This is true if we consider the decision boundary between any same-covariance two classes of an $\ell$-class soft-max regression classifier, starting from Gaussians for $p(x|y)$.
Thus, decision boundary between the two classes is linear.

This is true if we consider the decision boundary between any same-covariance two classes of an \( \ell \)-class soft-max regression classifier, starting from Gaussians for \( p(x|y) \).

Given discriminant functions of the form

\[
h_y(x) = -\frac{1}{2} \log |C_y| - \frac{1}{2} (x - \mu_y)^\top C_y^{-1} (x - \mu_y) + \log p(y) \tag{5.28}
\]

all decision boundaries \( \{x : h_i(x) = h_j(x)\} \) are linear between to classes (consider Voronoi tessellation) if \( C_y = C \) for all \( y \), or are otherwise quadratic.
LDA

- Thus, decision boundary between the two classes is linear.
- This is true if we consider the decision boundary between any same-covariance two classes of an $\ell$-class soft-max regression classifier, starting from Gaussians for $p(x|y)$.
- Given discriminant functions of the form

$$
h_y(x) = -\frac{1}{2} \log |C_y| - \frac{1}{2} (x - \mu_y)^\top C_y^{-1} (x - \mu_y) + \log p(y) \quad (5.28)
$$

all decision boundaries $\{x : h_i(x) = h_j(x)\}$ are linear between to classes (consider Voronoi tessellation) if $C_y = C$ for all $y$, or are otherwise quadratic.
- With 2-classes, need only 1-dimension. With 3-classes, need only two-dimensions.
LDA

- Thus, decision boundary between the two classes is linear.
- This is true if we consider the decision boundary between any same-covariance two classes of an $\ell$-class soft-max regression classifier, starting from Gaussians for $p(x|y)$.
- Given discriminant functions of the form

$$h_y(x) = -\frac{1}{2} \log |C_y| - \frac{1}{2} (x - \mu_y)^T C_y^{-1} (x - \mu_y) + \log p(y)$$

(5.28)

all decision boundaries $\{x : h_i(x) = h_j(x)\}$ are linear between to classes (consider Voronoi tessellation) if $C_y = C$ for all $y$, or are otherwise quadratic.

- With 2-classes, need only 1-dimension. With 3-classes, need only two-dimensions.
- In general, $\ell$-classes, need an $\ell - 1$-dimensional subspace on which to project that retains ability to discriminate (moving orthogonal to that subspace leads to constant posterior).
Linear Discriminant Analysis (LDA), \( \ell \) classes

- Within class scatter matrix

\[
S_W = \sum_{c=1}^{\ell} S_c \quad \text{where} \quad S_c = \sum_{i \text{ is class } c} (x^{(i)} - \mu_c)(x^{(i)} - \mu_c)^T
\]  

(5.29)

and \( \mu_c \) is the class \( c \) mean, \( \mu_c = \frac{1}{n_c} \sum_{i \text{ is of class } c} x^{(i)} \)
Linear Discriminant Analysis (LDA), \( \ell \) classes

- Within class scatter matrix

\[
S_W = \sum_{c=1}^{\ell} S_c \quad \text{where} \quad S_c = \sum_{i \text{ is class } c} (x^{(i)} - \mu_c)(x^{(i)} - \mu_c)^T
\]  

(5.29)

and \( \mu_c \) is the class \( c \) mean, \( \mu_c = \frac{1}{n_c} \sum_{i \text{ is of class } c} x^{(i)} \)

- Between class scatter matrix, \( n_c = \text{number of samples of class } c \).

\[
S_B = \sum_{c=1}^{\ell} n_c(\mu_c - \mu)(\mu_c - \mu)^T
\]  

(5.30)
Linear Discriminant Analysis (LDA), $\ell$ classes

- **Within class scatter matrix**

  \[ S_W = \sum_{c=1}^{\ell} S_c \text{ where } S_c = \sum_{i \text{ is class } c} (x^{(i)} - \mu_c)(x^{(i)} - \mu_c)^T \quad (5.29) \]

  and $\mu_c$ is the class $c$ mean, $\mu_c = \frac{1}{n_c} \sum_{i \text{ is of class } c} x^{(i)}$

