CS 189 Midterm Review

Fall 2018

Reminders

Exam is Thursday, Oct. 18, 9:30-11:00 in LKS 245.
DSP exam 9:00 in HP Auditorium.

SLEEP.

2 hours of extra sleep will help you more than 2 hours of cramming.

do NOT cheat.

A ‘bad’ grade is better than a 0 grade, or worse disciplinary action.

Agenda

1. **Regression**
   a. Least Squares
   b. Total Least Squares
2. **Classification**
   a. SVMs
   b. Perceptron
3. **Features**
   a. Kernels: a way to ‘increase’ dimensionality
   b. PCA: dimensionality reduction.
   c. CCA: correlating two datasets
4. **Optimization**
   a. Gradient descent
   b. Stochastic gradient descent

For each topic:

- All together: mini-recap of material
- (5-10 minutes)
- individual/small groups: try practice problems (10 minutes)
- All together: go over solutions: (10 minutes)

A **subset** of the material you are responsible for.

(quick warm-up). Individually :

1 Connections between OLS, Ridge Regression, TLS, PCA, and CCA

We will review several topics we have learned so far: ordinary least-squares, ridge regression, total least squares, principle component analysis, and canonical correlation analysis. We emphasize their basic attributes, including the objective functions and the explicit form of their solutions. We will also discuss the connections and distinctions between these methods.

(a) What are the objective functions and closed-form solutions to OLS, ridge regression, and TLS? How do the probabilistic interpretations vary?

(b) Consider the matrix inversion in the solution to OLS, ridge regression, and TLS. How do the eigenvalues compare to those of the matrix $\mathbf{X}^\top\mathbf{X}$?

(c) Suppose you have a data matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$ and output $\mathbf{y} \in \mathbb{R}^{n\times 1}$. Use PCA to compute the first k principal components of $[\mathbf{X} \quad \mathbf{y}]$. Describe how this solution would relate to performing TLS on the problem.

(d) Suppose you have a problem of fitting $\mathbf{y}$ from $\mathbf{X} \in \mathbb{R}^{n\times d}$. d is large in this case, and you want to reduce the dimensionality before doing the fitting. You have two choices for the dimensionality reduction: the first one being do PCA on $\mathbf{X}$ and the second one being use CCA between $\mathbf{X}$ and $\mathbf{y}$ to reduce dimension of $\mathbf{X}$ with $\mathbf{X}\mathbf{U}_p$, where $\mathbf{U}_p \in \mathbb{R}^{d\times p}$ is the p-dimensional projection matrix solved using CCA. Which of them is better in general?

Regression: Least Squares

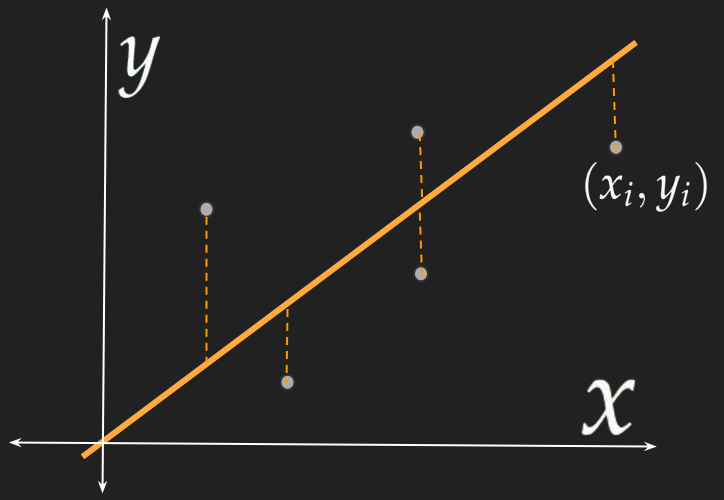
Ordinary least squares (OLS)

$$\boxed{\min_{w} \|Xw - y\|_2^2}$$

Noise Model:

$$y = Xw^* + \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$



Regression: Least Squares

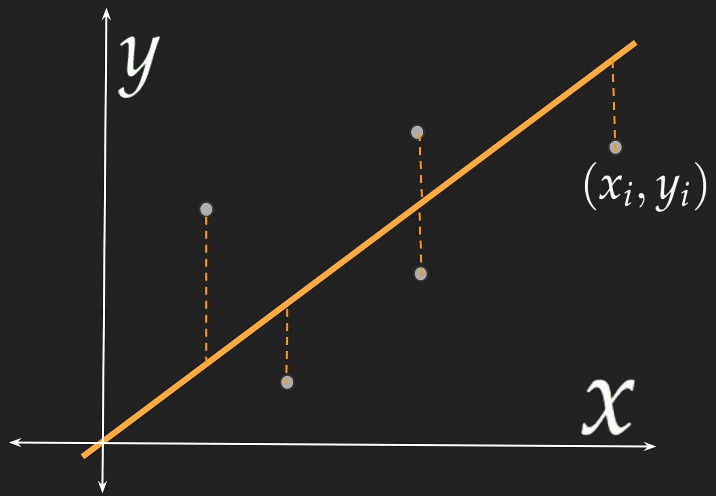
Ordinary least squares (OLS)

