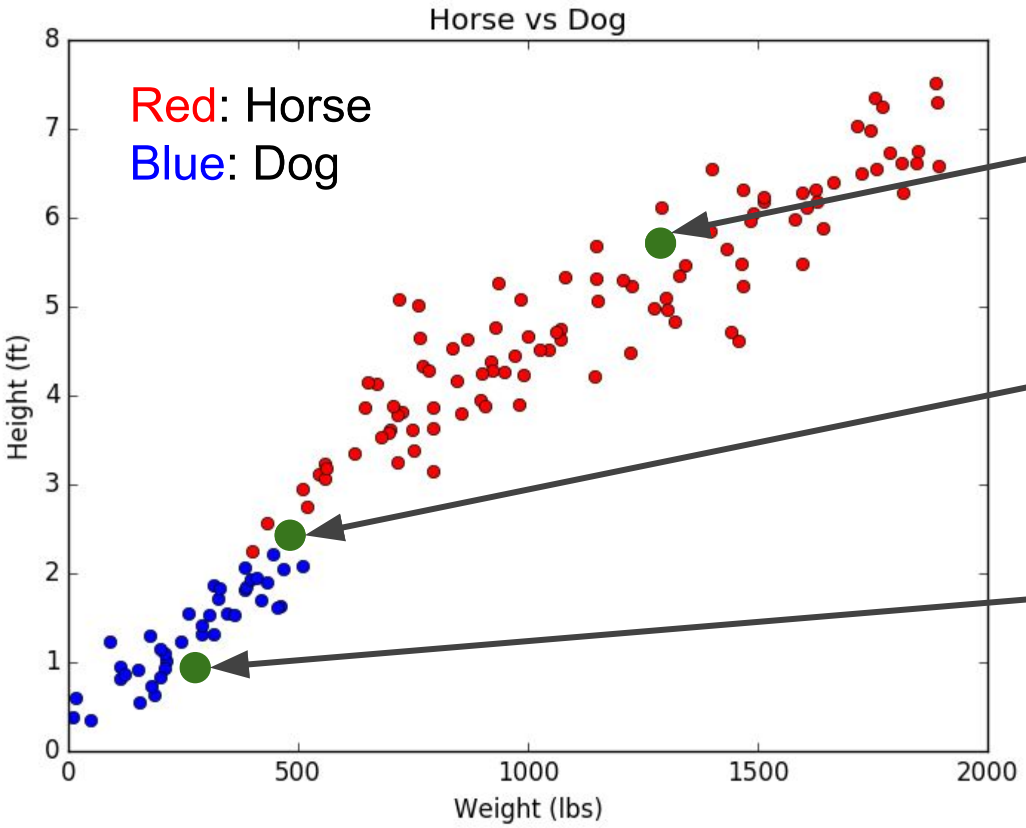
k-Nearest Neighbors (kNN)

Mark Llorente, based on slides from RH

galvanize

- k-Nearest Neighbors
- The Curse of Dimensionality
- Parametric vs Nonparametric Models

Is it a big dog or a small horse?



New datapoint:
Is it a horse or a dog?

New datapoint:
Is it a horse or a dog?

New datapoint:
Is it a horse or a dog?

The k-Nearest Neighbors algorithm:

Training algorithm:

1. Store all the data… that’s all.

Prediction algorithm (predict the class of a new point x’):

1. Calculate the distance from x’ to all points in your dataset.
2. Sort the points in your dataset by increasing distance from x’.
3. Predict the majority label of the *k* closest points.

What is *k*?