- **Between class scatter matrix**, $n_c = \text{number of samples of class } c.$

  \[ S_B = \sum_{c=1}^{\ell} n_c(\mu_c - \mu)(\mu_c - \mu)^T \quad (5.30) \]

- **First projection**: $a_1 \in \text{argmax}_{a \in \mathbb{R}^m} \frac{a^T S_B a}{a^T S_W a}$. 

Linear Discriminant Analysis (LDA), $\ell$ classes

- **Within class scatter matrix**

  \[
  S_W = \sum_{c=1}^{\ell} S_c \quad \text{where} \quad S_c = \sum_{i \text{ is class } c} (x^{(i)} - \mu_c)(x^{(i)} - \mu_c)^T \tag{5.29}
  \]

  and $\mu_c$ is the class $c$ mean, $\mu_c = \frac{1}{n_c} \sum_{\text{i is of class } c} x^{(i)}$

- **Between class scatter matrix**, $n_c =$ number of samples of class $c$.

  \[
  S_B = \sum_{c=1}^{\ell} n_c(\mu_c - \mu)(\mu_c - \mu)^T \tag{5.30}
  \]

- **First projection**: $a_1 \in \arg\max_{a \in \mathbb{R}^m} \frac{a^T S_B a}{a^T S_W a}$. Second orthogonal to first,

  $a_2 \in \arg\max_{a \in \mathbb{R}^m : \langle a, a_1 \rangle = 0} \frac{a^T S_B a}{a^T S_W a}$
Linear Discriminant Analysis (LDA), \( \ell \) classes

- Within class scatter matrix

\[
S_W = \sum_{c=1}^{\ell} S_c \quad \text{where} \quad S_c = \sum_{i \text{ is class } c} (x^{(i)} - \mu_c)(x^{(i)} - \mu_c)^T
\]  

(5.29)

and \( \mu_c \) is the class \( c \) mean, \( \mu_c = \frac{1}{n_c} \sum_{i \text{ is class } c} x^{(i)} \)

- Between class scatter matrix, \( n_c = \text{number of samples of class } c \).

\[
S_B = \sum_{c=1}^{\ell} n_c(\mu_c - \mu)(\mu_c - \mu)^T
\]  

(5.30)

- First projection: \( a_1 \in \text{argmax}_{a \in \mathbb{R}^m} \frac{a^T S_B a}{a^T S_W a} \). Second orthogonal to first, \( a_2 \in \text{argmax}_{a \in \mathbb{R}^m : \langle a, a_1 \rangle = 0} \frac{a^T S_B a}{a^T S_W a} \), and so on.
Linear Discriminant Analysis (LDA), $\ell$ classes

- **Within class scatter matrix**

  $$S_W = \sum_{c=1}^{\ell} S_c \text{ where } S_c = \sum_{\text{i is class } c} (x^{(i)} - \mu_c)(x^{(i)} - \mu_c)^T$$  \hspace{1cm} (5.29)

  and $\mu_c$ is the class $c$ mean, $\mu_c = \frac{1}{n_c} \sum_{\text{i is of class } c} x^{(i)}$

- **Between class scatter matrix**, $n_c =$ number of samples of class $c$.

  $$S_B = \sum_{c=1}^{\ell} n_c(\mu_c - \mu)(\mu_c - \mu)^T$$  \hspace{1cm} (5.30)

- **First projection**: $a_1 \in \arg\max_{a \in \mathbb{R}^m} \frac{a^T S_B a}{a^T S_W a}$. Second orthogonal to first, $a_2 \in \arg\max_{a \in \mathbb{R}^m : \langle a, a_1 \rangle = 0} \frac{a^T S_B a}{a^T S_W a}$, and so on.

