

CSDS 440: Machine Learning

Soumya Ray (he/him, sray@case.edu)

Olin 516

Office hours T, Th 11:15-11:45 or by appointment

Zoom Link

Recap

- In the feature selection problem we seek to identify features used by _____.
- Possible issues: FS needs a c____ s____, which is generally intractable. Also there is a c____ and e____ problem.
- Filter methods d____ the FS and learning tasks. They s____ each feature in turn and keep the _____.
- Mutual information checks r____ but not r_____.
- We fix this by moving to a s____ procedure. Each feature is scored based on _____.
- A reasonable filter method is the ____ criterion, which scores a feature by _____.
- What are some pros and cons of filter methods?
- Wrapper methods work by s____ through the feature space using the l____ p____ as a guide.
- In Forward feature selection we start with an e____ set of features. We test each feature to see if the g____ e____ improves when a____ to the set. To do this we need a v____ set. We a____ the b____ feature to our feature set and iterate.
- To obtain valid empirical results we must do FS i____ each fold. This is because _____.
- A problem with forward FS is _____.
- In Backward feature selection we start with a f____ set of features. We test each feature to see if the g____ e____ improves when r____ from the set. To do this we need a v____ set. We r____ the w____ feature from our feature set and iterate.
- A problem with backward FS is _____.
- How does boosted feature selection work?
- What are some pros and cons of wrapper methods?

Today

- Feature Selection and Dimensionality Reduction

Approach 3: Embedded methods

- Embedded methods alter the objective functions of learning algorithms to *simultaneously* select features and learn the classifier
 - A good way to incorporate the learner into feature selection, but requires modifying learning algorithms

Modifying objectives

- Many learning algorithms optimize a loss function on the training sample:


$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

- Embedded methods add terms to this objective to encourage “sparseness”
 - i.e., encourage many zero w_i ’s

Modifying objectives

- An objective function that does this might be:

$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_0$$



The zero-norm is the number of elements in a vector that are non-zero

- This isn't continuous, but it indicates connections to overfitting control


Feature Selection and Overfitting Control

- We also modified learning objectives to control overfitting
- In fact, FS and OC have similar objectives
 - Both are trying to “simplify” the learned model and encourage generalization
 - Selecting a good set of features should help with overfitting control
 - Conversely, a classifier that is robust to overfitting should be using a good set of features

Feature Selection through Overfitting Control

- So if we consider our usual overfitting control strategy

The 2-norm, or “L2 penalty”