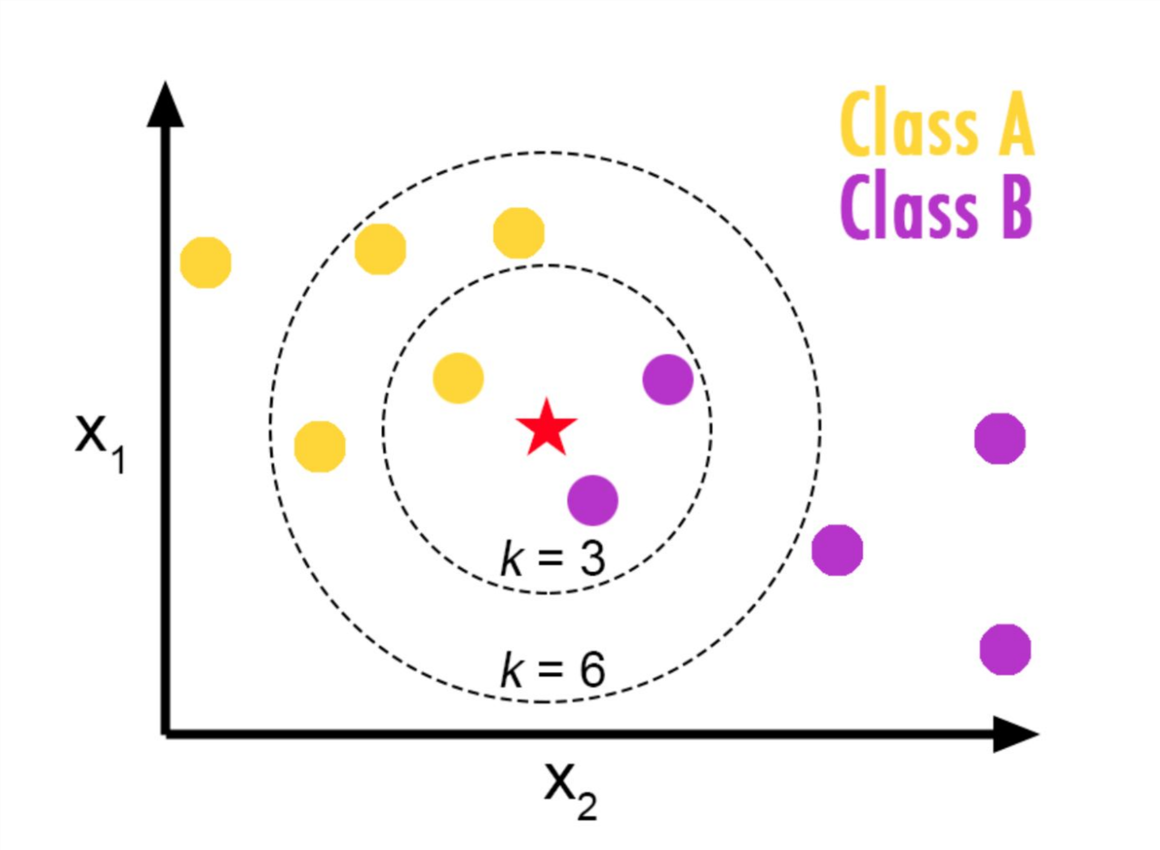
Distance Metrics

Euclidean Distance (L2): $$\sum_i (a_i - b_i)^2$$

Manhattan Distance (L1): $$\sum_i |a_i - b_i|$$

Cosine Distance = 1 - Cosine Similarity: $$1 - \frac{a \cdot b}{||a||||b||}$$

kNN: you pick k



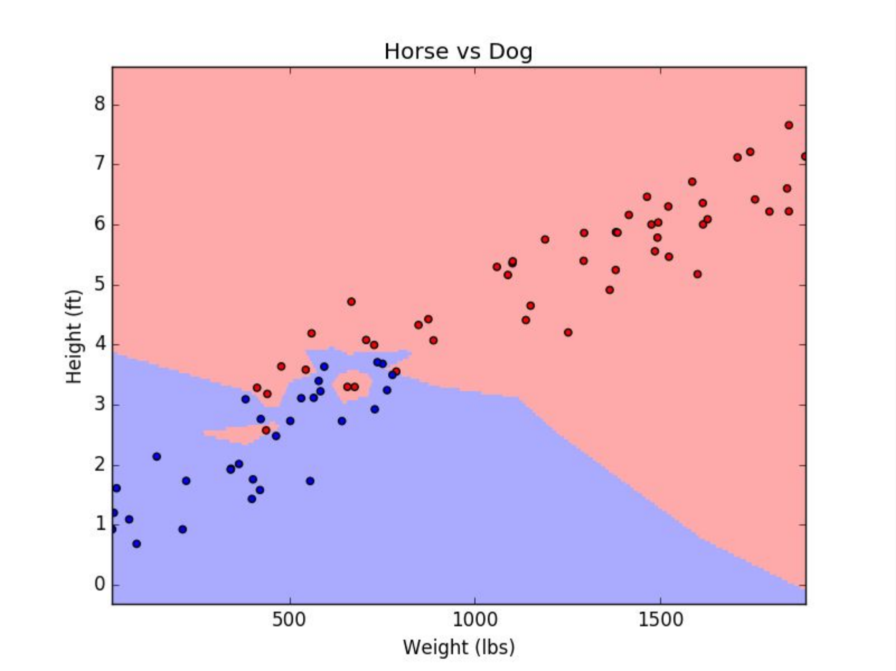
What is the prediction for ★ when k=3?

What is the prediction for ★ when k=6?

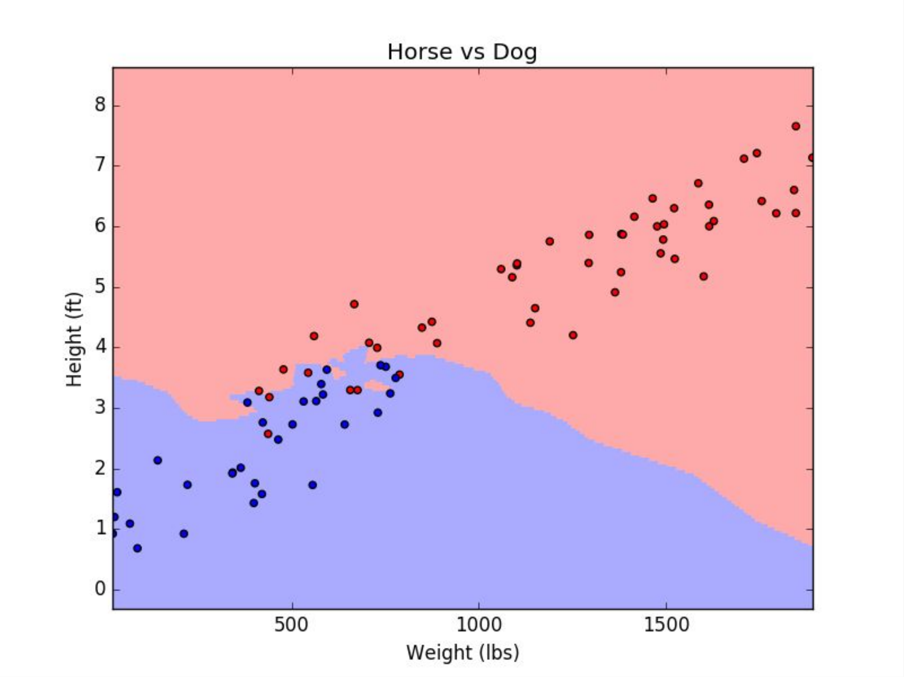
The only hyperparameter… *k*… the number of nearest neighbors to consider

