Machine Learning I

Lecture 2 - Matrix Differentiation and Optimization

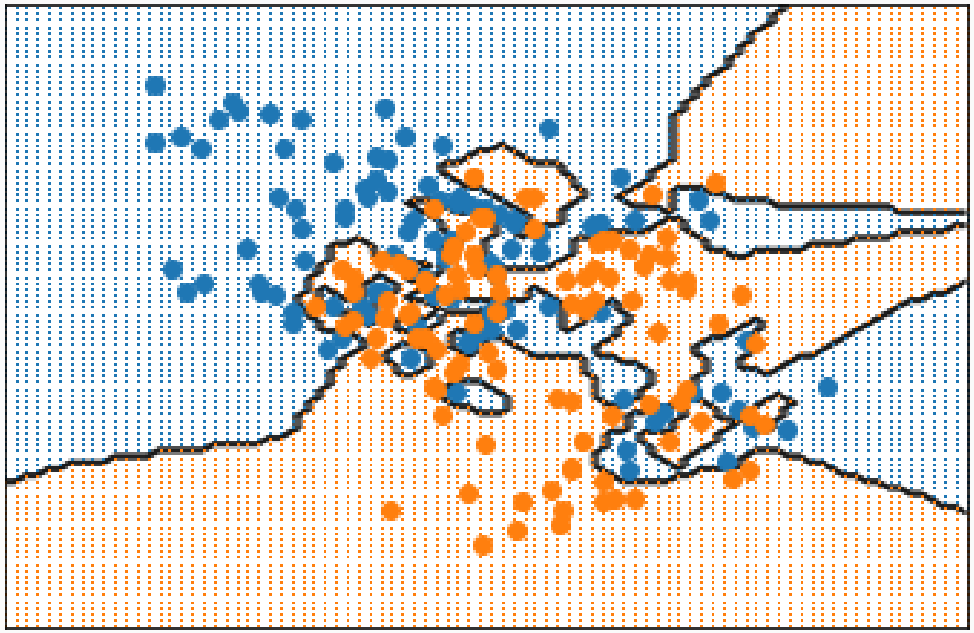
Nathaniel Bade

Northeastern University Department of Mathematics

Table of contents

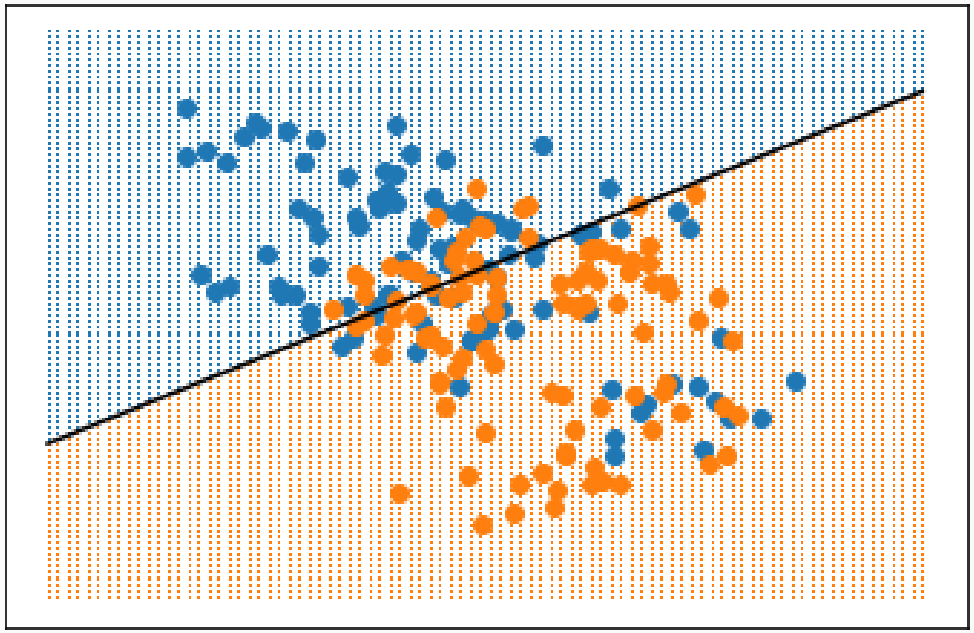
Bias-Variance Trade Off

Bias-Variance Trade Off



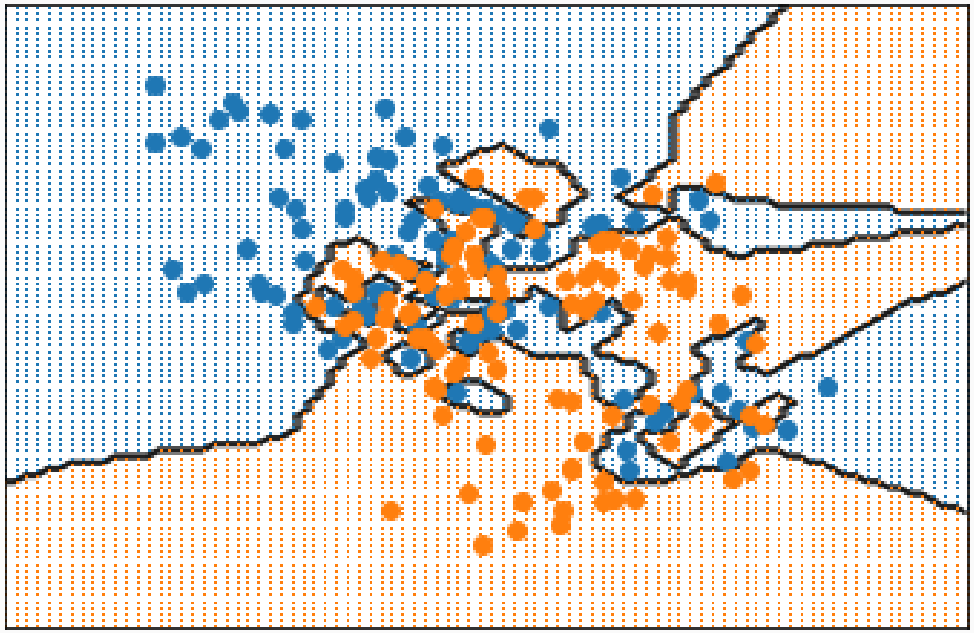
Last time we discussed two methods of performing binary classification: the first was to label each point by the vote of its k-nearest neighbors. The second was to perform linear regression on $Y = P(y_i = \text{Orange})$, that is the probability that the i'th training point is orange.

Bias-Variance Trade Off



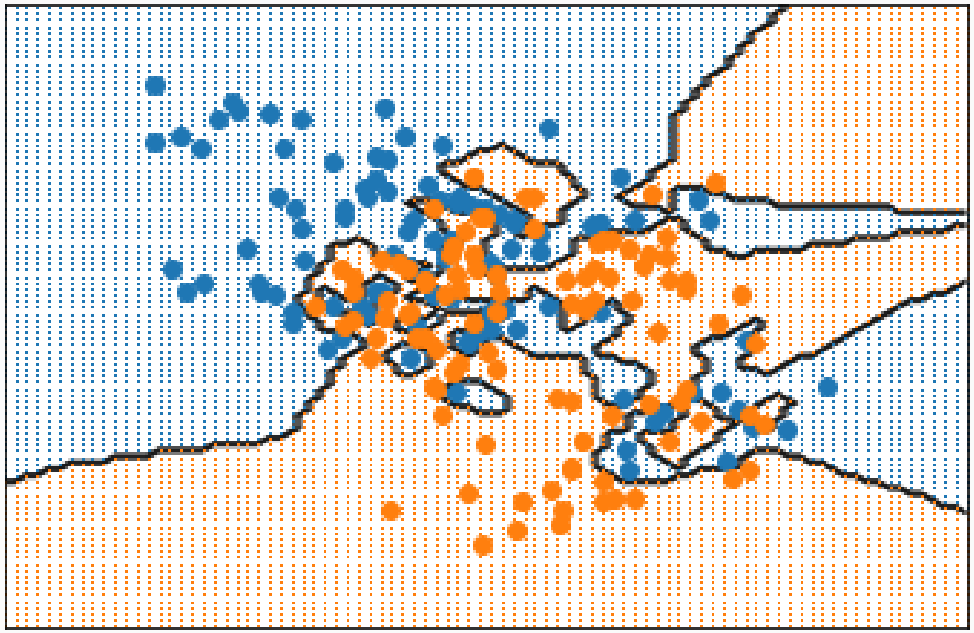
Last time we discussed two methods of performing binary classification: the first was to label each point by the vote of its k-nearest neighbors. The second was to perform linear regression on $Y = P(y_i = \text{Orange})$, that is the probability that the i'th training point is orange.