$$\boxed{\min_{w} \|Xw - y\|_2^2}$$

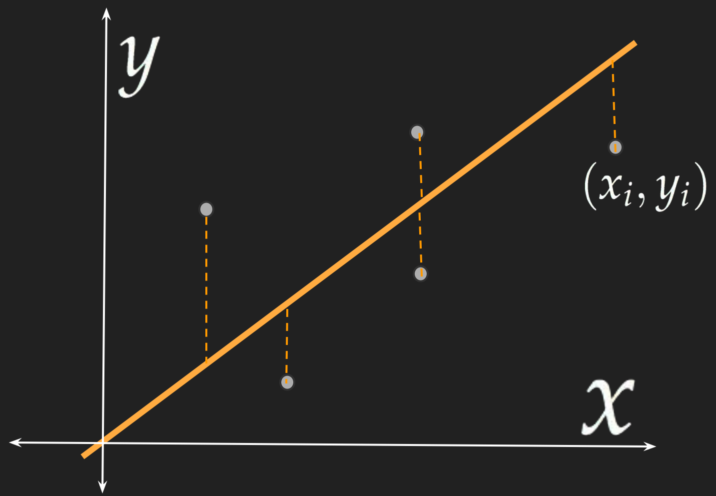
Ridge Regression

$$\min_{w} \|Xw - y\|_2^2 + \lambda \|w\|_2^2$$

a **regularization**



Regression: Least Squares



Ordinary least squares (OLS)

$$\boxed{\min_w \|Xw - y\|_2^2}$$

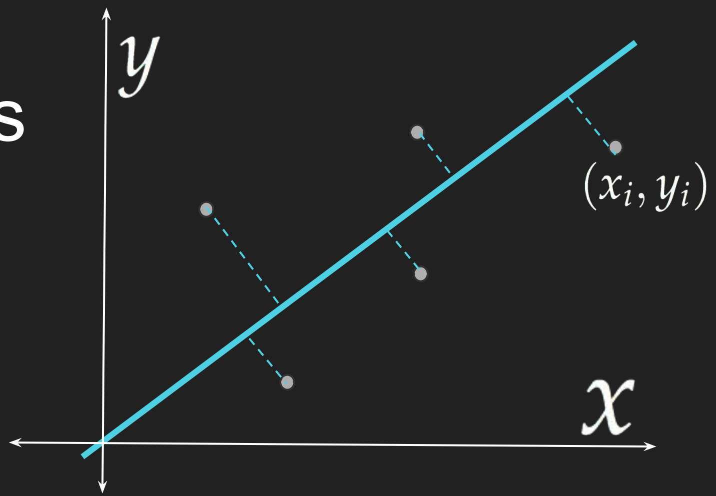
Ridge Regression

$$\min_w \|Xw - y\|_2^2 + \lambda \|w\|_2^2$$

a **regularization**

Why would we want to add the ridge regression term?

Regression: Total Least Squares



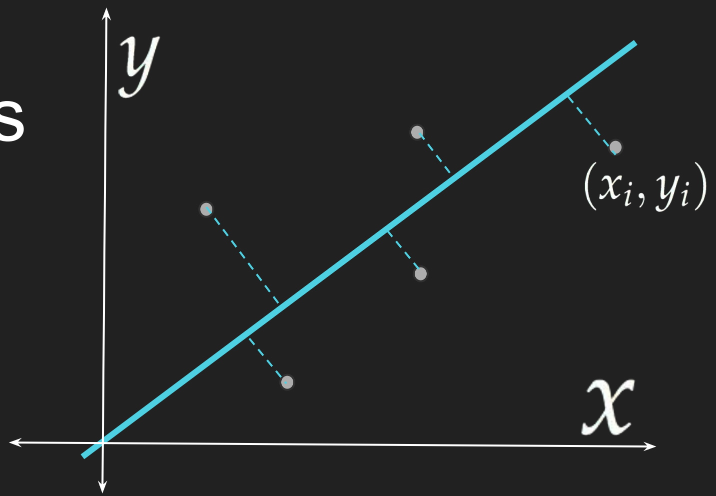
Ordinary least squares (OLS)

$$\boxed{\min_{w,\varepsilon_y} \|\varepsilon_y\|_2^2 \quad s.t.\ \varepsilon_y = Xw - y}$$

Total Least Squares

$$\boxed{\min_{w,\varepsilon_x,\varepsilon_y} \left\|\begin{bmatrix}\varepsilon_x & \varepsilon_y\end{bmatrix}\right\|_F^2 \quad s.t.\ y + \varepsilon_y = (X + \varepsilon_x)w}$$

Regression: Total Least Squares



Ordinary least squares (OLS)

$$\boxed{\min_{w,\varepsilon_y} \|\varepsilon_y\|_2^2 \quad s.t.\ \varepsilon_y = Xw - y}$$

Total Least Squares

$$\boxed{\min_{w,\varepsilon_x,\varepsilon_y} \left\|\begin{bmatrix}\varepsilon_x & \varepsilon_y\end{bmatrix}\right\|_F^2 \quad s.t.\ y + \varepsilon_y = (X + \varepsilon_x)w}$$

What does it mean??

Individually or in small groups ((b) if extra time)

2 Linear regression (16 points)

In this problem, we study the loss function for ridge regression:

$$\frac{1}{2}\|Xw - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2, \tag{1}$$

where X a data matrix in $\mathbb{R}^{n\times d}$ and y is the response in $\mathbb{R}^n$. Let the regularization weight $\lambda > 0$. Recall that the closed-form solution to ridge regression is

$$\hat{w} = (X^T X + \lambda I_d)^{-1} X^T y. \tag{2}$$

where $I_d \in \mathbb{R}^{d\times d}$ is an identity matrix of dimension d.