- In general, directions of greatest variance of the matrix $S_W^{-1} S_B$, maximizes the between class spread relative to within-class spread. I.e., we can do PCA on $S_W^{-1} S_B$ to get LDA reduction.
**Linear Discriminant Analysis (LDA), $\ell$ classes**

- **Within class scatter matrix**
  \[
  S_W = \sum_{c=1}^{\ell} S_c \quad \text{where} \quad S_c = \sum_{i \text{ is class } c} (x^{(i)} - \mu_c) (x^{(i)} - \mu_c)^T \quad (5.29)
  \]
  and $\mu_c$ is the class $c$ mean, $\mu_c = \frac{1}{n_c} \sum_{i \text{ is class } c} x^{(i)}$

- **Between class scatter matrix**, $n_c =$ number of samples of class $c$.
  \[
  S_B = \sum_{c=1}^{\ell} n_c (\mu_c - \mu)(\mu_c - \mu)^T \quad (5.30)
  \]

- **First projection**: $a_1 \in \arg\max_{a \in \mathbb{R}^m} a^T S_B a \ a^T S_{ Wa }$. Second orthogonal to first, $a_2 \in \arg\max_{a \in \mathbb{R}^m : \langle a, a_1 \rangle = 0} a^T S_B a \ a^T S_{ Wa }$, and so on.

- In general, directions of greatest variance of the matrix $S_W^{-1} S_B$, maximizes the between-class spread relative to within-class spread. I.e., we can do PCA on $S_W^{-1} S_B$ to get LDA reduction.

- **LDA dimensionality reduction**: find $\ell' \times m$ matrix linear transform on $x$, project to subspace corresponding to $\ell'$ greatest eigenvalues of $S_W^{-1} S_B$. 
LDA projection, dimensionality upper limits of transform

- $\ell' \leq \ell - 1$ is highest dimensional subspace we can project onto since $S_B$ has rank at most $\ell - 1$. 

\[
S_B = \left( \mu_1 \mu_2 \ldots \mu_{\ell} \right) \left( \mu_1 \mu_2 \ldots \mu_{\ell} \right)^T
\]
LDA projection, dimensionality upper limits of transform

- \( \ell' \leq \ell - 1 \) is highest dimensional subspace we can project onto since \( S_B \) has rank at most \( \ell - 1 \).
- Between class scatter matrix, \( n_c = \) number of samples of class \( c \).

\[
S_B = \sum_{c=1}^{\ell} n_c (\mu_c - \mu)(\mu_c - \mu)^T
\]

(5.31)
LDA projection, dimensionality upper limits of transform

- $\ell' \leq \ell - 1$ is highest dimensional subspace we can project onto since $S_B$ has rank at most $\ell - 1$.
- Between class scatter matrix, $n_c = \text{number of samples of class } c$.
  \[
  S_B = \sum_{c=1}^{\ell} n_c (\mu_c - \mu)(\mu_c - \mu)^T \tag{5.31}
  \]
- Each vector $\in \mathbb{R}^m$, we have $\ell$ rank-1 updates, so immediately rank at most $\ell$ (ignoring matrix conditioning and potential numerical issues).
LDA projection, dimensionality upper limits of transform

- $\ell' \leq \ell - 1$ is highest dimensional subspace we can project onto since $S_B$ has rank at most $\ell - 1$.
- Between class scatter matrix, $n_c = \text{number of samples of class } c$.
  \[
  S_B = \sum_{c=1}^{\ell} n_c (\mu_c - \mu)(\mu_c - \mu)^T \tag{5.31}
  \]
- Each vector $\in \mathbb{R}^m$, we have $\ell$ rank-1 updates, so immediately rank at most $\ell$ (ignoring matrix conditioning and potential numerical issues).
- However, we have
  \[
  S_B = \begin{bmatrix}
  (\mu_1 - \mu) & (\mu_2 - \mu) & \cdots & (\mu_\ell - \mu)
  \end{bmatrix}
  \begin{pmatrix}
  n_1 & 0 & \cdots & 0 \\
  0 & n_2 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & n_\ell
  \end{pmatrix}
  \begin{bmatrix}
  (\mu_1 - \mu)^T \\
  (\mu_2 - \mu)^T \\
  \vdots \\
  (\mu_\ell - \mu)^T
  \end{bmatrix}
  \tag{5.32}
  \]
LDA projection, dimensionality upper limits of transform