Bias-Variance Trade Off



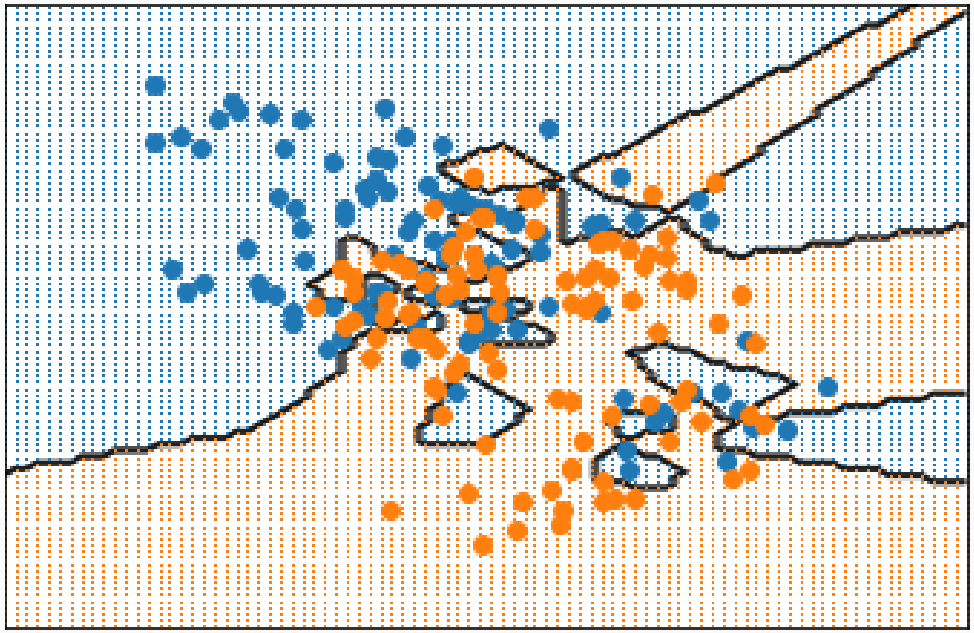
There is another disadvantage to using 1 nearest neighbors: it is very sensitive to the choice of training data.

Bias-Variance Trade Off



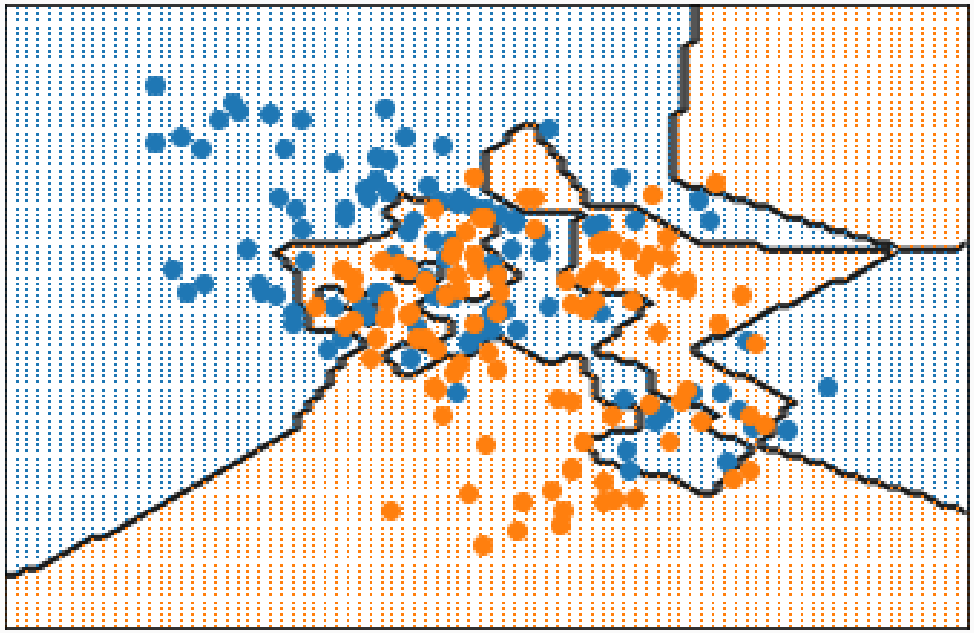
Taking two samples of the training data, 1NN produces two drastically different regressions, especially in the region with the most points.

Bias-Variance Trade Off



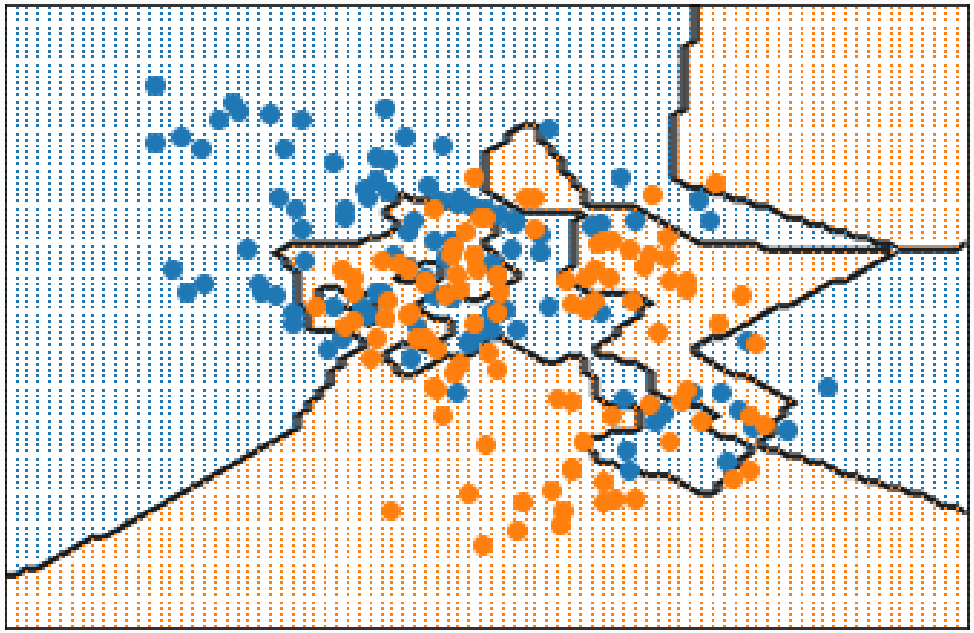
Taking two samples of the training data, 1NN produces two drastically different regressions, especially in the region with the most points.

Bias-Variance Trade Off



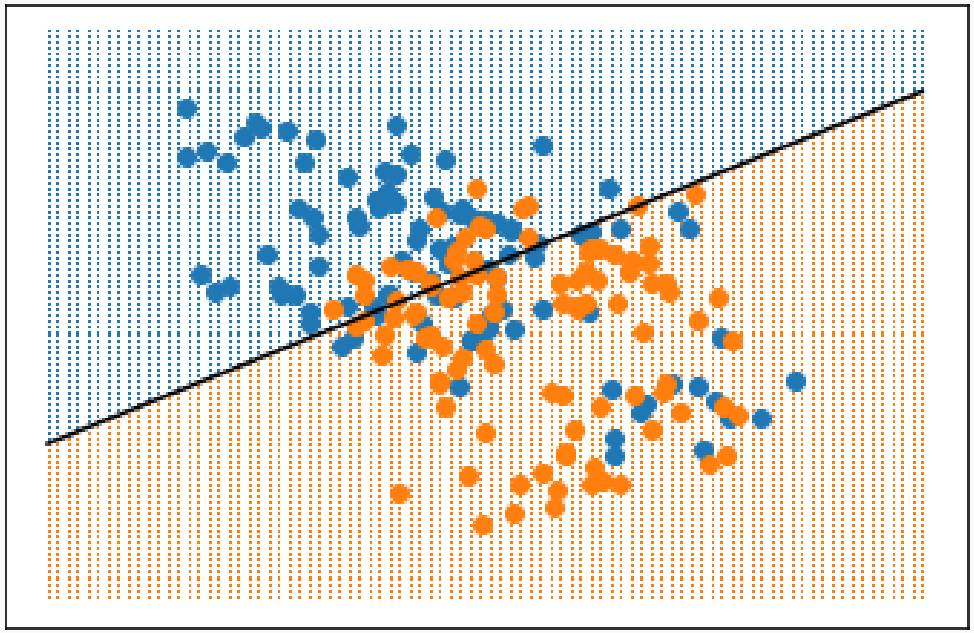
Taking two samples of the training data, 1NN produces two drastically different regressions, especially in the region with the most points.

Bias-Variance Trade Off



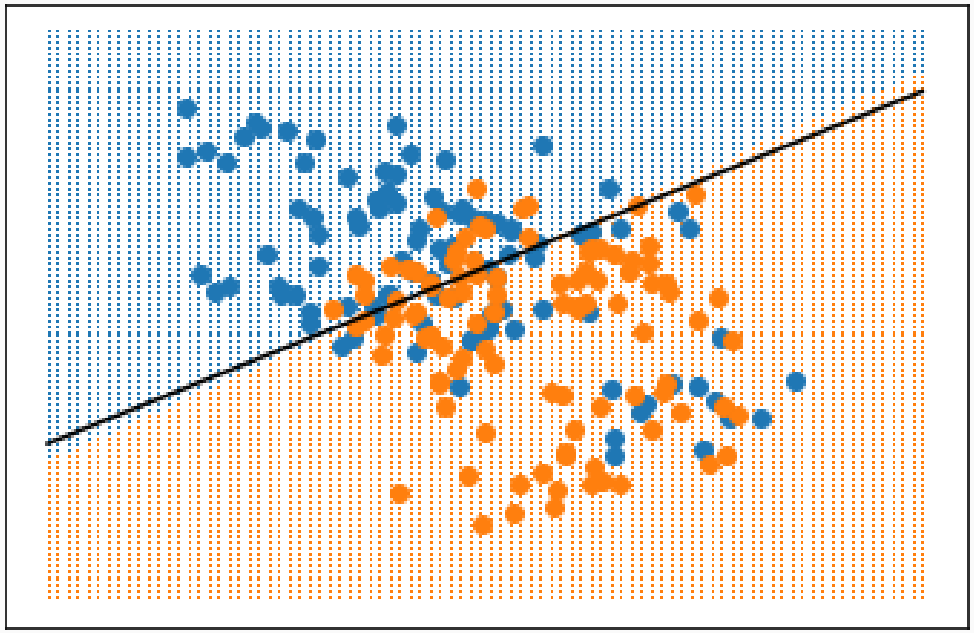
This sensitivity to training data is known as the variance of the model.