galvanize

k=1

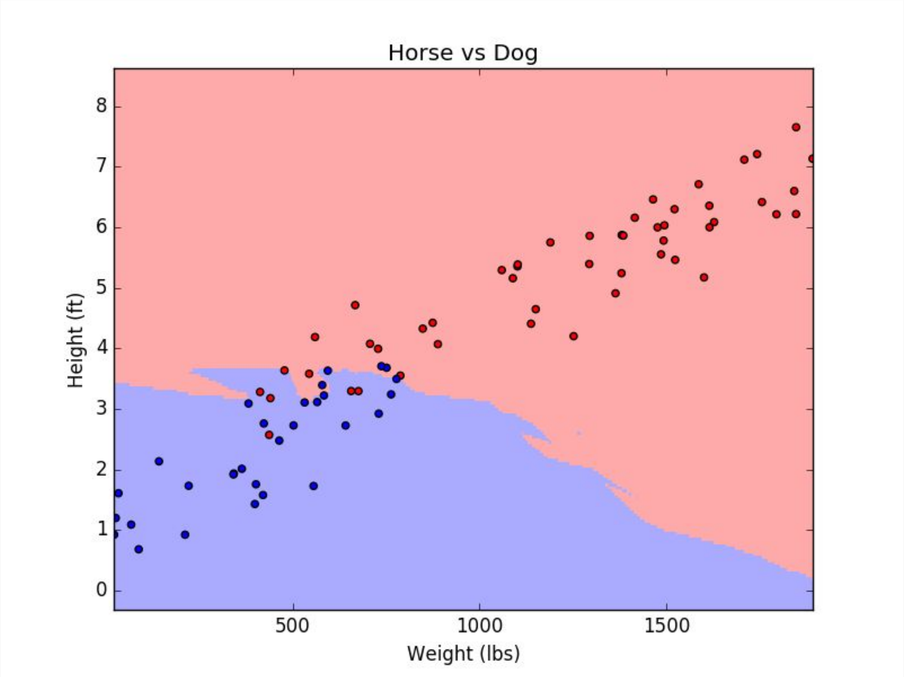


k=5

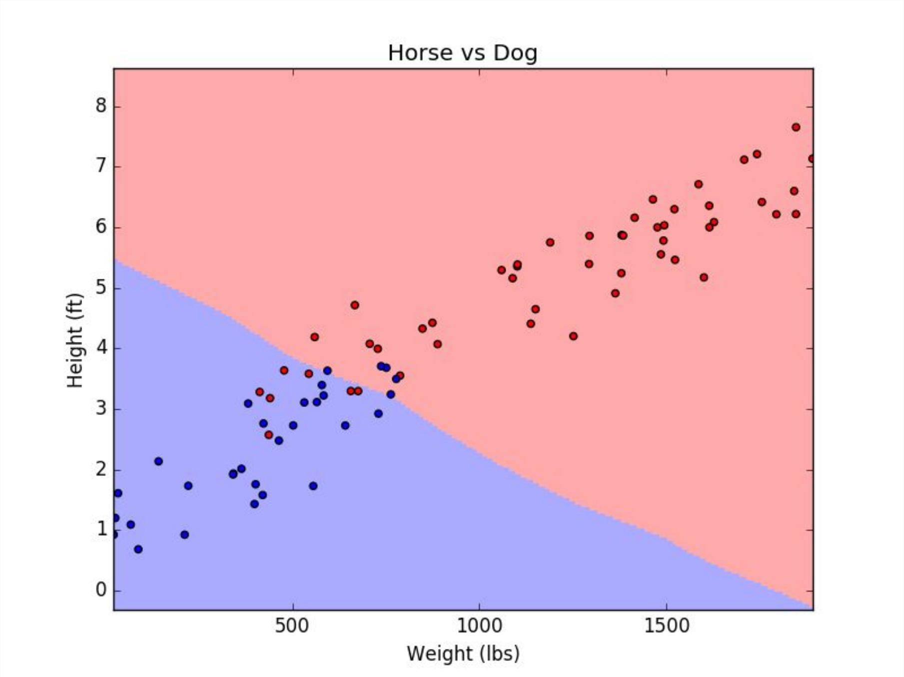


The only hyperparameter… *k*… the number of nearest neighbors to consider

k=10

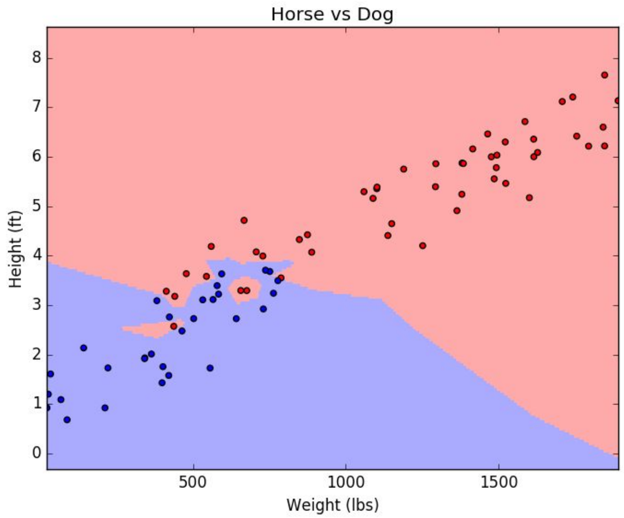


k=50



Which model seems overfit?