- $\ell' \leq \ell - 1$ is highest dimensional subspace we can project onto since $S_B$ has rank at most $\ell - 1$.
- Between class scatter matrix, $n_c = \text{number of samples of class } c$.
  \[ S_B = \sum_{c=1}^{\ell} n_c (\mu_c - \mu)(\mu_c - \mu)^T \]  
  (5.31)
- Each vector $\in \mathbb{R}^m$, we have $\ell$ rank-1 updates, so immediately rank at most $\ell$ (ignoring matrix conditioning and potential numerical issues).
- However, we have
  \[ S_B = \begin{bmatrix} (\mu_1 - \mu) & (\mu_2 - \mu) & \ldots & (\mu_\ell - \mu) \end{bmatrix} \begin{pmatrix} n_1 & 0 & \ldots & 0 \\ 0 & n_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & n_\ell \end{pmatrix} \begin{bmatrix} (\mu_1 - \mu)^T \\ (\mu_2 - \mu)^T \\ \vdots \\ (\mu_\ell - \mu)^T \end{bmatrix} \]  
  (5.32)
  and since $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$
LDA projection, dimensionality upper limits of transform

- \( \ell' \leq \ell - 1 \) is highest dimensional subspace we can project onto since \( S_B \) has rank at most \( \ell - 1 \).
- Between class scatter matrix, \( n_c = \) number of samples of class \( c \).

\[
S_B = \sum_{c=1}^{\ell} n_c (\mu_c - \mu)(\mu_c - \mu)^T \tag{5.31}
\]

- Each vector \( \in \mathbb{R}^m \), we have \( \ell \) rank-1 updates, so immediately rank at most \( \ell \) (ignoring matrix conditioning and potential numerical issues).
- However, we have

\[
S_B = \left[ (\mu_1 - \mu) \ (\mu_2 - \mu) \ldots (\mu_{\ell} - \mu) \right] \begin{pmatrix} n_1 & 0 & \ldots & 0 \\ 0 & n_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & n_{\ell} \end{pmatrix} \begin{bmatrix} (\mu_1 - \mu)^T \\ (\mu_2 - \mu)^T \\ \vdots \\ (\mu_{\ell} - \mu)^T \end{bmatrix} \tag{5.32}
\]

and since \( \text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)) \), and since \( \sum_{i=1}^{\ell} n_i(\mu_i - \mu) = 0 \) which means that \( \text{rank}([(\mu_1 - \mu) \ (\mu_2 - \mu) \ldots (\mu_{\ell} - \mu)]) \leq \ell - 1 \).
LDA projection, dimensionality upper limits of transform

- \( \ell' \leq \ell - 1 \) is highest dimensional subspace we can project onto since \( S_B \) has rank at most \( \ell - 1 \).
- Between class scatter matrix, \( n_c = \) number of samples of class \( c \).

\[
S_B = \sum_{c=1}^{\ell} n_c(\mu_c - \mu)(\mu_c - \mu)^T \tag{5.31}
\]

- Each vector \( \in \mathbb{R}^m \), we have \( \ell \) rank-1 updates, so immediately rank at most \( \ell \) (ignoring matrix conditioning and potential numerical issues).
- However, we have

\[
S_B = \left[ (\mu_1 - \mu) \ (\mu_2 - \mu) \ \ldots \ (\mu_\ell - \mu) \right] \begin{pmatrix} n_1 & 0 & \ldots & 0 \\ 0 & n_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & n_\ell \end{pmatrix} \begin{pmatrix} (\mu_1 - \mu)^T \\ (\mu_2 - \mu)^T \\ \vdots \\ (\mu_\ell - \mu)^T \end{pmatrix} \tag{5.32}
\]

and since \( \text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)) \), and since \( \sum_{i=1}^{\ell} n_i(\mu_i - \mu) = 0 \) which means that \( \text{rank}\left(\left[ (\mu_1 - \mu) \ (\mu_2 - \mu) \ \ldots \ (\mu_\ell - \mu) \right] \right) \leq \ell - 1 \), thus \( \text{rank}(S_B) \leq \ell - 1 \).
The next slide is a repeat from earlier.
Uniformly at random unit vectors on unit sphere