Bias-Variance Trade Off



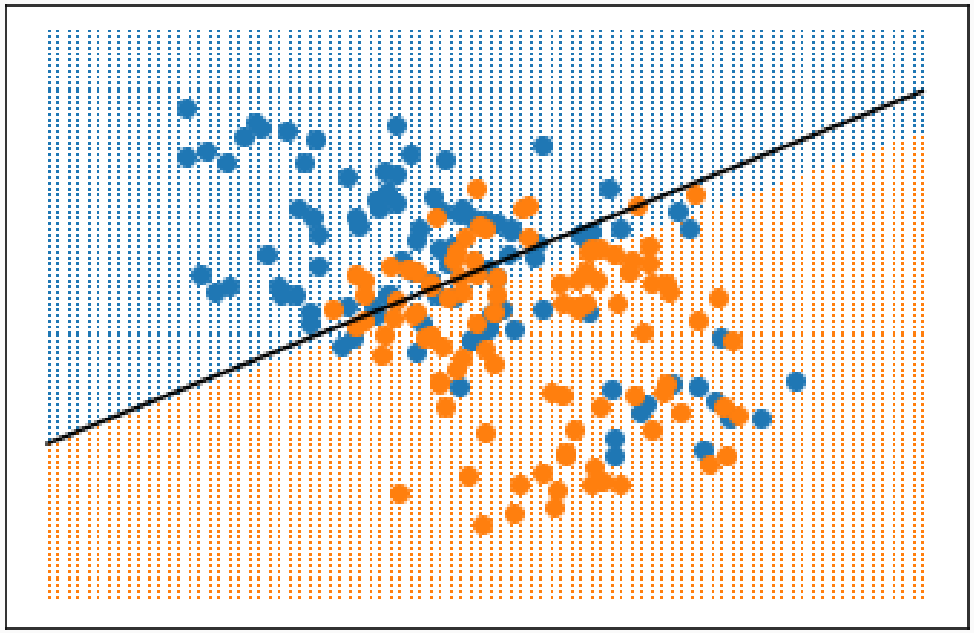
By contrast, if we take the same half of the training data and recompute the linear classifier, the change in the fit is very low, especially in the region with the most data points.

Bias-Variance Trade Off



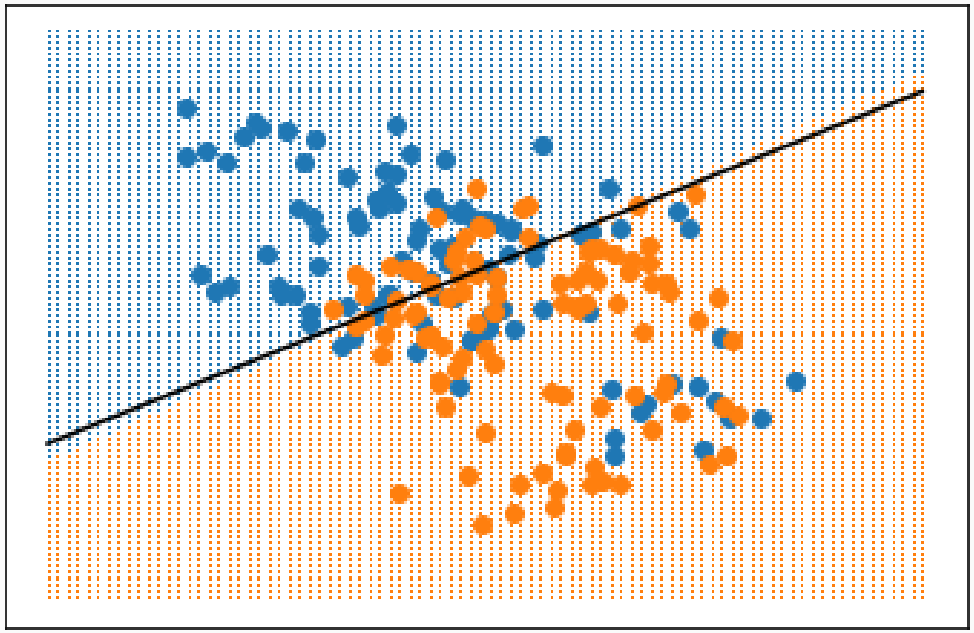
By contrast, if we take the same half of the training data and recompute the linear classifier, the change in the fit is very low, especially in the region with the most data points.

Bias-Variance Trade Off



By contrast, if we take the same half of the training data and recompute the linear classifier, the change in the fit is very low, especially in the region with the most data points.

Bias-Variance Trade Off

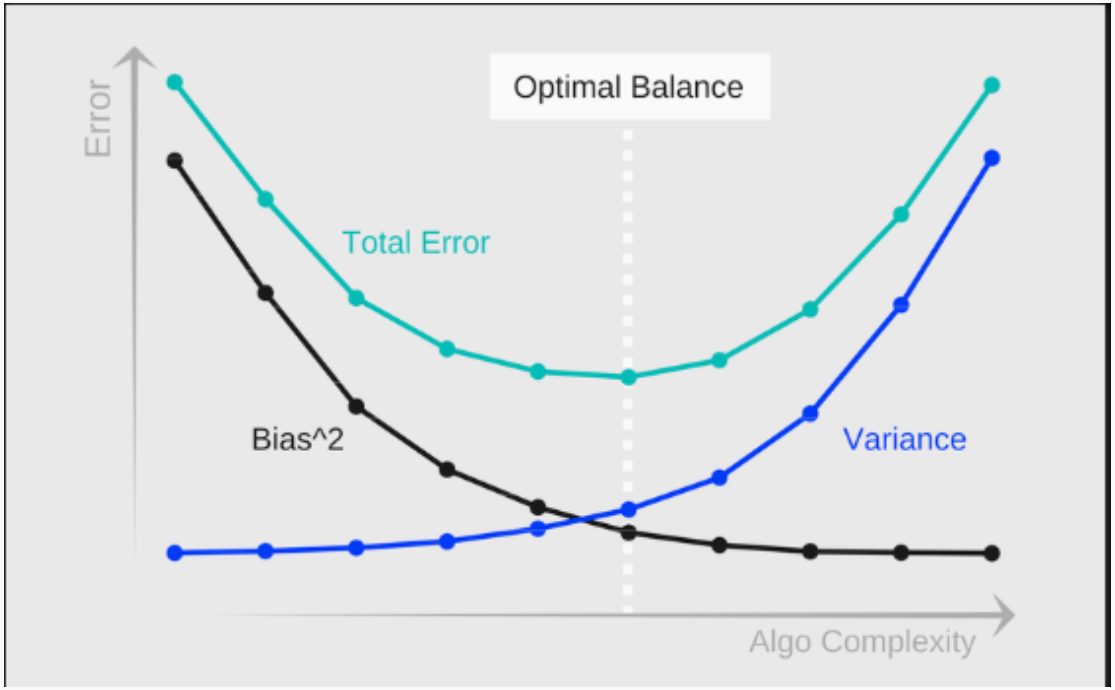


This contrast between the two algorithms is know as the **bias-variance trade off**.

For a class of models, the **bias** roughly is the expected error of the "best" classifier in the model class given a random set of training data.

The **variance** is roughly the sensitivity of the model to the training data.

Bias-Variance Trade Off

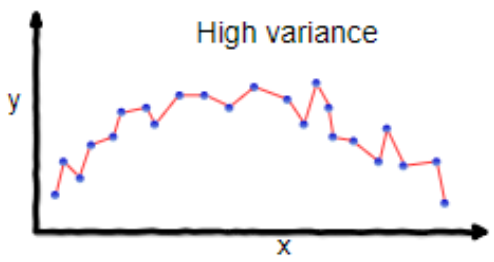


The idea is that the total error Err can be split into three parts:

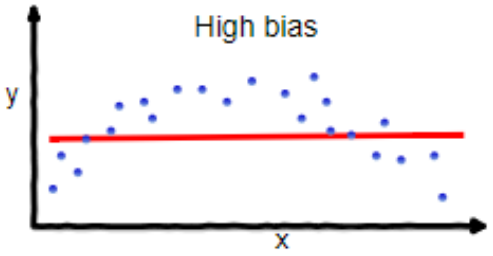
$$\text{Err} = (\text{Bias})^2 + \text{Var} + \text{Irreducible Error}$$

The best fit lies somewhere in between the extreme ends.