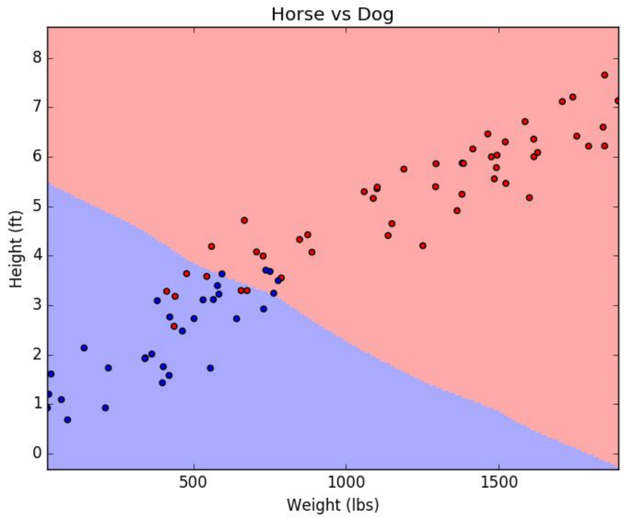
k=1



Btw, ss a <u>general</u> rule, start with:

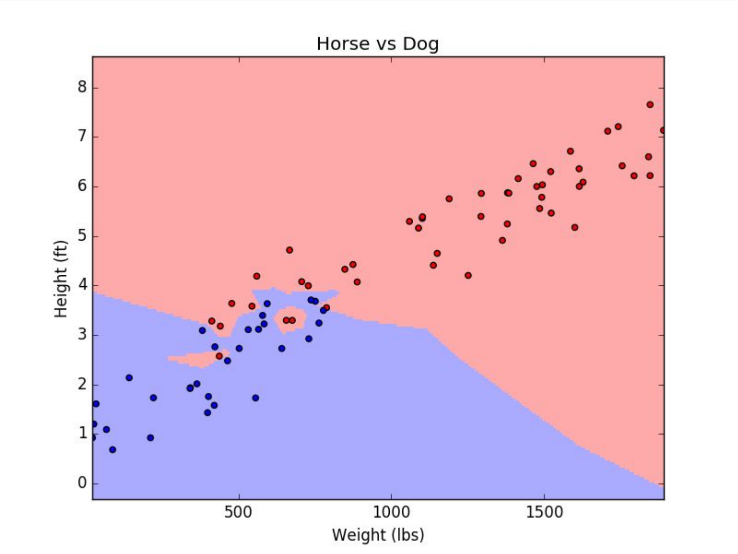
$$k = \sqrt{n}$$

k=50

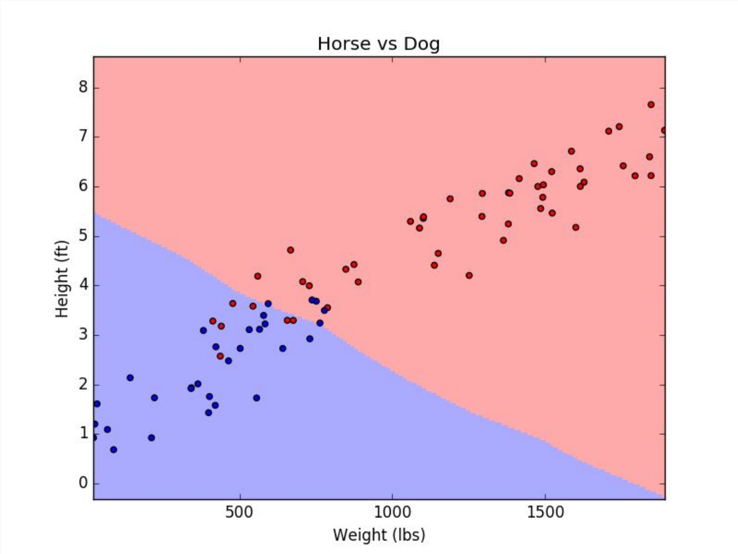


Breakout: What happens to model variance when *k* increases?