(a) (8 points) Augment the matrix X with d additional rows $ce_1^T, ce_2^T, \ldots, ce_d^T$ to get the matrix $X' \in \mathbb{R}^{n+d\times d}$, where c is a given constant and e_i^T is a unit vector whose ith element is 1 and the rest of the elements are zero, and augment y with d zeros to get $y' \in \mathbb{R}^{n+d}$:

$$X' = \begin{bmatrix} X \\ ce_1^T \\ ce_2^T \\ \vdots \\ ce_d^T \end{bmatrix} \qquad y' = \begin{bmatrix} y \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Write out the closed form solution for w' for the ordinary least squares problem on X', y'.

Derive the concrete value of c that corresponds to the ridge regression problem (1) in terms of λ. Conclude that the ridge regression estimate for w can be obtained by doing ordinary least squares on an augmented dataset.

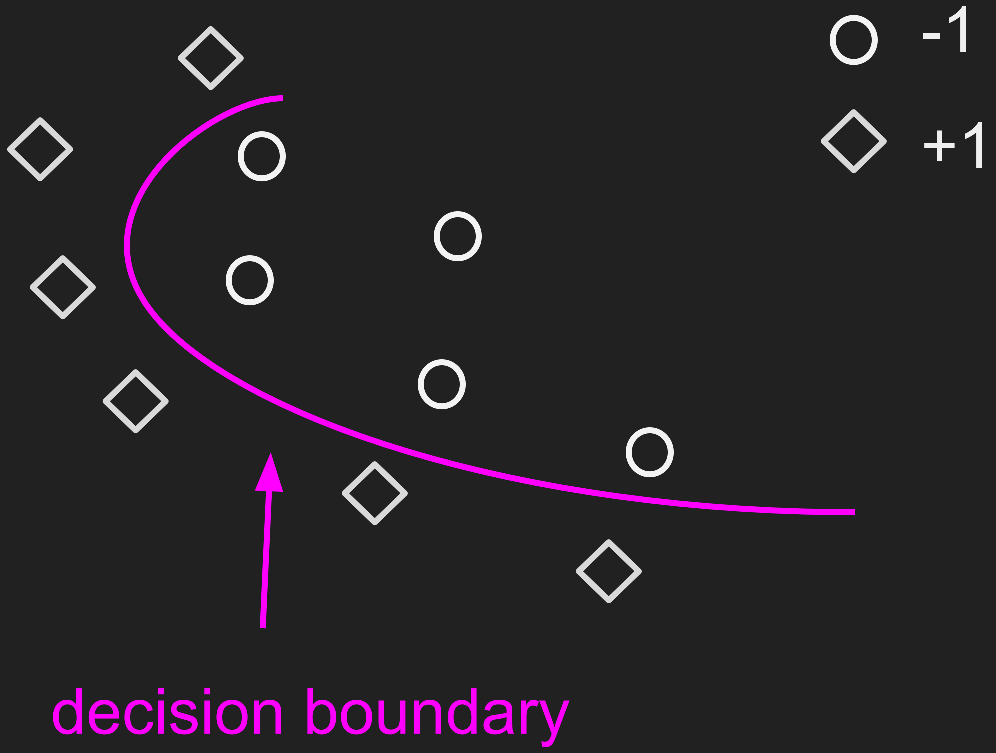
(b) (8 points) **Prove that applying gradient descent on the ridge regression loss function with a suitable fixed step size results in geometric convergence.** If $X^\top X$ has maximum and minimum eigenvalues M and m, **what fixed step size should we choose as a function of M, m, and ridge weight λ?**

Hint: Reduce the problem to ordinary least squares. Recall that applying gradient descent on the least squares problem

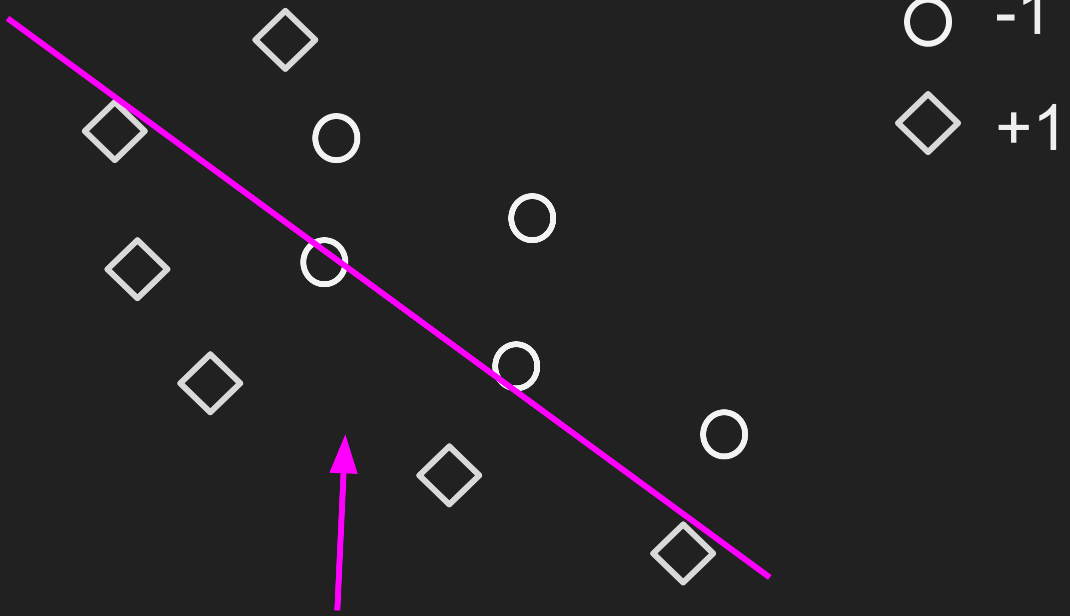
$$\min_x f(x) = \min_x \frac{1}{2}\|Ax - b\|_2^2$$

results in geometric convergence when $A^T A$ is positive definite and using a constant step size $\gamma = \frac{2}{\lambda_{\min}(A^T A) + \lambda_{\max}(A^T A)}$. Geometric convergence means that $f(x_k) - f(x^*) \le c' Q^k$ for some $0 \le Q < 1$ and for some $c' > 0$. You may use this result without proof.