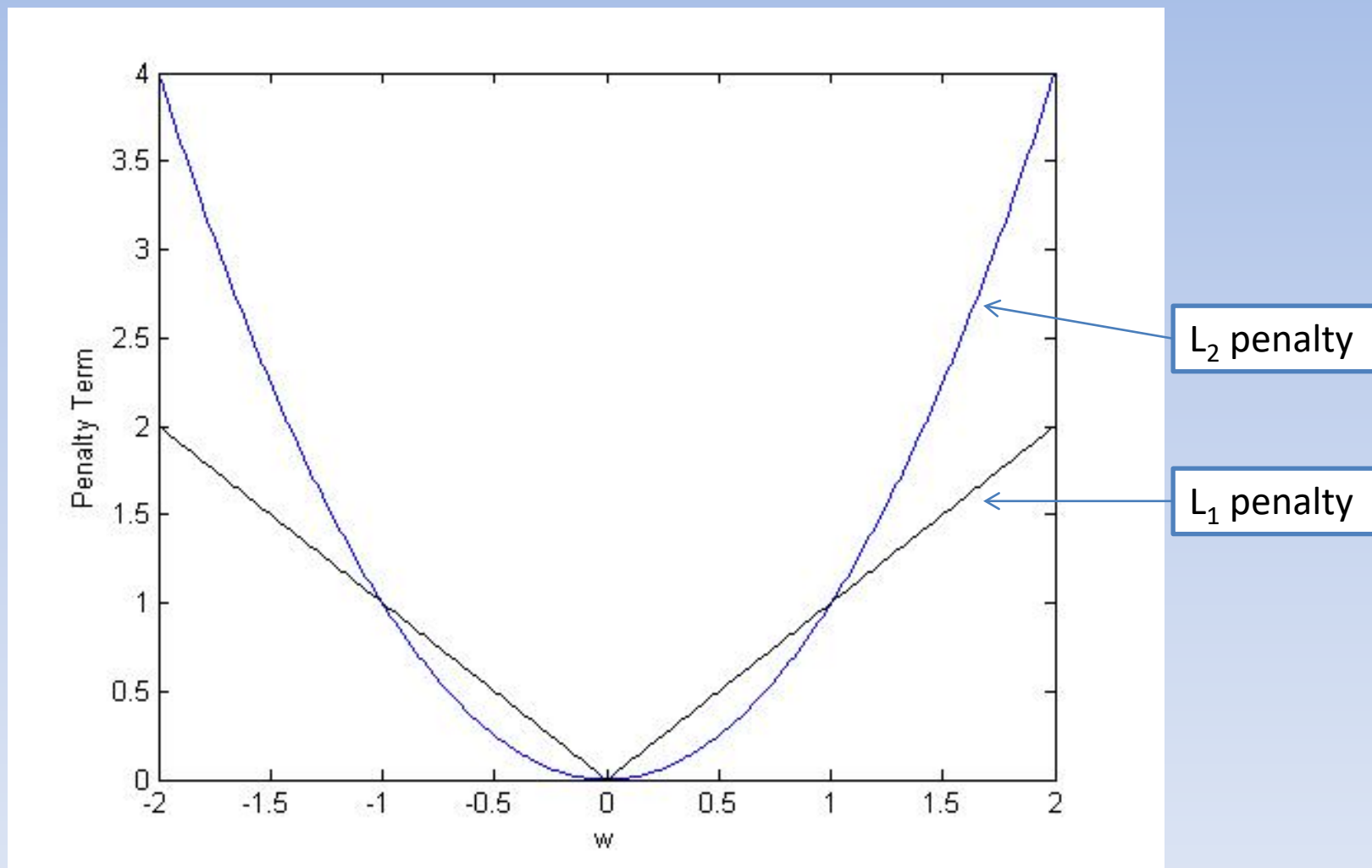
$$L_{OC}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2, \quad \|\mathbf{w}\|_2^2 = \sum_i w_i^2$$

- This also encourages sparseness because each nonzero w pays a penalty in the objective
 - Is this a good way to select features?

Yes and No

- It turns out that the 2-norm penalty on \mathbf{w} is somewhat effective, but it is possible to do better
- This is because the penalty for any nonzero w decreases quadratically as w becomes small, so usually the solution generally ends up with a lot of very small, but nonzero w 's

Illustration




The LASSO

- The L_1 penalty:

$$L_{FS}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_1, \|\mathbf{w}\|_1 = \sum_i |w_i|$$

The one-norm is the sum of the absolute values of elements in a vector



- Does not have this problem
- This is also called the “LASSO” penalty (“least absolute shrinkage and selection operator”) in the statistics literature

Why does the lasso work?