k=1

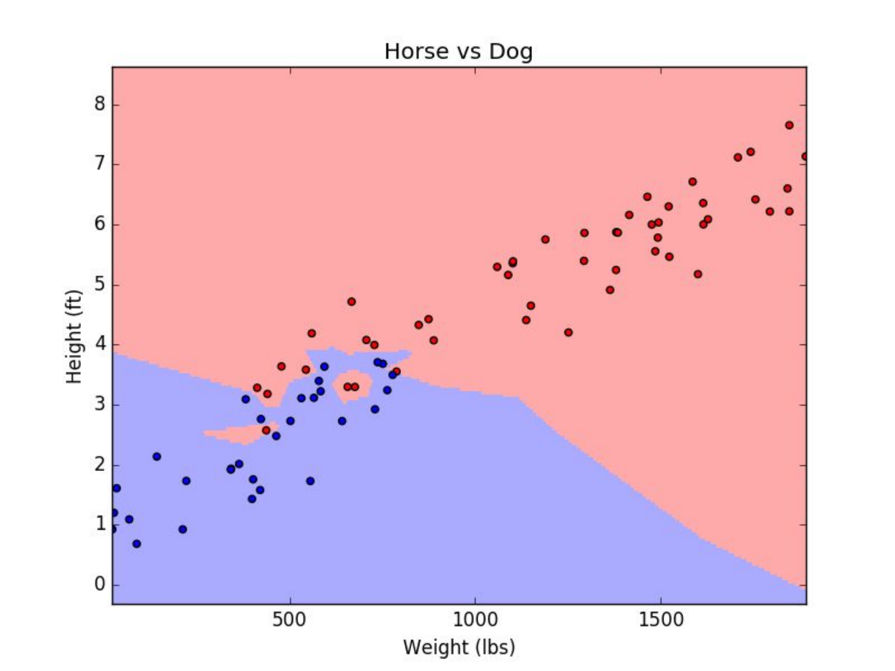


k=50

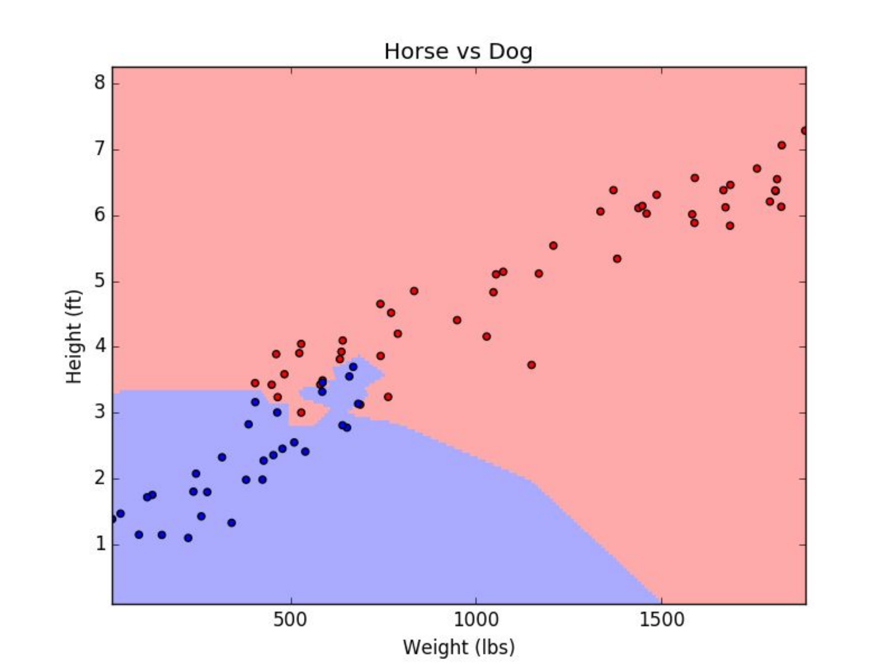


See the high variance?
Each dataset is randomly generated from the same population.

k=1

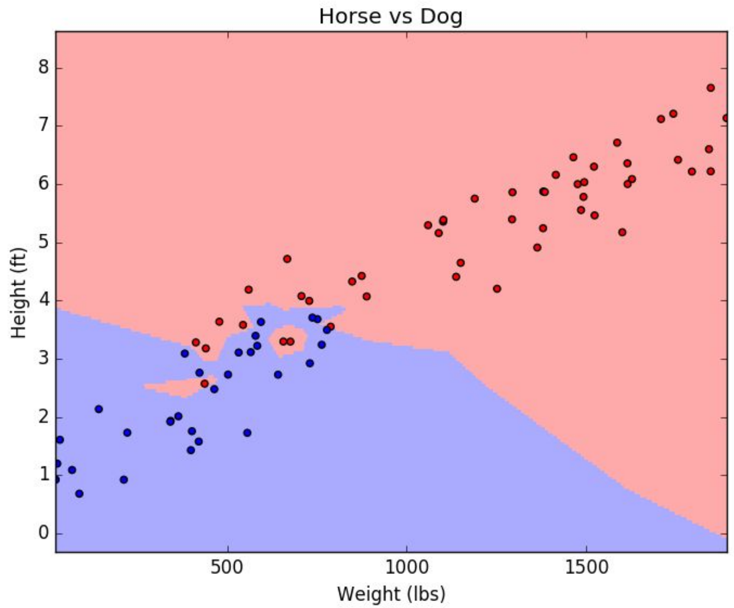


k=1

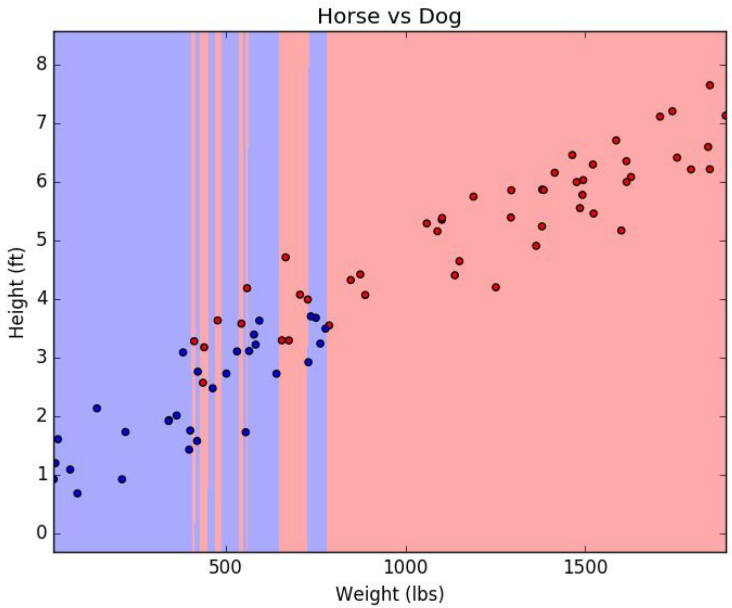


Be careful with the scale of your features!