Classification



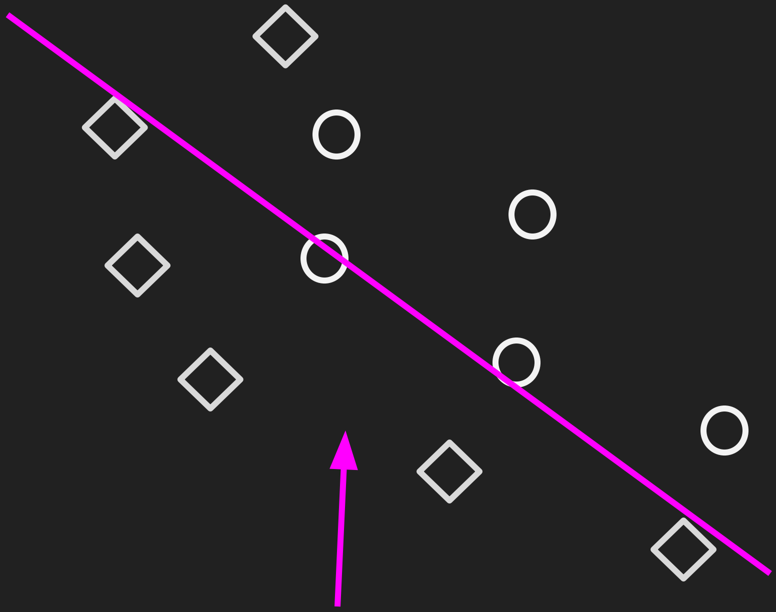
Classification



linear decision boundary

$$\boxed{\mathcal{H}_w = \{x : w^\top x = 0\}}$$

Classification



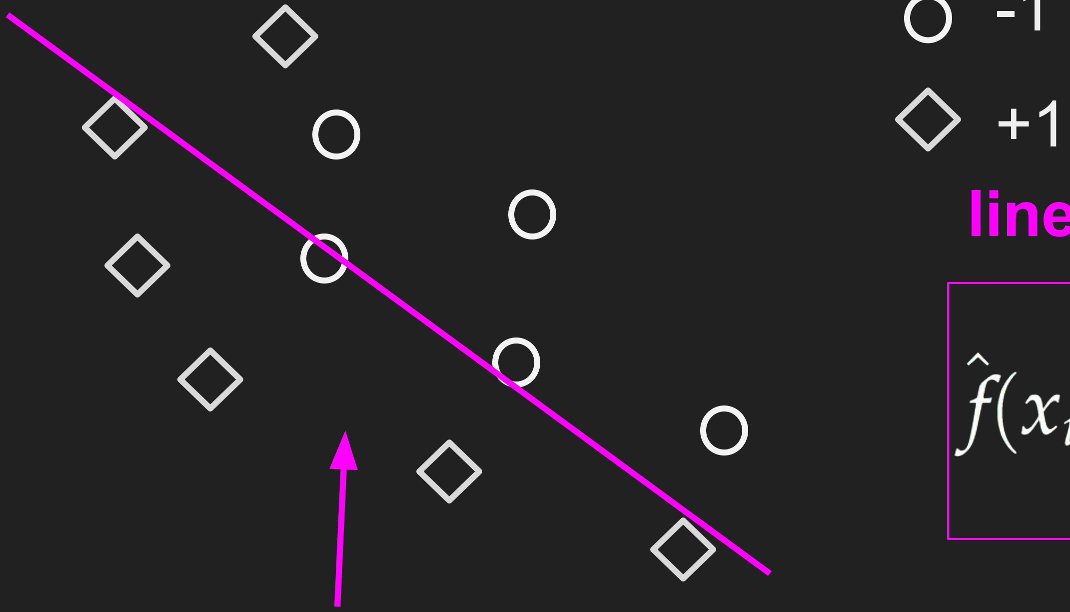
linear classifier:

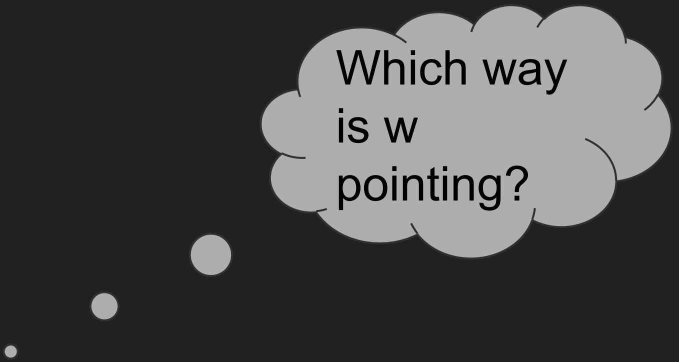
$$\boxed{\hat{f}(x_i) = \begin{cases} 1 & if\, w^\top x_i \geq 0 \\ -1 & if\, w^\top x_i < 0 \end{cases}}$$

linear decision boundary

$$\boxed{\mathcal{H}_w = \{x : w^\top x = 0\}}$$

Classification





linear classifier:

$$\boxed{\hat{f}(x_i) = \begin{cases} 1 & if\, w^\top x_i \geq 0 \\ -1 & if\, w^\top x_i < 0 \end{cases}}$$