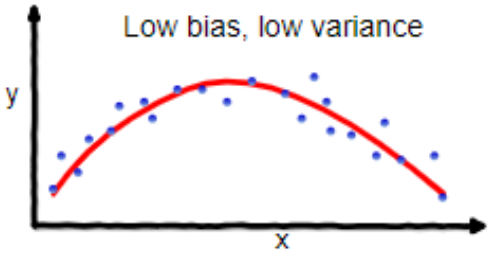
Bias-Variance Trade Off



overfitting



underfitting



Good balance

The idea is that the total error Err can be split into three parts:

$$\text{Err} = (\text{Bias})^2 + \text{Var} + \text{Irreducible Error}$$

The best fit lies somewhere in between the extreme ends. We will return and make this rigorous at the end of the semester.

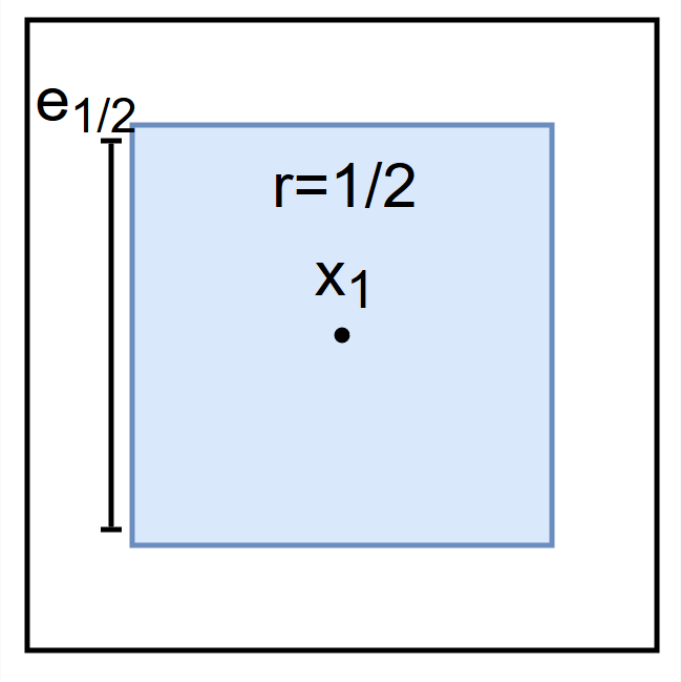
The Curse of Dimensionality

The Curse of Dimensionality

Of our two models, we found that the linear model was stable, but very biased. On the other, the k-nearest neighbors is unstable but much less biased. Since stability can be overcome with a large enough data set, why don't we always use k-nearest neighbors?

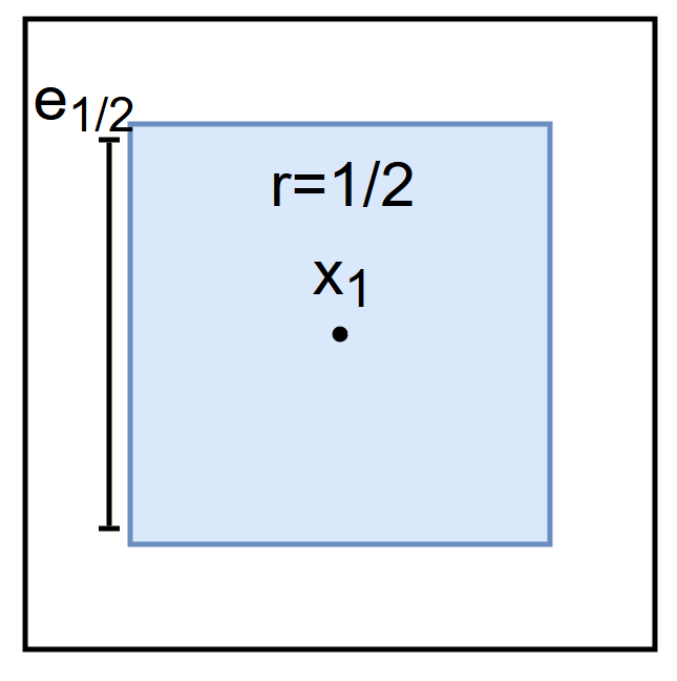
The first reason is that for a sufficient amount of data of high enough dimension it may be computationally hard to *find* the k-nearest neighbors, but in practice this is usually surmountable. The real reason is that geometry in higher dimensions often behaves counter-intuitively, and as the dimension of our input vector gets large,the meaning of "nearest" becomes less clear.

The Curse of Dimensionality



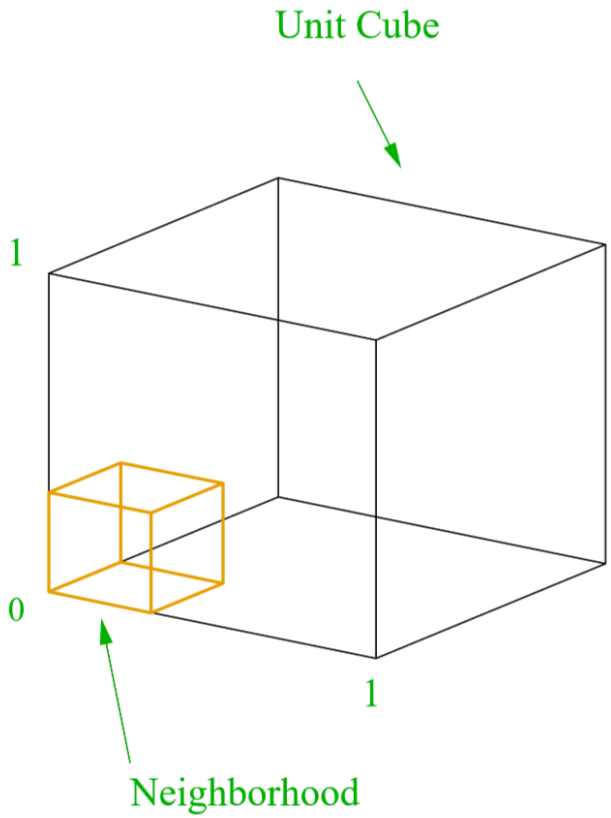
Example: Assume a p dimensional input with uniformly distributed datapoints in the unit cube. For a cubical neighbor around x_i to capture a proportion r of the data points, it will have to cover a proportion r of the volume. For edges of length e_r, this means $(e_r)^p = r$, or $e = r^{\frac{1}{p}}$.

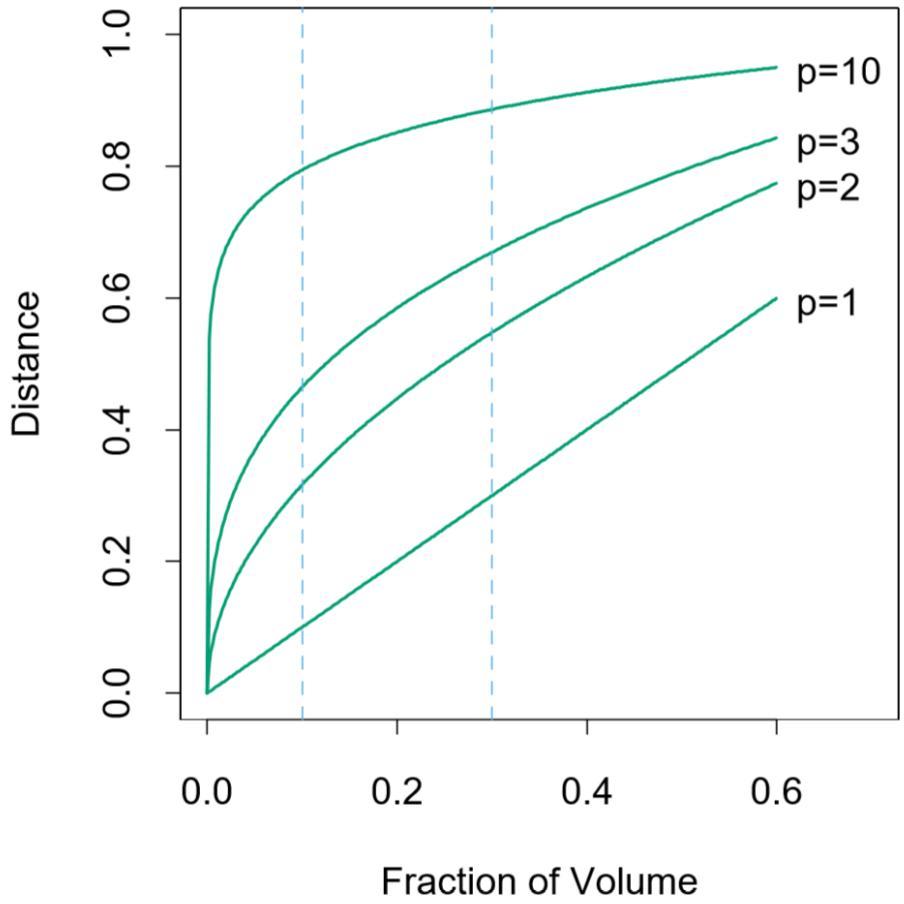
The Curse of Dimensionality



In 10 dimensions $e_{.01} = .63$, so capturing 1% of the data requires covering over half the range for each input. It's not clear that this is now a "local" average we're taking.