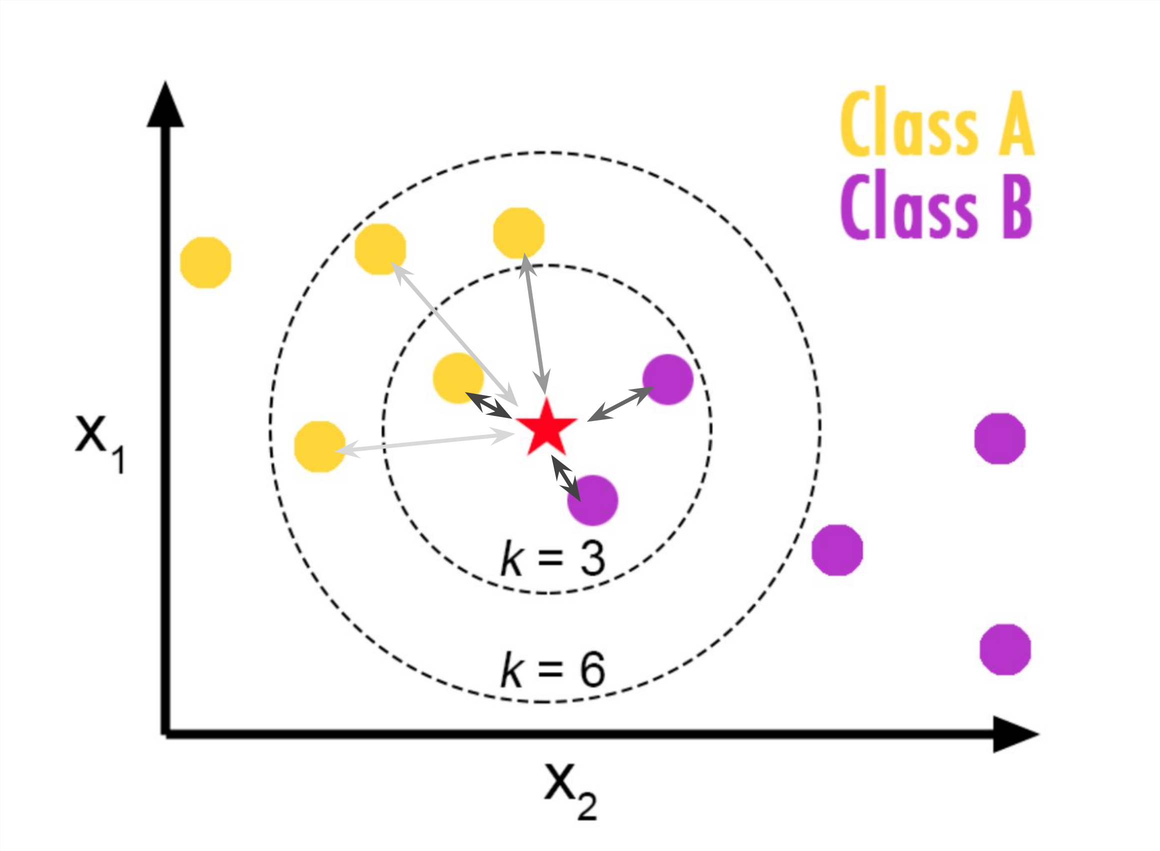
k=1, scaled features



k=1, original-scale features



Point weighting (weighted voting)



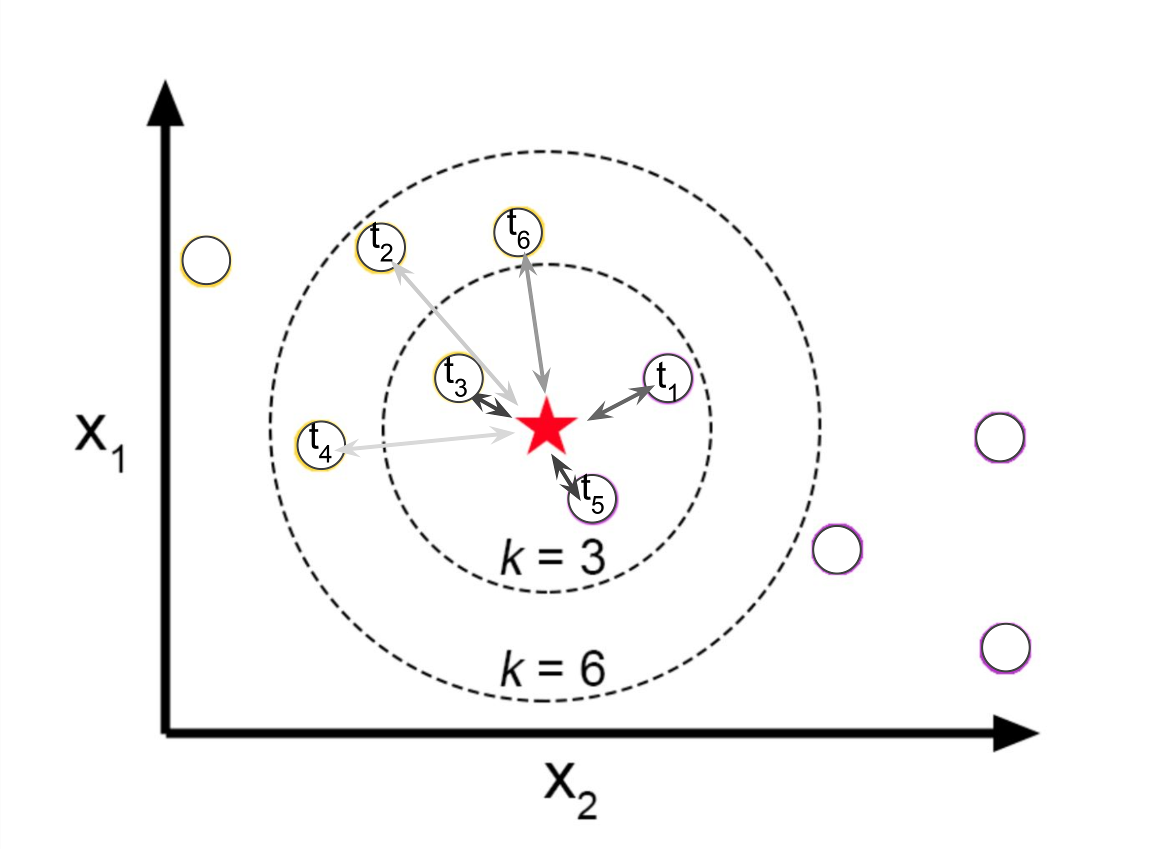
Let the k nearest points have distances:

$$d_1, d_2, \ldots, d_k$$

The i^{th} point votes with a weight of:

$$\frac{1}{d_i}$$

small distances are weighted more!



Let the *k* nearest points have distances:

$$d_1, d_2, \ldots, d_k$$

Let the *k* nearest points have targets:

$$t_1, t_2, \ldots, t_k$$

How can we do regression with kNN?

Predict the mean value of the k neighbors, or predict a weighted average.

kNN in high dimensions...

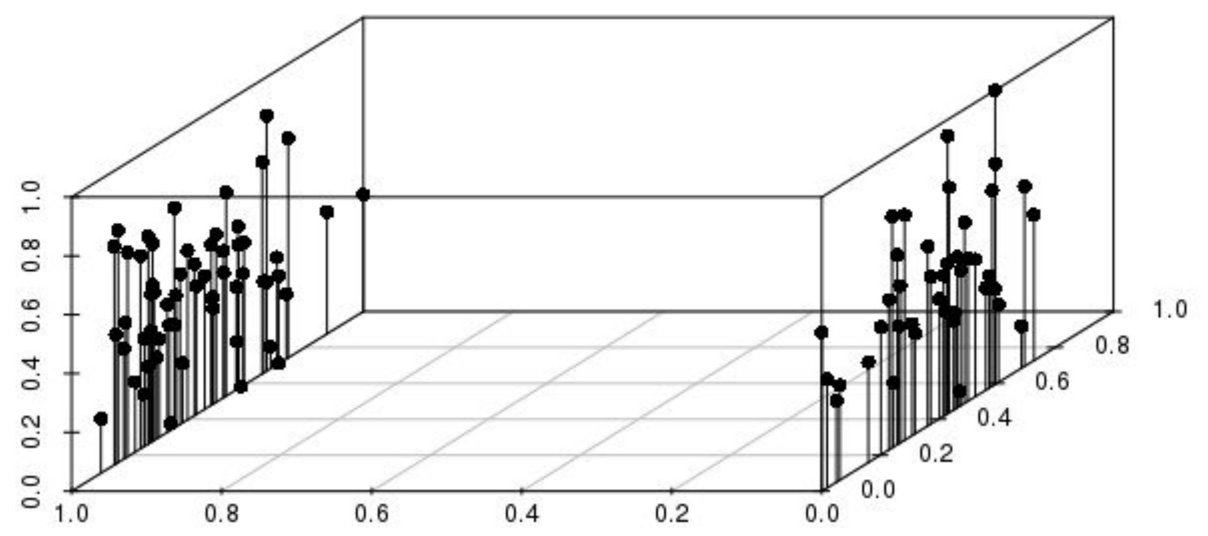
kNN is problematic when used with high dimensional (d) spaces…
... but it works pretty well (in *general*) for d<5

The nearest neighbors can be very **“far away” in high dimensions**…

Say you want to use a neighborhood of 10% (i.e. k = 0.1*n)

Let’s see how this looks as we increase the dimensionality… (next slide)

kNN in high dimensions, binary feature (for emphasis)



kNN in high dimensions

Mark, Go to Board

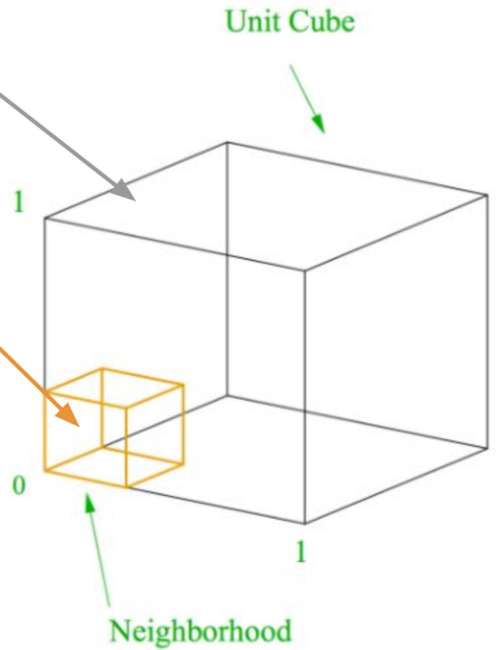
The Curse of Dimensionality **(another perspective)**

galvanize

Say we have a unit (hyper)cube.