- Let $S_{m-1}(r)$ indicate surface “area” of $m$-dimensional sphere radius $r$.
- Define uniform distribution $U_{S_{m-1}(1)}$ on $S_{m-1}(1)$ and independently draw two random vectors $x, y \sim U_{S_{m-1}(1)}$, hence $\|x\|_2 = \|y\|_2 = 1$.
- Two vectors are orthogonal if $\langle x, y \rangle = 0$ and are nearly so if $|\langle x, y \rangle| < \epsilon$ for small $\epsilon$.
- It can be shown that if $x, y$ are independent random vectors uniformly distributed on the $m$-dimensional sphere, then:

$$\Pr(|\langle x, y \rangle| < \epsilon) > 1 - e^{-m\epsilon^2/2} \quad (5.17)$$

which means that uniformly at random vectors in high dimensional space are almost always nearly orthogonal!

- One of the reasons why high dimensional random projections preserve information — if two random high dimensional vectors are almost orthogonal, then projections onto them will also be almost orthogonal.
Non-linear Dimensionality Reduction: Autoencoders

- Construct two mappings, an encoder $\varphi_{\theta_e} : \mathbb{R}^m \rightarrow \mathbb{R}^{m'}$ and a decoder $\varphi_{\theta_d} : \mathbb{R}^{m'} \rightarrow \mathbb{R}^m$. 
Non-linear Dimensionality Reduction: Autoencoders

- Construct two mappings, an encoder $e_{\theta_e} : \mathbb{R}^m \rightarrow \mathbb{R}^{m'}$ and a decoder $d_{\theta_d} : \mathbb{R}^{m'} \rightarrow \mathbb{R}^m$.

- Given $x \in \mathbb{R}^m$, $d_{\theta_d}(e_{\theta_e}(x))$ is $x$ reconstructed, and this has error $E_p(x)[\|x - d_{\theta_d}(e_{\theta_e}(x))\|]$. 

---

Prof. Jeff Bilmes
EE511/Spring 2020/Adv. Intro ML - Week 5 - April 27th/29nd, 2020
Construct two mappings, an encoder $\mathbf{e}_{\theta_e} : \mathbb{R}^m \rightarrow \mathbb{R}^{m'}$ and a decoder $\mathbf{d}_{\theta_d} : \mathbb{R}^{m'} \rightarrow \mathbb{R}^m$.

Given $x \in \mathbb{R}^m$, $\mathbf{d}_{\theta_d}(\mathbf{e}_{\theta_e}(x))$ is $x$ reconstructed, and this has error $E_p(x)[\|x - \mathbf{d}_{\theta_d}(\mathbf{e}_{\theta_e}(x))\|]$.

Data set $\mathcal{D}_n = \{x^{(i)}\}_i$, optimization, with $\theta = (\theta_e, \theta_d)$ becomes:

$$\min_{\theta} E_p(x)[\|x - \mathbf{d}_{\theta_d}(\mathbf{e}_{\theta_e}(x))\|] \approx \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} [\|x^{(i)} - \mathbf{d}_{\theta_d}(\mathbf{e}_{\theta_e}(x^{(i)}))\|] \quad (5.33)$$
Non-linear Dimensionality Reduction: Autoencoders