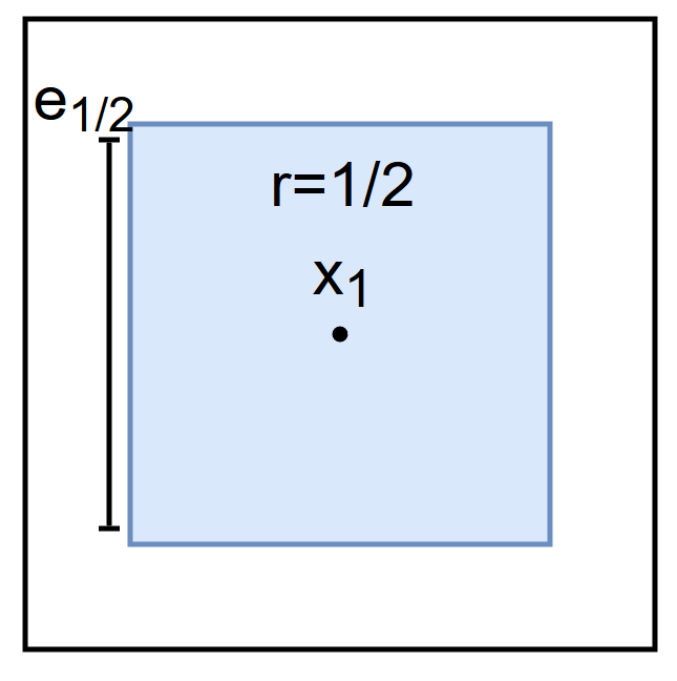
The Curse of Dimensionality





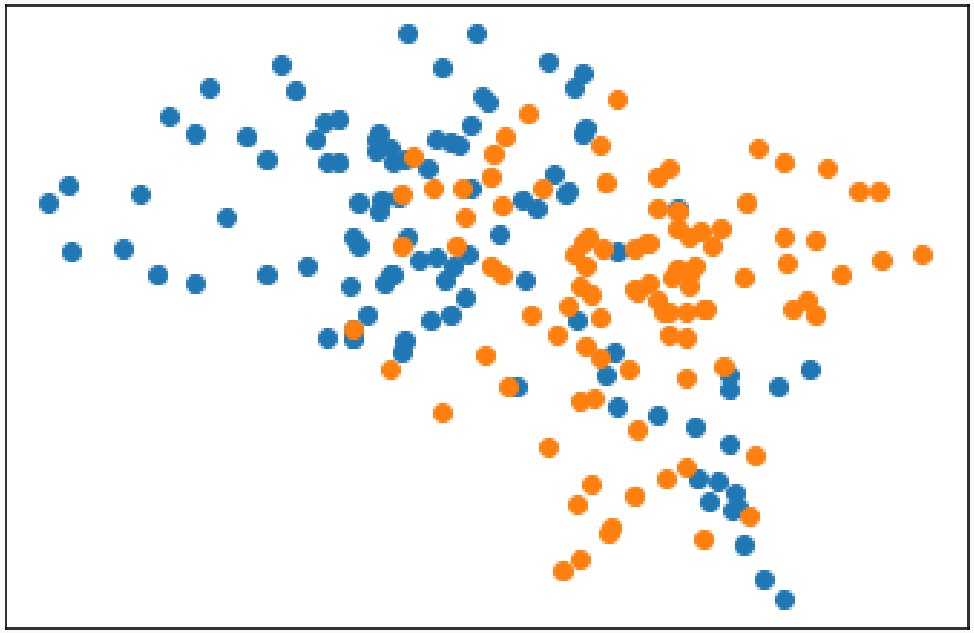
Indeed, we see that as p increases, length of a side required to probe the same fraction of volume increases drastically.

The Curse of Dimensionality



What about 1000 dimensions (still small for something like photo classification). Here, $e_{.01} = .9954$ so we would expect a cube of with side lengths .9954 to only capture 1% of data points uniformly distributed in a unit cube.

The Curse of Dimensionality



In low dimensions, objects tend to be clustered towards the center of the range. In high dimensions, the generic position for uniformly distributed data point is close to the boundary.

The Curse of Dimensionality

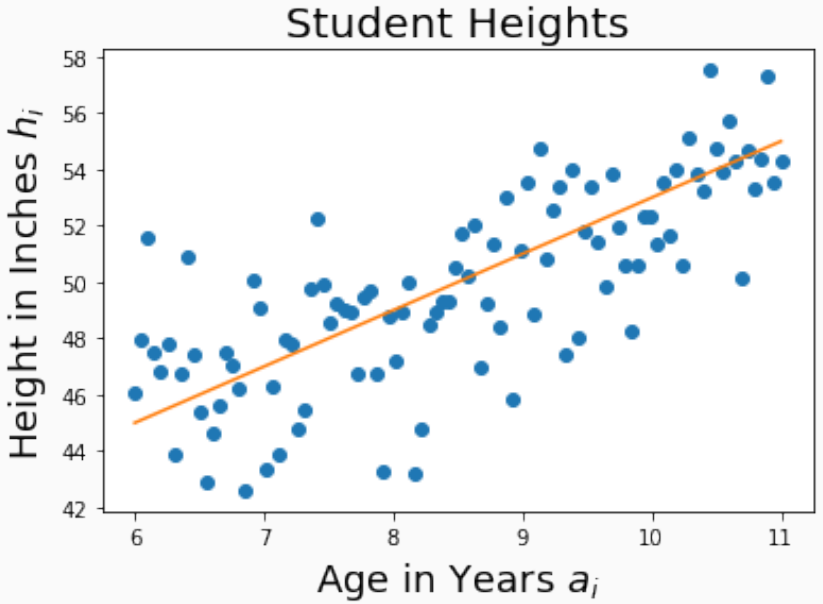
For another face of the curse, lets consider the relative volume of the unit sphere in high dimensions. In $2k$ dimensional space, the volume of the sphere of radius r is

$$V(r) = \frac{\pi^k}{k!} r^{2k}.$$

For $r = 1$ and $k = 500$, we have $V(r) \approx 3.1 \times 10^{-886}$. So the largest sphere contained in the domain contains almost 0% of the data points.

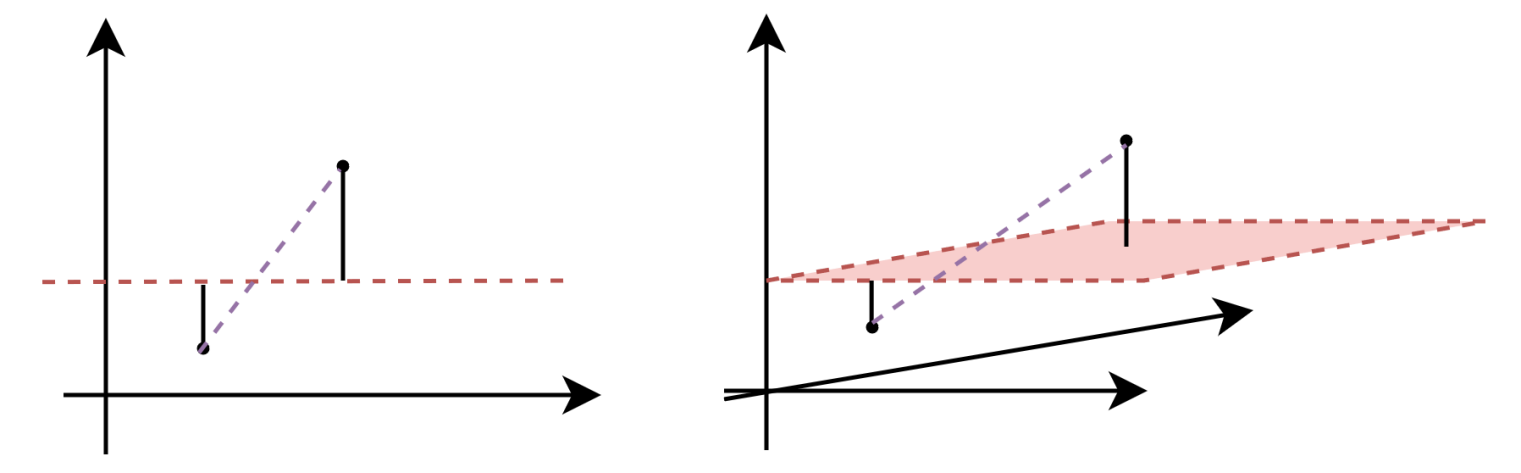
Inverting the formula, we find we can capture 1% of the data points in a sphere of radius 7.6, but this means that even capturing 1% of our data requires looking at a distance far outside of our domain.

The Curse of Dimensionality



It would be ridiculous to extrapolate our result from the height dataset out to $h = 20$, k-nearest neighbors in high dimensions would be theoretically probing that same space.