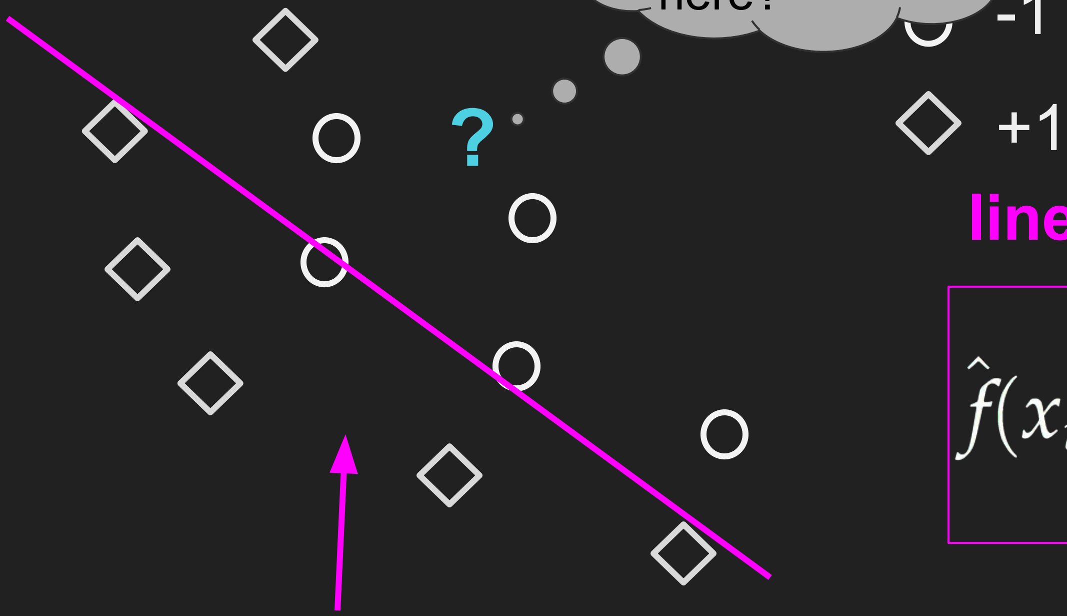
linear decision
boundary

$$\boxed{\mathcal{H}_w = \{x : w^\top x = 0\}}$$

Classification

What is our prediction here?

Which way is w pointing?



linear decision boundary

linear classifier:

$$\hat{f}(x_i) = \begin{cases} 1 & if\, w^\top x_i \geq 0 \\ -1 & if\, w^\top x_i < 0 \end{cases}$$

$$\mathcal{H}_w = \{x : w^\top x = 0\}$$

Individually or in small groups

3 Classification and Separability

In this problem, we will explore how adding the label as a feature affects separability of a dataset.

(a) Consider a dataset $\{x_i, y_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$, $y \in \{-1, 1\}$ which is *not* linearly separable.

Now for each point in the dataset, consider augmenting each feature vector with the label, so that we get a new feature vector $z_i \in \mathbb{R}^{d+1}$, where:

$$z_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

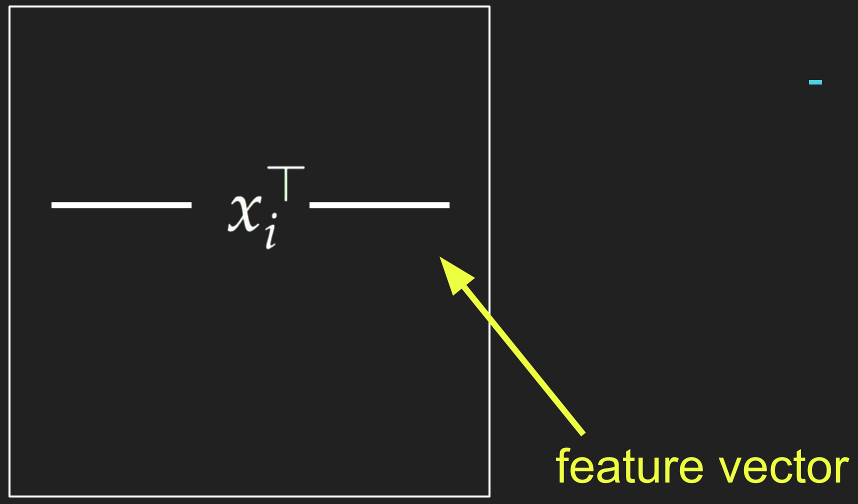
Show that using the augment feature vectors, the dataset is linearly separable.

(b) How will we expect the linear decision boundary defined above to perform when applied to a new unseen set of features $\mathbf{X}^{new} \in \mathbb{R}^{n_2 \times d}$?

Features

Feature matrix:

$$\mathbf{X} \in \mathbb{R}^{\mathbf{n} \times \mathbf{d}}$$



But we can **change** the features for each instance:

- If features not expressive enough, we can use feature mapping ϕ to **expand** features
  - Eg. polynomial features
- If too many features to fit in memory, or computation is hard, can **reduce dimensionality**

Kernels: ‘Lifting’ the features into higher dimensions

$$x_i, x_j \in \mathbb{R}^d$$

feature vectors

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

kernel function

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ \vdots & \ddots & & \vdots \\ & & & \\ k(x_n, x_1) & & \dots & k(x_n, x_n) \end{bmatrix}$$

Gram matrix

Kernels: ‘Lifting’ the features into higher dimensions

$$x_i, x_j \in \mathbb{R}^d$$

feature vectors

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

ke

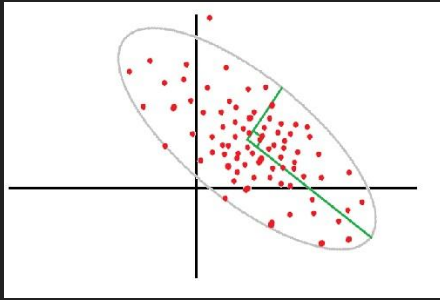
What are the **dimensions** of these quantities?