- Another way to write the objectives

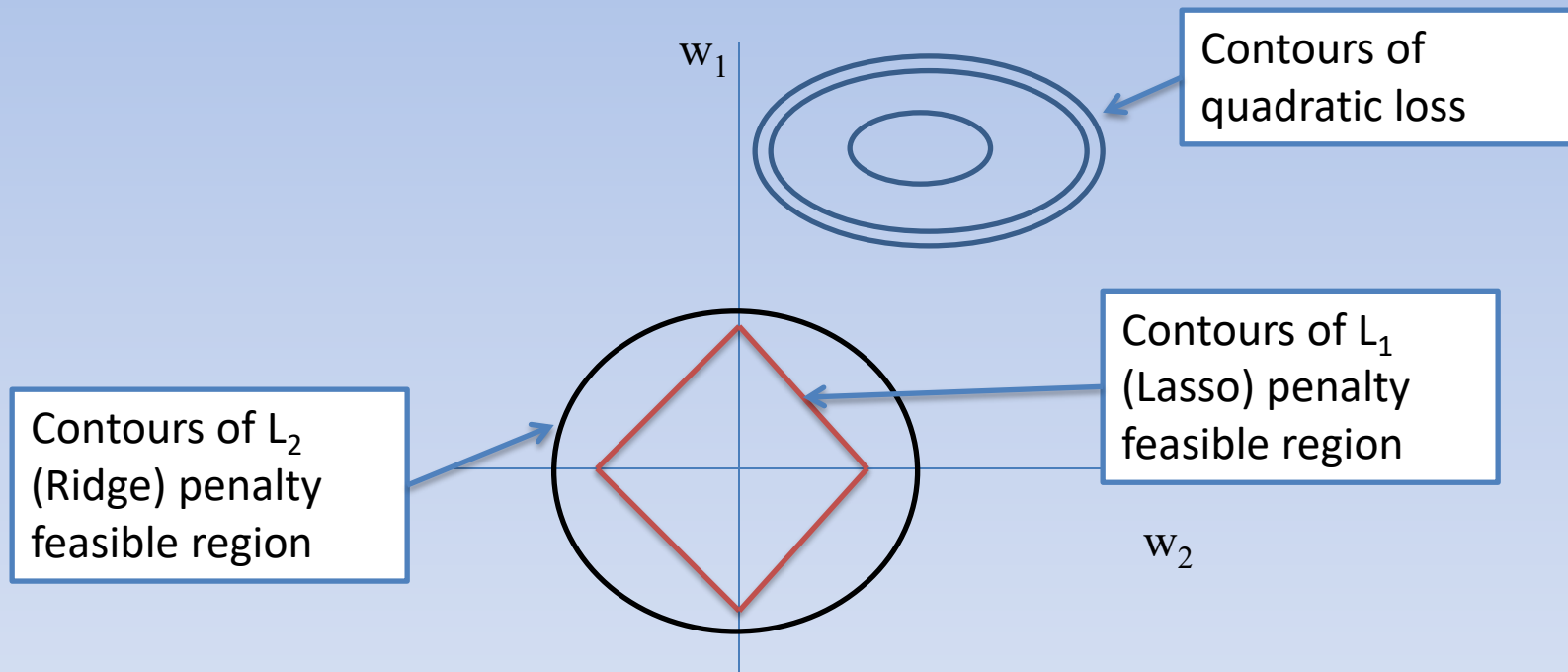
$$\min_{\mathbf{w}} L_{FS}(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_1$$

$$\equiv \min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2, s.t. \|\mathbf{w}\|_1 \leq B$$

$$\min_{\mathbf{w}} L_{OC}(\mathbf{w}) = \min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_2^2$$

$$\equiv \min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2, s.t. \|\mathbf{w}\|_2^2 \leq B$$

Illustration



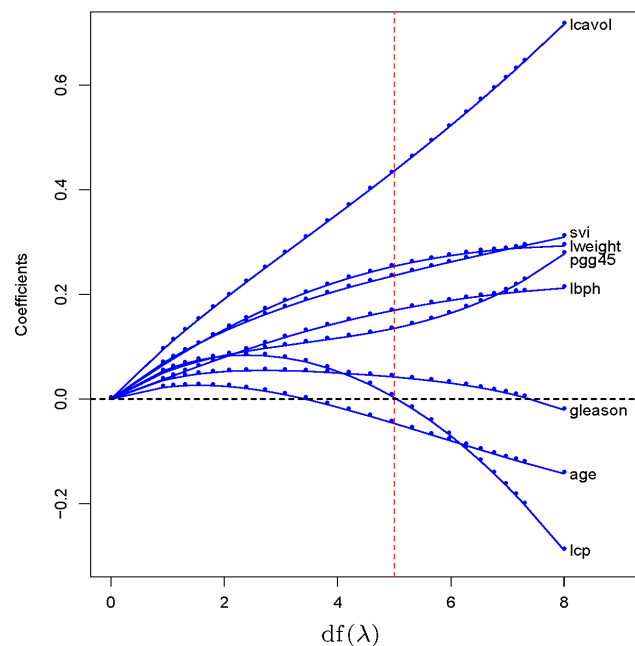


FIGURE 3.8. Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter λ is varied. Coefficients are plotted versus $df(\lambda)$, the effective degrees of freedom. A vertical line is drawn at $df = 5.0$, the value chosen by cross-validation.