The Curse of Dimensionality



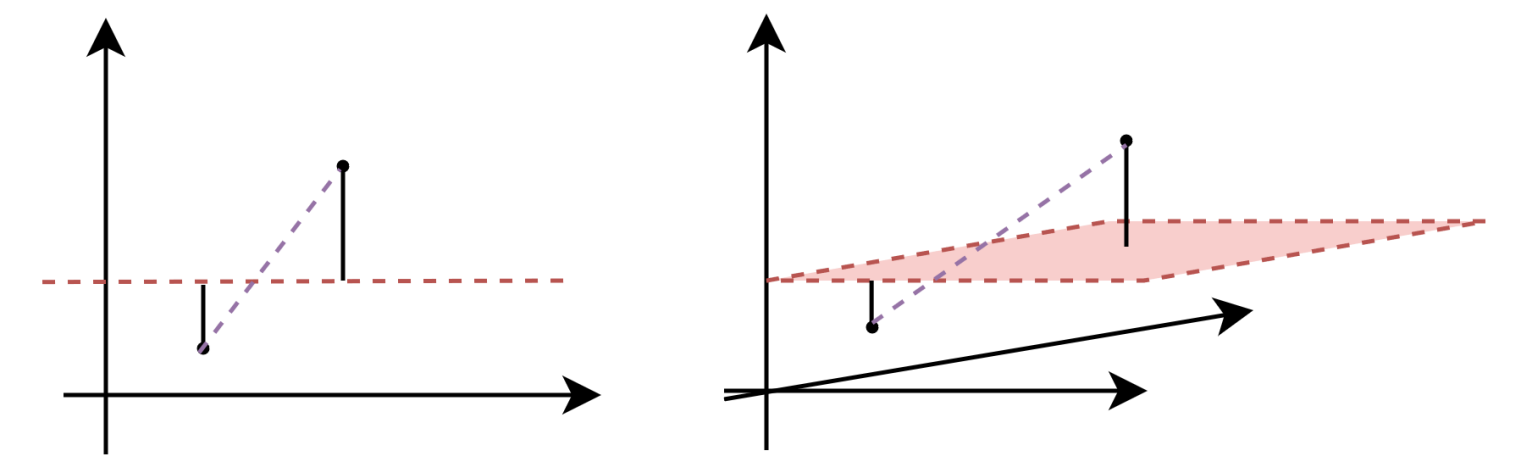
A final example is that given two random data points in $[0,1]^2$, the distance between datapoints is

$$\sqrt{\Delta X_1^2 + \Delta X_2^2}\,.$$

In 3d, this is at least as large:

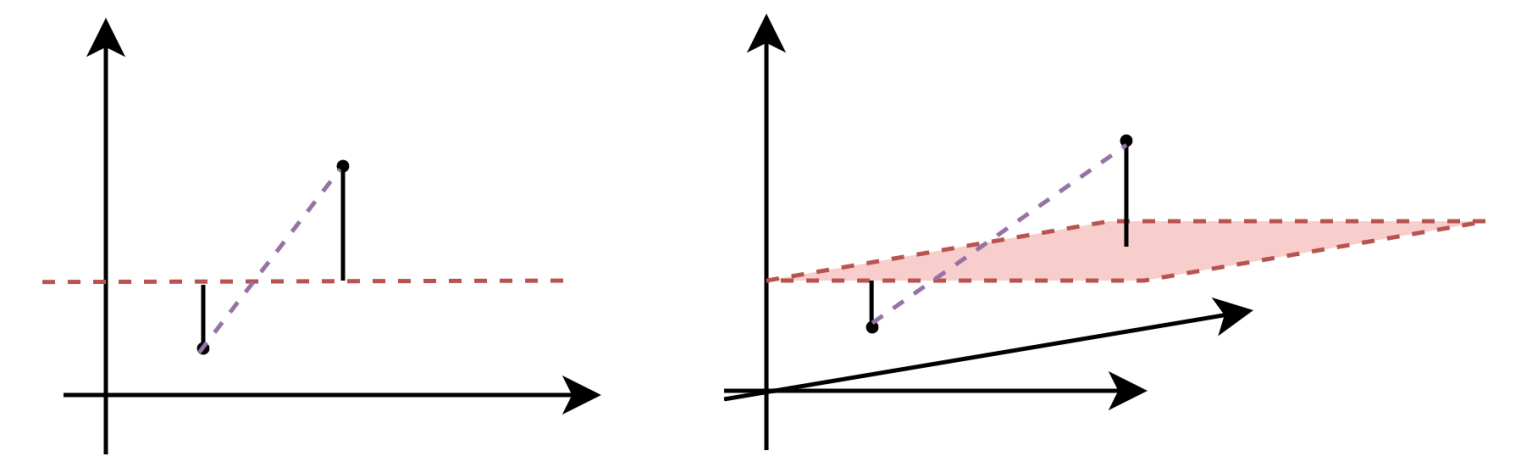
$$\sqrt{\Delta X_1^2 + \Delta X_2^2 + \Delta X_3^2}\,.$$

The Curse of Dimensionality



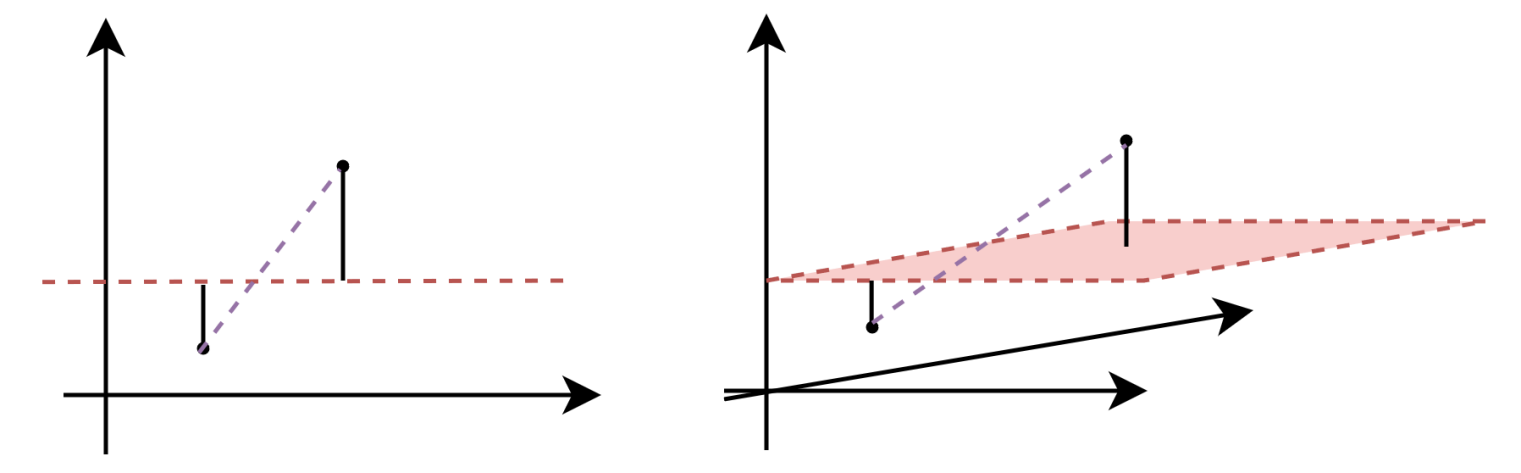
By contrast, the distance to a hyperplane remains unchanged. Practically, this means that as the dimension rises, fewer and fewer points will "close," possibly violating the k-NN assumption that similar points share similar labels.

The Curse of Dimensionality



As distances become large in high dimensional space the relative distances to a hyperplane becomes small. This can be seen as an advantage of linear classifiers since points can be meaning distinct, but it also may be a disadvantage, since a small perturbation can lead to a relatively large change in the classification.

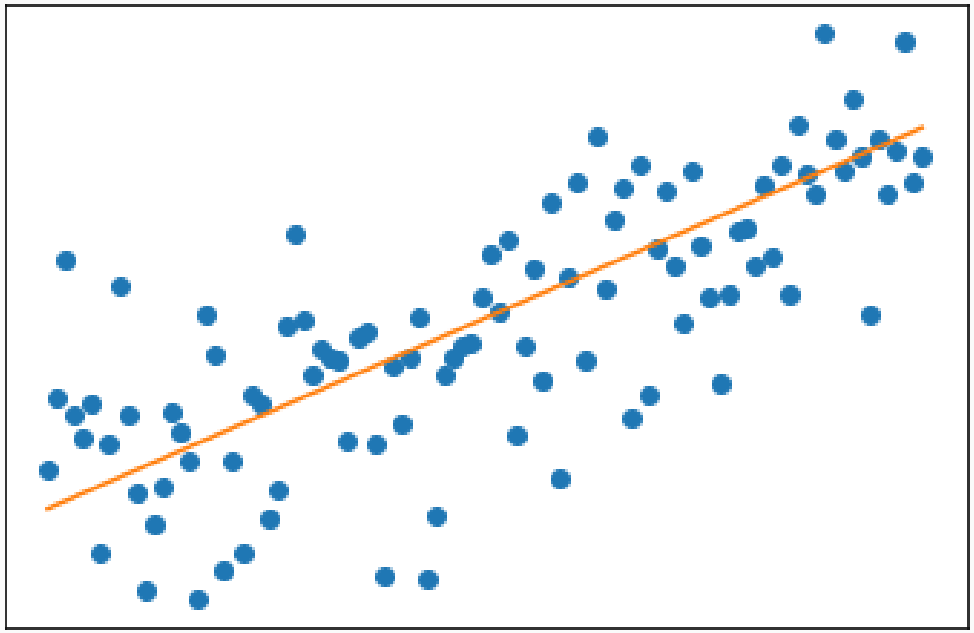
The Curse of Dimensionality



That said, by relying on rigid assumptions, the linear model has very little bias and is quite stable even for large dimensional data sets, whereas k-nearest neighbors is much less stable. But if the assumptions are wrong all conclusions may be incorrect. We will see many examples between the extremes of k-NN and linear models.