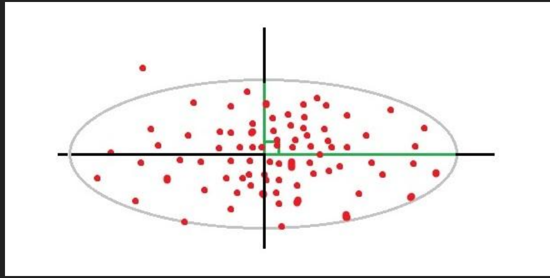
$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_n) \\ \vdots & \ddots & & \vdots \\ k(x_n, x_1) & & \dots & k(x_n, x_n) \end{bmatrix}$$

Gram matrix

Principal Component Analysis: PCA:

$$\mathbf{X} \in \mathbb{R}^{\mathbf{n \times d}} \qquad \mathbf{X_{PCA}} \in \mathbb{R}^{\mathbf{n \times k}}$$





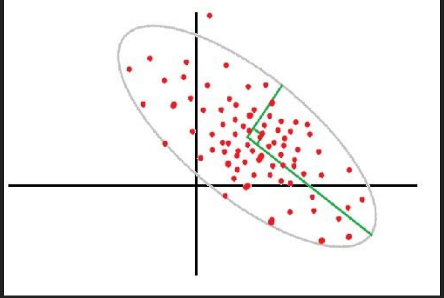
dimensionality **reduction**

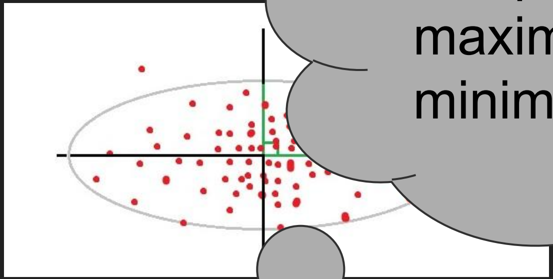
From SVD of X

$$X_{PCA} = U\mathrm{diag}(\sigma_1, ..., \sigma_k, 0, ...0)V^\top$$

Principal Component Analysis: PCA:

$\mathbf{X} \in \mathbb{R}^{\mathbf{n \times d}}$ $\qquad$ $\mathbf{X_{PCA}} \in$





Projection onto first k components: maximizes ______ and minimizes ______?

dimensionality **reduction**

From SVD of X

$$X_{PCA} = U\mathrm{diag}(\sigma_1, ..., \sigma_k, 0, ...0)V^\top$$

Canonical Correlation Analysis (CCA)

Maximizes **correlations** between the canonical variables

$$Cor(X, Y) = \rho(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}}$$

Canonical Correlation Analysis (CCA)

Maximizes **correlations** between the canonical variables

$$Cor(X, Y) = \rho(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}}$$

Explain a main difference between PCA and CCA. What would happen if we ran CCA with datasets (X,X) (the same dataset twice)?

Individually or in small groups (probably just (a))

In lectures, discussion, and homework, we learned how to use PCA to do dimensionality reduction by projecting the data to a subspace that captures most of the variability. This works well for data that is roughly Gaussian shaped, but many real-world high dimensional datasets have underlying low-dimensional structure that is not well captured by linear subspaces. However, when we lift the raw data into a higher-dimensional feature space by means of a nonlinear transformation, the underlying low-dimensional structure once again can manifest as an approximate subspace. Linear dimensionality reduction can then proceed. As we have seen in class so far, kernels are an alternate way to deal with these kinds of nonlinear patterns without having to explicitly deal with the augmented feature space. This problem asks you to discover how to apply the "kernel trick" to PCA.

Let $\mathbf{X} \in \mathbb{R}^{n\times\ell}$ be the data matrix, where n is the number of samples and ℓ is the dimension of the raw data. Namely, the data matrix contains the data points $\mathbf{x}_j \in \mathbb{R}^\ell$ as rows

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} \in \mathbb{R}^{n\times\ell}. \tag{7}$$

(a) (5 pts) **Compute $\mathbf{X}\mathbf{X}^\top$ in terms of the singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n\times n}$, $\mathbf{\Sigma} \in \mathbb{R}^{n\times\ell}$ and $\mathbf{V} \in \mathbb{R}^{\ell\times\ell}$.** Notice that $\mathbf{X}\mathbf{X}^\top$ is the matrix of pairwise Euclidean inner products for the data points. **How would you get $\mathbf{U}$ if you only had access to $\mathbf{X}\mathbf{X}^\top$?**

(b) (7 pts) Given a new test point $\mathbf{x}_{test} \in \mathbb{R}^\ell$, one central use of PCA is to compute the projection of $\mathbf{x}_{test}$ onto the subspace spanned by the k top singular vectors $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k$.

Express the scalar projection $z_j = \mathbf{v}_j^\top \mathbf{x}_{test}$ onto the j-th principal component as a function of the inner products

$$\mathbf{X}\mathbf{x}_{test} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_{test} \rangle \\ \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_{test} \rangle \end{pmatrix}. \tag{8}$$

Assume that all diagonal entries of $\mathbf{\Sigma}$ are nonzero and non-increasing, that is $\sigma_1 \geq \sigma_2 \geq \dots > 0$.