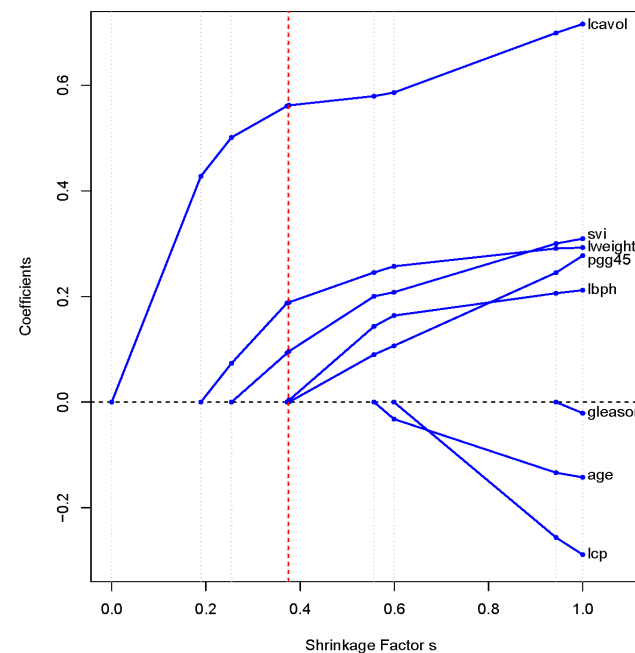


FIGURE 3.10. Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t / \sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 9; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed;

Optimizing the Lasso

- Various strategies have been worked out for optimizing the lasso
 - Introducing auxiliary variables for \mathbf{w}
 - Alternating Direction Method of Multipliers
 - Pathwise Coordinate Descent

Nice property of the Lasso

- What happens if the true model is not linear?
 - Turns out that, under some suitable conditions, the Lasso will *still* select the correct set of features!!
 - Y. Zhang, S. Ray and W. Guo (2016).
On the Consistency of Feature Selection with Lasso for Non-Linear Targets.
Appears in the Proceedings of the 33rd International Conference on Machine Learning (ICML-16), New York City, New York, USA

Summary: Feature Selection

- Filters
 - Computationally cheap, can perform well in some cases, but ignore classifier and have trouble with missing data
 - The JMI score is a good approach to try here
- Wrappers
 - Generally perform well when lots of data available, but computationally expensive
 - Boosting is a good approach to try here
- Embedded
 - Often perform well, a good middle ground, but need to modify learning formulations
 - Using the Lasso works well here

Dimensionality Reduction

- A problem related to feature selection, but with some differences
- Primarily, the new set of “features” does not have to be a subset of the original feature set
- Also, not necessarily interested in classification

Dimensionality Reduction

- Given: A set of examples \mathbf{x}_i over D features in \mathbb{R}^D
- Do: Find a *subspace* of dimension d , $d < D$, so that the *geometry of $\{\mathbf{x}_i\}$ projected into the d -dimensional space is preserved as closely as possible*

FS and DR

Feature Selection

- Outputs subset of original features
- Takes class label and generalization into account
- Does not affect or improves interpretability of classifier
- Combinatorial optimization problem, generally only heuristically solvable

Dimensionality Reduction

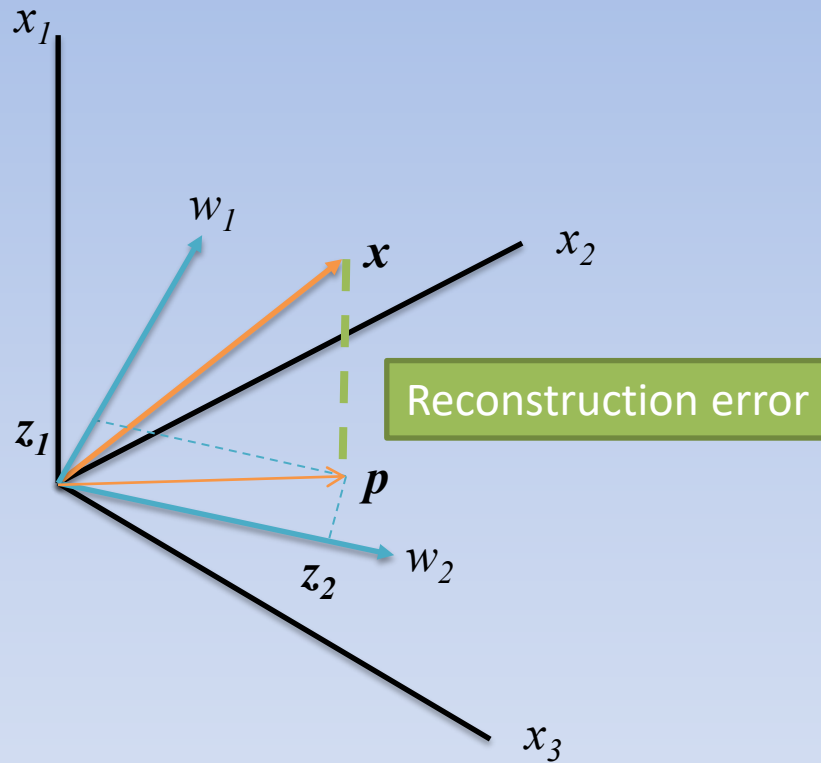
- May output combinations of original features
- General “geometric” criteria, may not depend on class labels or generalization
- May affect or hurt interpretability of classifier
- Smooth, sometimes convex optimization problem, may be solvable efficiently in closed form