Matrix Differentiation

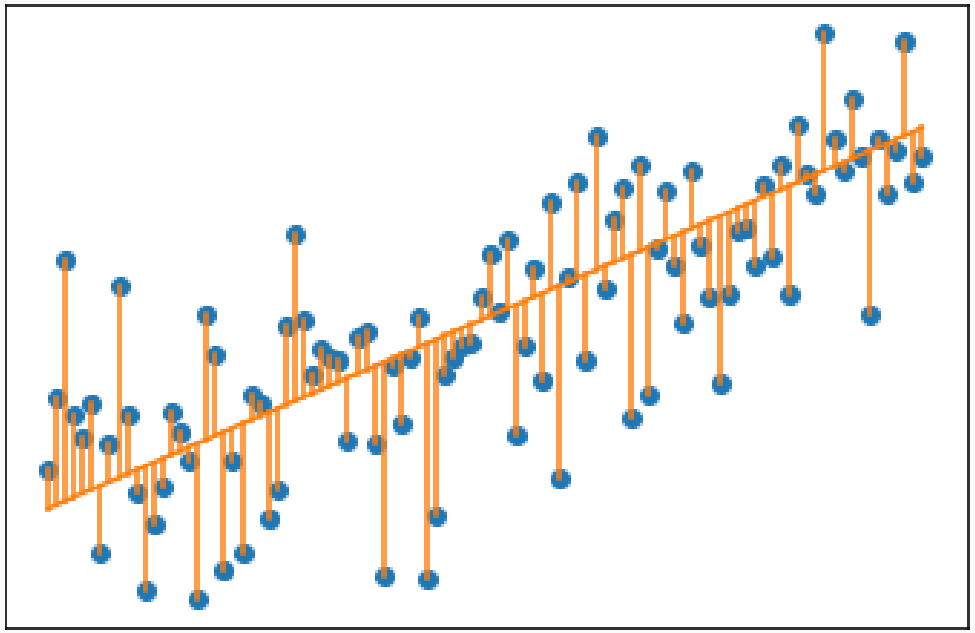
Linear Regression



Last time, we discuss the problem of fitting a linear function to a set of datapoints with domain $X \in \mathbb{R}^p$ and labels $Y \in \mathbb{R}$. We noted that by redefining $X = [X_1, \ldots, X_p]$ to $X = [1, X_1, \ldots, X_p]$, the linear function can be compactly written:

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p = \beta^T X.$$

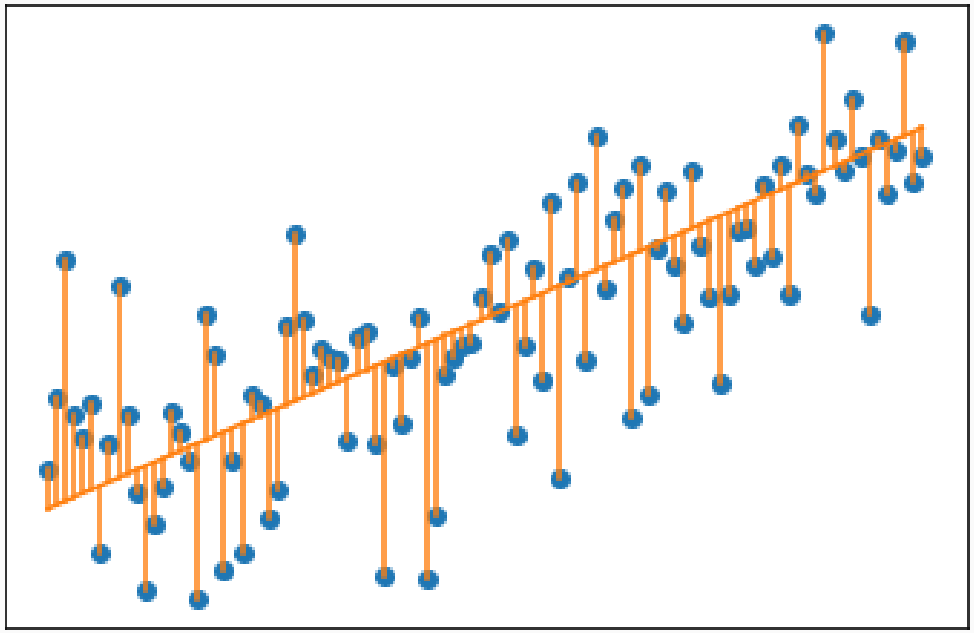
Linear Regression



We can then fit a linear function to a set of datapoint with domain $\mathbb{R}^P$ and labels $\mathbb{R}$ by finding β that minimizes the residual sum squared

$$\text{RSS}(\beta) = \sum_{i=1}^{N} (y_i - \beta^T x_i)^2 \,.$$

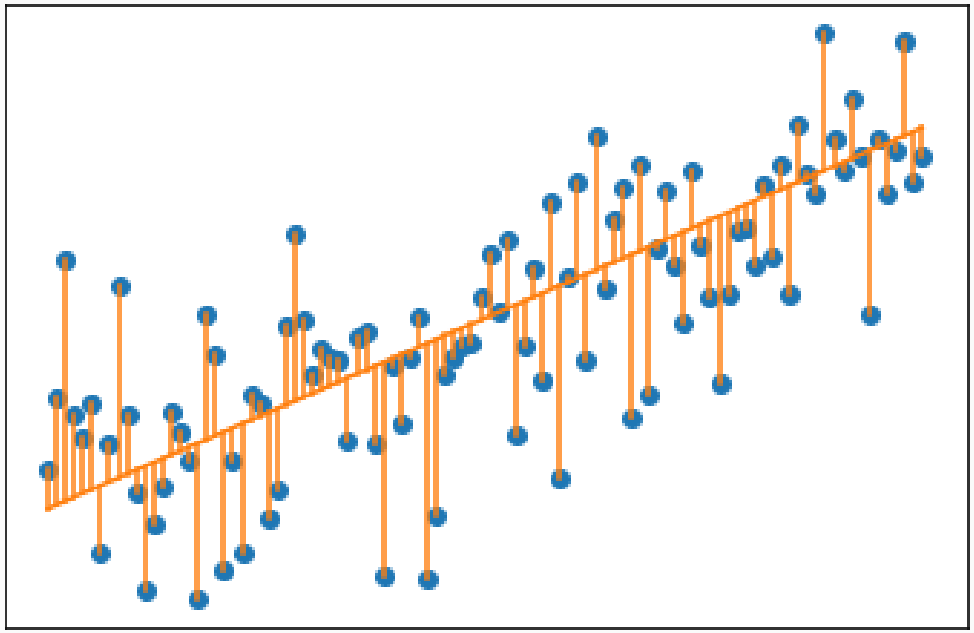
Linear Regression



We then showed that we could rewrite the RSS as a matrix multiplication, and noted that in matrix form the formula holds for $Y \in \mathbb{R}^k$, β a $p \times K$ matrix:

$$\text{RSS}(\beta) = \sum_{i=1}^{N} (y_i - \beta^T x_i)^2 = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$

Linear Regression



Finding an expression for β that minimizes RSS(β) will solve a huge family of regression problems. Using matrix derivatives we can show the solution takes on a particularly nice form:

$$\frac{\partial \text{RSS}(\beta)}{\partial \beta} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0$$

Matrix Derivatives

There are two formats for matrix derivatives. For $Y \in \mathbb{R}^m$, $X \in \mathbb{R}^n$,

Numerator Layout Notation:

$$\frac{\partial Y}{\partial X} = \begin{bmatrix} \frac{\partial Y_1}{\partial X_1} & \cdots & \frac{\partial Y_1}{\partial X_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial Y_m}{\partial X_1} & \cdots & \frac{\partial Y_m}{\partial X_n} \end{bmatrix}$$

For $Y \in \mathbb{R}$,

$$\nabla Y = \frac{\partial Y}{\partial X} = \begin{bmatrix} \frac{\partial Y_1}{\partial X_1} & \cdots & \frac{\partial Y_1}{\partial X_n} \end{bmatrix}.$$

Denominator Layout Notation:

$$\frac{\partial Y}{\partial X} = \begin{bmatrix} \frac{\partial Y_1}{\partial X_1} & \cdots & \frac{\partial Y_m}{\partial X_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial Y_1}{\partial X_n} & \cdots & \frac{\partial Y_m}{\partial X_n} \end{bmatrix}.$$

For $Y \in \mathbb{R}$,

$$\nabla Y = \frac{\partial Y}{\partial X} = \begin{bmatrix} \frac{\partial Y_1}{\partial X_1} & \cdots & \frac{\partial Y_1}{\partial X_n} \end{bmatrix}^T.$$

Matrix Derivatives

Similarly, for a scalar $Y \in \mathbb{R}$ and a matrix $X \in \mathbb{R}^{m \times n}$,

Numerator Layout Notation:

$$\frac{\partial Y}{\partial X} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \cdots & \frac{\partial y}{\partial x_{m1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{1n}} & \cdots & \frac{\partial y}{\partial x_{mn}} \end{bmatrix}$$

Denominator Layout Notation:

$$\frac{\partial Y}{\partial X} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \cdots & \frac{\partial y}{\partial x_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{m1}} & \cdots & \frac{\partial y}{\partial x_{mn}} \end{bmatrix}.$$

Both notations are use, often without specifying. Sometime authors even switch back and forth so always check the matrix dimensions. We will exclusively use **Denominator Layout Notation** so that ∇Y is naturally a column vector.