Hint: Express $\mathbf{V}^\top$ in terms of the singular values $\mathbf{\Sigma}$, the left singular vectors $\mathbf{U}$ and the data matrix $\mathbf{X}$. If you want to use the compact form of the SVD, feel free to do so.

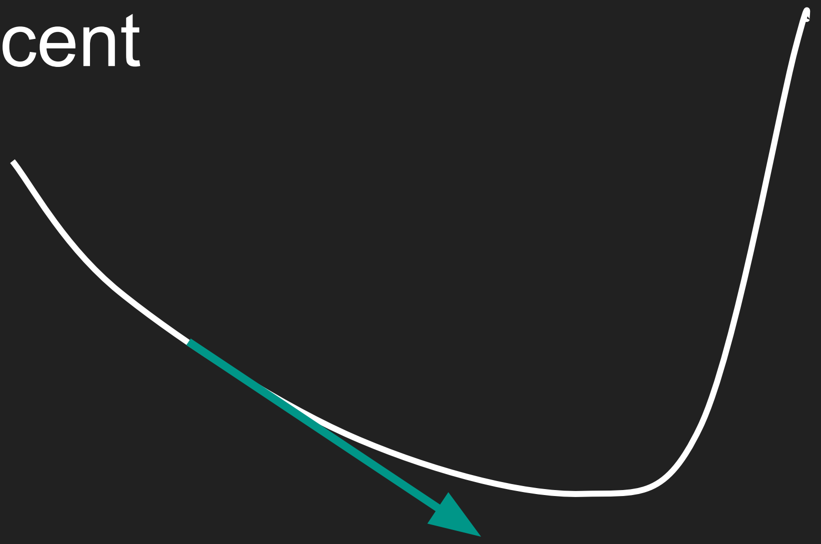
(c) (12 pts) How would you define kernelized PCA for a general kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ (to replace the Euclidean inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$)? For example, the RBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\delta^2}\right)$.

Describe this in terms of a procedure which takes as inputs the training data points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathbb{R}^\ell$ and the new test point $\mathbf{x}_{test} \in \mathbb{R}^\ell$, and outputs the analog of the previous part's z_j coordinate in the kernelized PCA setting. You should include how to compute $\mathbf{U}$ from the data, as well as how to compute the analog of $\mathbf{X}\mathbf{x}_{test}$ from the previous part.

Invoking the SVD or computing eigenvalues/eigenvectors is fine in your procedure, as long as it is clear what matrix is having its SVD or eigenvalues/eigenvectors computed. The kernel $k(\cdot, \cdot)$ can be used as a black-box function in your procedure as long as it is clear what arguments it is being given.

Optimization: Gradient Descent

$-\nabla f$ is a **descent direction**



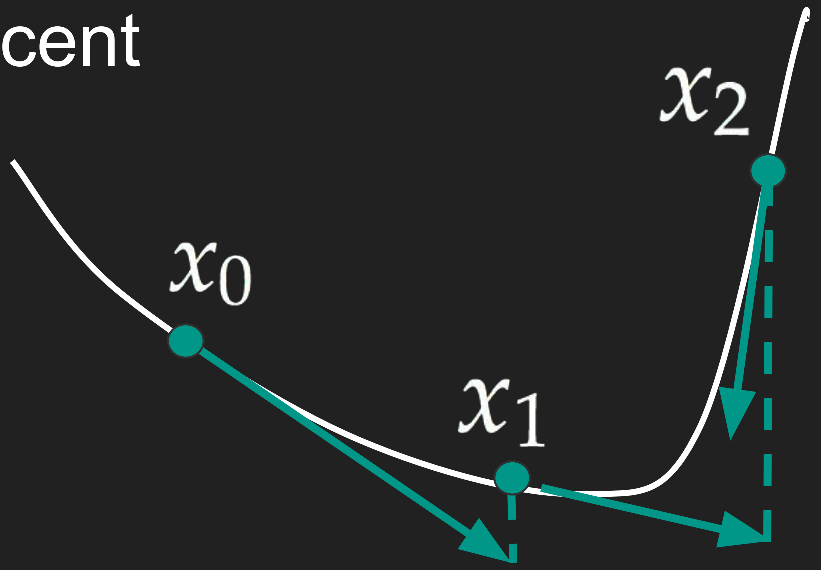
Optimization: Gradient Descent

-∇f is a **descent direction**

Gradient Descent:

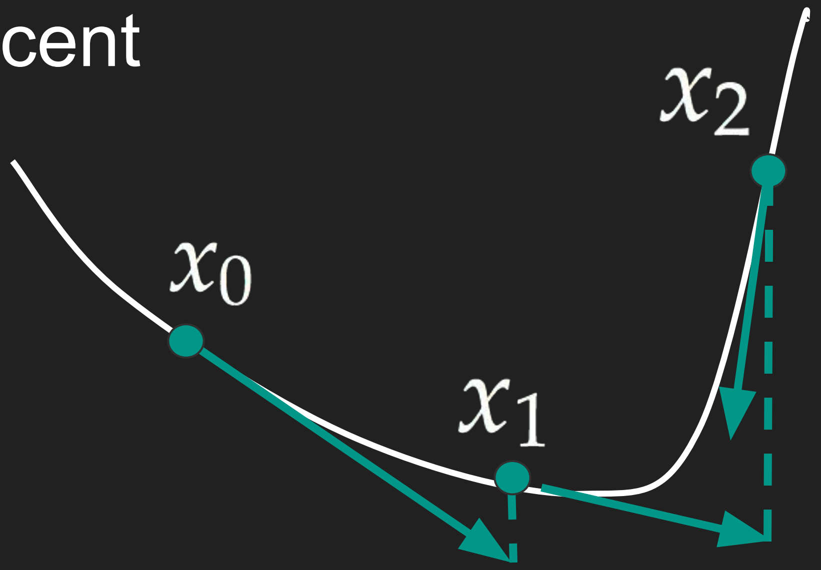
$$x_k = x_{k-1} - \alpha_k \nabla f(x_{k-1})$$

step size



Optimization: Gradient Descent

- ∇f is a **descent direction**



Gradient Descent:

$$x_k = x_{k-1} - \alpha_k \nabla f(x_{k-1})$$

step size

Does the step size matter?