- Construct two mappings, an encoder $\Phi_e : \mathbb{R}^m \rightarrow \mathbb{R}^{m'}$ and a decoder $\Phi_d : \mathbb{R}^{m'} \rightarrow \mathbb{R}^m$.
- Given $x \in \mathbb{R}^m$, $\Phi_d(\Phi_e(x))$ is $x$ reconstructed, and this has error $E_p(x)[\|x - \Phi_d(\Phi_e(x))\|]$.
- Data set $D_n = \{x^{(i)}\}_i$, optimization, with $\theta = (\theta_e, \theta_d)$ becomes:

$$\min_{\theta} E_p(x)[\|x - \Phi_d(\Phi_e(x))\|] \approx \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \|x^{(i)} - \Phi_d(\Phi_e(x^{(i)}))\| \quad (5.33)$$

- Then $\Phi_e(x)$ is our learnt representation of $x$. If $m' < m$ or $m' \ll m$ this is dimensionality reduction, similar to compression.
Non-linear Dimensionality Reduction: Autoencoders

- Construct two mappings, an encoder $e_{\theta_e} : \mathbb{R}^m \rightarrow \mathbb{R}^{m'}$ and a decoder $d_{\theta_d} : \mathbb{R}^{m'} \rightarrow \mathbb{R}^m$.
- Given $x \in \mathbb{R}^m$, $d_{\theta_d}(e_{\theta_e}(x))$ is $x$ reconstructed, and this has error $E_p(x)[\|x - d_{\theta_d}(e_{\theta_e}(x))\|]$.
- Data set $\mathcal{D}_n = \{x^{(i)}\}_i$, optimization, with $\theta = (\theta_e, \theta_d)$ becomes:

$$\min_{\theta} E_p(x)[\|x - d_{\theta_d}(e_{\theta_e}(x))\|] \approx \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} [\|x^{(i)} - d_{\theta_d}(e_{\theta_e}(x^{(i)}))\|] \quad (5.33)$$

- Then $e_{\theta_e}(x)$ is our learnt representation of $x$. If $m' < m$ or $m' \ll m$ this is dimensionality reduction, similar to compression.
- As $m'$ gets smaller, more pressure to ensure the dimensions are independent (increase capacity).
Non-linear Dimensionality Reduction: Autoencoders

- Construct two mappings, an encoder \( e_{\theta_e} : \mathbb{R}^m \rightarrow \mathbb{R}^{m'} \) and a decoder \( d_{\theta_d} : \mathbb{R}^{m'} \rightarrow \mathbb{R}^m \).
- Given \( x \in \mathbb{R}^m \), \( d_{\theta_d}(e_{\theta_e}(x)) \) is \( x \) reconstructed, and this has error \( E_p(x)[\|x - d_{\theta_d}(e_{\theta_e}(x))\|] \).
- Data set \( D_n = \{x^{(i)}\}_i \), optimization, with \( \theta = (\theta_e, \theta_d) \) becomes:

\[
\min_{\theta} E_p(x)[\|x - d_{\theta_d}(e_{\theta_e}(x))\|] \approx \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} [\|x^{(i)} - d_{\theta_d}(e_{\theta_e}(x^{(i)}))\|] \quad (5.33)
\]

- Then \( e_{\theta_e}(x) \) is our learnt representation of \( x \). If \( m' < m \) or \( m' \ll m \) this is dimensionality reduction, similar to compression.
- As \( m' \) gets smaller, more pressure to ensure the dimensions are independent (increase capacity).
- Has become very successful, many different flavors (denoising autoencoder, variational autoencoder) using (deep) neural networks as encoders/decoders. representation learning.
DNN-based Auto-Encoder

- Encoder from $x$ to $\tilde{x} = \vartheta_d(\varrho_e(x))$ where $m = 9$ and $m' = 3$. 

![Diagram of DNN-based Auto-Encoder](image_url)
DNN-based Auto-Encoder

- Encoder from $x$ to $\tilde{x} = \vartheta_d(\varepsilon_e(x))$ where $m = 9$ and $m' = 3$.
- In some sense, a form of non-linear PCA.

![Diagram of DNN-based Auto-Encoder](image-url)