Principal Components Analysis

- Given: set of points \mathbf{x}_i centered in \mathbb{R}^D
- Find: an orthonormal set of d vectors $\mathbf{w}_j \in \mathbb{R}^D$, and the associated coefficients $\mathbf{z}_i \in \mathbb{R}^d$ so that the *projected points* $\mathbf{p}_i = W\mathbf{z}_i$ minimize the *reconstruction error*:

$$J(W, Z) = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i - \mathbf{p}_i\|^2 = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i - W\mathbf{z}_i\|^2$$

Projection

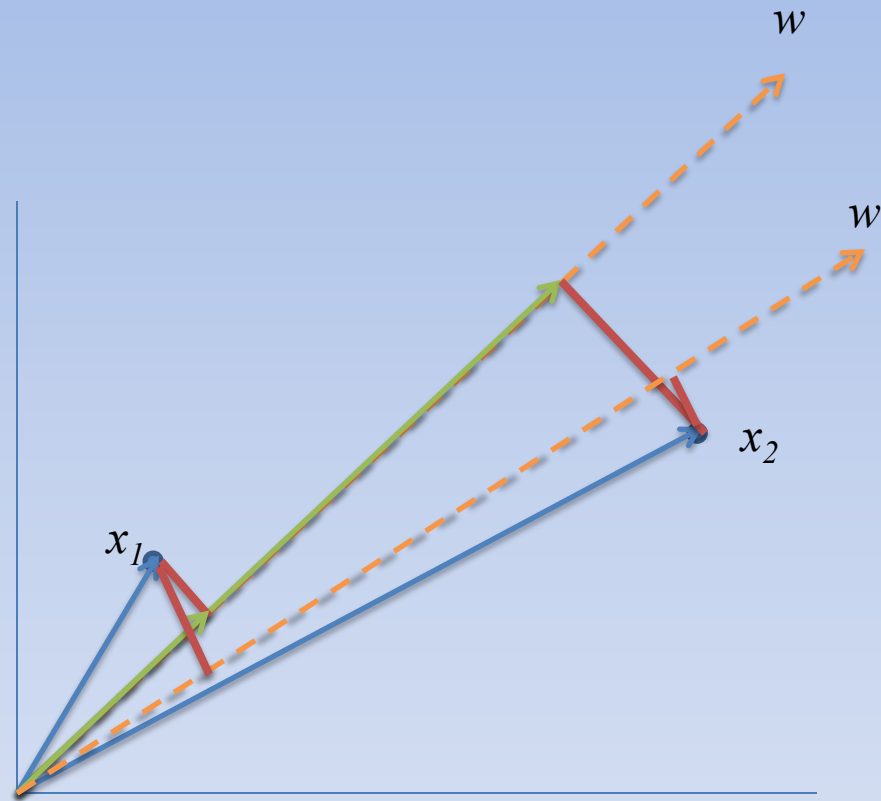


Solution to PCA

- We can solve the optimization problem incrementally by asking for the first projection:

$$J(\mathbf{w}, z) = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i - z_i \mathbf{w}\|^2$$