Optimization: Stochastic Gradient Descent

Gradient Descent:

$$x_k = x_{k-1} - \alpha_k \nabla f(x_{k-1})$$

Stochastic Gradient Descent:

$$x_k = x_{k-1} - \alpha_k g(x_{k-1})$$

$$\mathbb{E}[g(x)] = \nabla f(x)$$

replace the gradient with a **random function**
whose **expectation** is the gradient

Optimization: Stochastic Gradient Descent

Gradient Descent:

$$x_k = x_{k-1} - \alpha_k \nabla f(x_{k-1})$$

When would we choose SGD over GD?

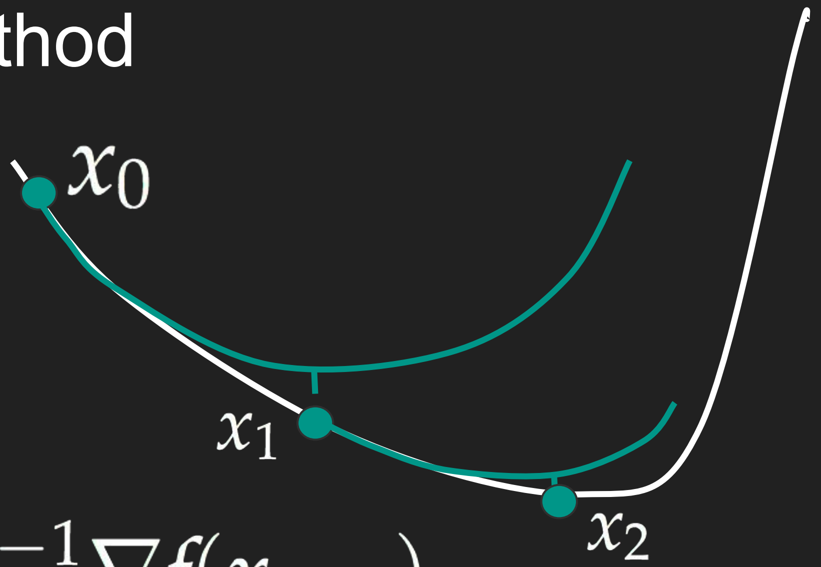
Stochastic Gradient Descent:

$$x_k = x_{k-1} - \alpha_k g(x_{k-1})$$

$$\mathbb{E}[g(x)] = \nabla f(x)$$

replace the gradient with a **random function**
whose **expectation** is the gradient

Optimization: Newton's Method



$$x_k = x_{k-1} - (\nabla^2 f(x_{k-1}))^{-1} \nabla f(x_{k-1})$$

second-order curvature information

Optimization: Newton's Method

$$x_k = x_{k-1} - (\nabla^2 f(x_{k-1}))^{-1} \nabla f(x_{k-1})$$

second-order curvature information

What if f(x) is quadratic?

Individually or in small groups:

1. Gradient Descent and Convexity

Here is the **Smoothed version of the Hinge loss function** (with parameter t):

$$f(y) = \begin{cases} \frac{1}{2} - ty & \text{if } ty \leq 0, \\ \frac{1}{2}(1-ty)^2 & \text{if } 0 < ty < 1, \\ 0 & \text{if } 1 \leq ty \end{cases}$$

Given $x_1, x_2, \ldots, x_n$ data points and $y_1, y_2, \ldots, y_n$ labels. we define the optimization problems as follows:

$$\min_w \frac{1}{n} \sum_{i=1}^{n} f(w^\top x_i - y_i)$$

Define: $L(w) = \frac{1}{n} \sum_{i=1}^{n} f(w^\top x_i - y_i)$

- Is $L(w)$ convex?
- Write out the Gradient Descent update equation.
- Write out the Stochastic Gradient Descent update equation.

Extra Questions

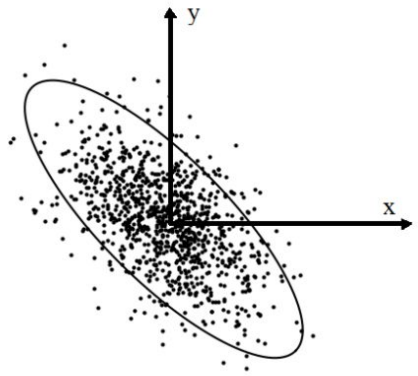
If there’s time, here’s some more practice problems.

PCA

(4 points) Suppose that we observe the position and velocity of an object moving **along a line** in 3D space. At any point on the line, the object can have any speed. Our position observations measure the x, y, and z coordinates of the object, and the velocity observations measure the x, y, and z components of the velocity. We collect a large set of observations and run PCA on the set. **How many principal components would we expect to use to represent this data set?**

PCA

Select which covariance matrix was most likely used to generate the following multivariate Gaussian distribution



where the positive x direction is to the right and the positive y direction is up.

- ○ $\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$
- ○ $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$
- ○ $\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$
- ○ $\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$

Data Hygiene

- What is the difference between the holdout method, cross-validation, and leave one out?
- Why would we use these methods?
- Write pseudocode for each method.