Matrix Derivatives

In the following, we will break with convention temporarily and denote by $y, x \in \mathbb{R}$ scalars, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{A}$ an $m \times n$ matrix that does not depend on other variables. Denote the components by $\mathbf{x}_j$ and $\mathbf{A}_{ij}$.

P1: Let $y = \mathbf{b}^\mathsf{T}\mathbf{x}$. Then $\frac{\partial y}{\partial \mathbf{x}} = \mathbf{b}$.

Proof:

$$\frac{\partial y}{\partial \mathbf{x}_j} = \frac{\partial}{\partial \mathbf{x}_j}\left(\sum_k \mathbf{b}_k \mathbf{x}_k\right) = \mathbf{b}_j \,.$$

Since by convention $\frac{\partial y}{\partial \mathbf{x}}$ is a column vector whose i'th component is $\mathbf{b}_i$, $\frac{\partial y}{\partial \mathbf{x}} = \mathbf{b}$.

Matrix Derivatives

P2: Let $\mathbf{y} = \mathbf{A}\mathbf{x}$. Then $\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}^T$.

Proof:

$$\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_j} = \frac{\partial}{\partial \mathbf{x}_j}\left(\sum_k \mathbf{A}_{ik}\mathbf{x}_k\right) = \mathbf{A}_{ij}$$

Since by convention $\frac{\partial}{\partial \mathbf{x}_j}(\mathbf{y}_i)$ is a matrix whose ji'th component is a_{ij}, so $\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \mathbf{A}^T$.

P3: Let $y = \mathbf{b}^T\mathbf{A}\mathbf{x}$. Then $\frac{\partial y}{\partial \mathbf{x}} = \mathbf{A}^T\mathbf{b}$.

Proof: Let $\mathbf{w} = \mathbf{A}^T\mathbf{b}$, so that $y = \mathbf{w}^T\mathbf{x}$. This follows by (**P1**).

Matrix Derivatives

P4: Let $y = \mathbf{x}^T\mathbf{A}\mathbf{x}$, for $\mathbf{A}$ a square matrix. Then $\frac{\partial y}{\partial \mathbf{x}} = (\mathbf{A}^T + \mathbf{A})\mathbf{x}$.

Proof: By definition,

$$\frac{\partial y}{\partial \mathbf{x}_j} = \frac{\partial}{\partial \mathbf{x}_j}\left(\sum_{i,k} \mathbf{A}_{i,k}\mathbf{x}_i\mathbf{x}_k\right) = \sum_i \mathbf{A}_{i,j}\mathbf{x}_i + \sum_k \mathbf{A}_{j,k}\mathbf{x}_k\,.$$

So $\frac{\partial y}{\partial \mathbf{x}}$ is a column vector whose j'th component is given by the RHS. The column vector whose j'th component is $\sum_k \mathbf{A}_{j,k}\mathbf{x}_k$ is $\mathbf{A}\mathbf{x}$.

Similarly, $\mathbf{x}^T\mathbf{A}$ is a row vector whose j'th component is $\sum_i \mathbf{A}_{i,j}\mathbf{x}_i$, so $\mathbf{A}^T\mathbf{x}$ is the similarly structured column vector.

Note that if $\mathbf{A}$ is symmetric than $\frac{\partial y}{\partial \mathbf{x}_j} = 2\mathbf{A}^T\mathbf{x}$.

Matrix Derivatives

P5: Let $a = \mathbf{y}^T\mathbf{x}$, where both $\mathbf{x}$ and $\mathbf{y}$ are functions of a vector $\mathbf{z}$. Then

$$\frac{\partial a}{\partial \mathbf{z}} = \frac{\partial \mathbf{y}}{\partial \mathbf{z}}\mathbf{x} + \frac{\partial \mathbf{x}}{\partial \mathbf{z}}\mathbf{y}$$

.

Proof: By definition,

$$\begin{aligned}\frac{\partial a}{\partial \mathbf{z}_j} = \frac{\partial}{\partial \mathbf{z}_j}\left(\sum_k \mathbf{x}_k\mathbf{y}_k\right) &= \sum_k \mathbf{x}_k\frac{\partial \mathbf{y}_k}{\partial \mathbf{z}_j} + \mathbf{y}_k\frac{\partial \mathbf{x}_k}{\partial \mathbf{z}_j} \\ &= \frac{\partial \mathbf{y}}{\partial \mathbf{z}_j}\mathbf{x} + \frac{\partial \mathbf{x}}{\partial \mathbf{z}_j}\mathbf{y}\,.\end{aligned}$$

In the last line, we have used the fact that $\frac{\partial \mathbf{y}_k}{\partial \mathbf{z}_j}$ is an element of the matrix $\frac{\partial \mathbf{y}}{\partial \mathbf{z}}$ with the k's parameterizing the row and the j's parameterizing the columns. Therefore $\frac{\partial \mathbf{y}}{\partial \mathbf{z}_j}$ is a *row* vector.

Matrix Derivatives

Question: Find

$$\frac{\partial(\mathbf{x}^T\mathbf{x})}{\partial\mathbf{x}} =$$

and

$$\frac{\partial(\mathbf{x}^T\mathbf{a})^2}{\partial\mathbf{x}} =$$

Answer:

$$\frac{\partial(\mathbf{x}\mathbf{x}^T)}{\partial\mathbf{x}} = 2\mathbf{x}$$

and

$$\frac{\partial(\mathbf{x}^T\mathbf{a})^2}{\partial\mathbf{x}} = 2(\mathbf{x}^T\mathbf{a})\mathbf{a}$$

Matrix Derivatives

Question: Find Lets take a moment to think about what is required in a proof for

$$\frac{\partial(\mathbf{x}^T\mathbf{a})^2}{\partial \mathbf{x}} = 2(\mathbf{x}^T\mathbf{a})\mathbf{a}$$

Incomplete proof:

$$\frac{\partial(\mathbf{x}^T\mathbf{a})^2}{\partial \mathbf{x}} = 2(\mathbf{x}^T\mathbf{a})\frac{\partial(\mathbf{x}^T\mathbf{a})}{\partial \mathbf{x}} = 2(\mathbf{x}^T\mathbf{a})\mathbf{a}\,.$$

Here, you need to justify the second step. It is true that this holds for scalars by the chain rule for the gradient but a word of justification is needed.

Matrix Derivatives

Question: Find Lets take a moment to think about what is required in a proof for

$$\frac{\partial(\mathbf{x}^T\mathbf{a})^2}{\partial \mathbf{x}} = 2(\mathbf{x}^T\mathbf{a})\mathbf{a}$$

Complete proof:

By the one variable chain rule,

$$\frac{\partial(\mathbf{x}^T\mathbf{a})^2}{\partial \mathbf{x}_i} = 2(\mathbf{x}^T\mathbf{a})\frac{\partial(\mathbf{x}^T\mathbf{a})}{\partial \mathbf{x}_i} = 2(\mathbf{x}^T\mathbf{a})\mathbf{a}_i\,.$$

Since $\frac{\partial(\mathbf{x}^T\mathbf{a})^2}{\partial \mathbf{x}}$ is a column vector with i'th entry $2(\mathbf{x}^T\mathbf{a})\mathbf{a}_i$, it can be written

$$\frac{\partial(\mathbf{x}^T\mathbf{a})^2}{\partial \mathbf{x}_i} = 2(\mathbf{x}^T\mathbf{a})\mathbf{a}\,.$$

Matrix Derivatives

Additional Problems: Assume that $\mathbf{x}$ and $\mathbf{y}$ depend on $\mathbf{z}$. Show that

Exercise 1

$$\frac{\partial(\mathbf{y}^T \mathbf{A}\mathbf{x})}{\partial \mathbf{z}} = \frac{\partial \mathbf{y}}{\partial z} A\mathbf{x} + \frac{\partial \mathbf{x}}{\partial z} A^T \mathbf{y}$$

Exercise 2

Let $\mathbf{A}$ be a invertable matrix that depends on a scalar x. Use the fact that $\mathbf{A}^{-1}\mathbf{A} = I$ to show that

$$\frac{d\mathbf{A}^{-1}}{dx} = \mathbf{A}^{-1} \frac{d\mathbf{A}}{dx} \mathbf{A}^{-1}$$