Example



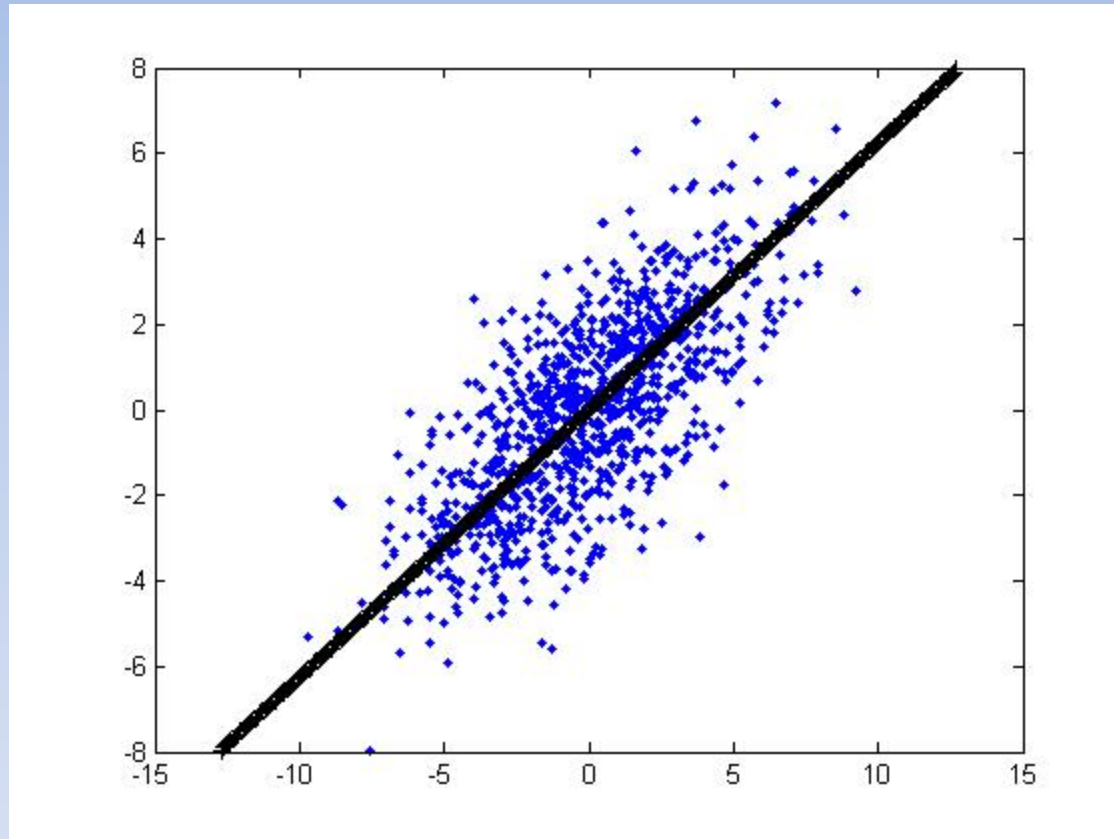
Solution to PCA

- We can show that the solution \mathbf{w} is the eigenvector of the **empirical covariance** matrix (for centered data):

$$\Sigma = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T$$

- With the largest eigenvalue
 - This is the direction that maximizes the variance of the data
- The coefficients z_i are given by: $z_i = \mathbf{w}^T \mathbf{x}_i$

Example



Solution to PCA

- Continuing, we can repeat for the residuals $\mathbf{x}' = (\mathbf{x} - \mathbf{z}\mathbf{w})$
- The d -dimensional linear projection that minimizes reconstruction error is then the d eigenvectors of Σ corresponding to the largest eigenvalues

Scree plots

- How to choose how many dimensions?
 - (no external signal like generalization error)
- Can plot the reconstruction error as a function of dimensions, and stop when reconstruction error goes below some threshold (e.g. 5%)
- Also can plot the eigenvalues of the retained eigenvectors
 - “Scree” plots
 - Look for “elbows”

Notes on PCA

- Generally want input to be standardized (zero mean, unit variance)
 - Since PCA works on the covariance matrix, can be affected if features have different scales
- Assumes data are generated by a Gaussian distribution
- Note that the components are generally not sparse in terms of the original features
 - i.e. most of the components will be defined in terms of most of the features, the w_{ij} 's are generally nonzero

Notes on PCA

- Implementation requires eigenvector decomposition, which can be expensive
- Iterative solutions are possible that empirically converge much faster
- Extensions can take class labels into account