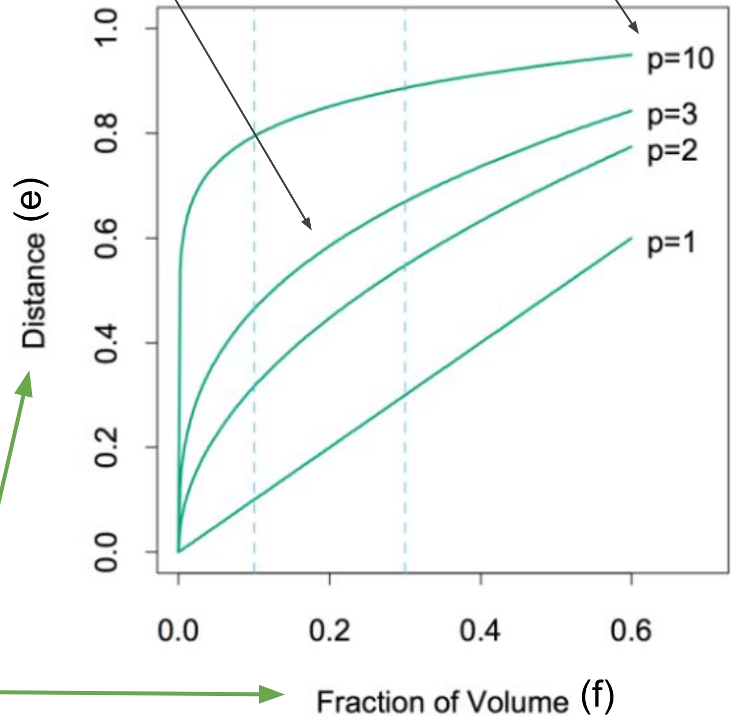
We want to create another (hyper)cube inside the outer cube so that we fill X% of the outer cube.

How long must the edges of the inner cube be?



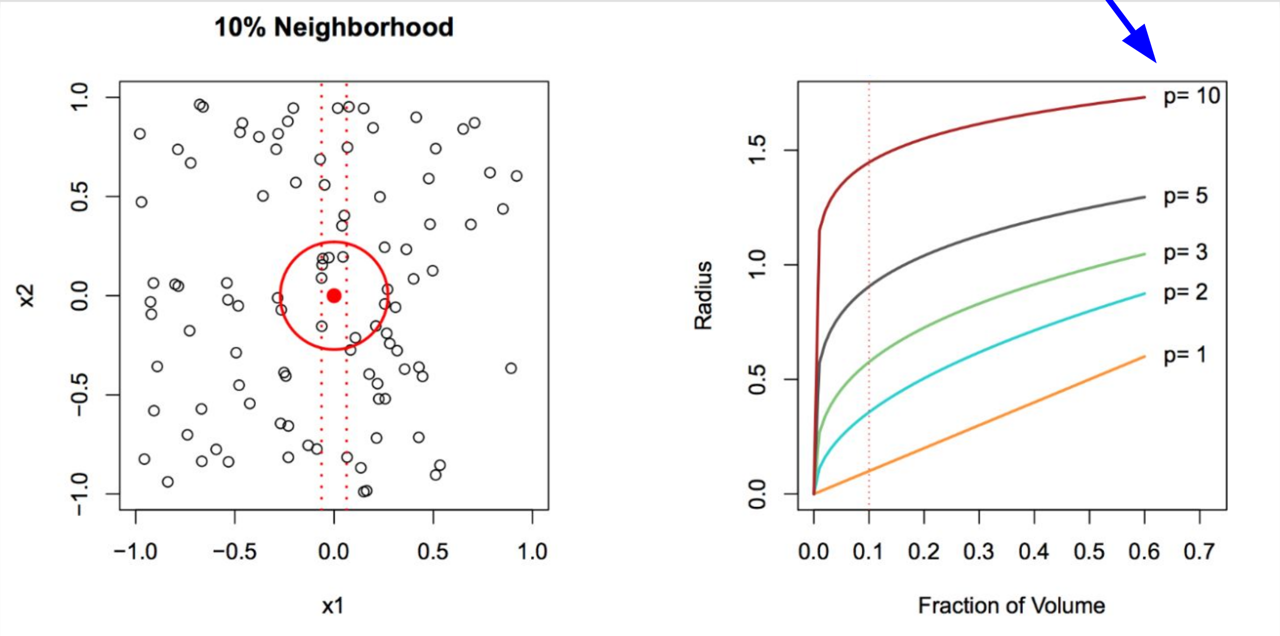
$$e = f^{1/p}$$

p = dimensionality



The Curse of Dimensionality

p = dimensionality



When p=1, we are only considering x1. When p=2, we are considering x1 and x2.

Notice the required radius in 2D is much larger than the required radius in 1D.

As we increase the dimensionality, we lose the concept of locality.

The Curse of Dimensionality **(another perspective)**

Say you have a dataset with 100 samples, each with only one predictor.

But, one predictor doesn't tell you enough, so you collect a new dataset, and this time you measure 10 predictors for each sample.

How many samples do you need in your new (10 predictor) dataset to achieve the same "sample density" as you originally had (in the one-predictor dataset)?

Just 100^10, that not that many… just

100,000,000,000,000,000,000

The Curse of Dimensionality (another perspective)

"What about for a slightly less extreme changes in d?"

N_0	original number of data points
$V_0 = L^p$	volume of "predictor space" where **L** is an arbitrary (unit) length for the hypercube that our data resides in. **p** is number of predictors
$n_0 = N/V_0$	where $\mathbf{n_0}$ is our original data density

Go from **p** → **q** (e.g. N_0=1,000 for 3 predictors → N'=? for 7 predictors)

How many data points do we need to maintain $n'=n_0$? *What is the new N' we need with respect to the old N?*

Strategy: Find density for 1-dimension then up-convert to new dimension.

$n' = N'/V'$ where $V'=L^q$ [← note **p** has been replaced with new number of predictors **q**]

$n_{1\text{-}d}=(N_0/V_0)^{1/p}$

$n' = n_{1\text{-}d}^{\,q} = (N_0/V_0)^{q/p} = N_0^{\,q/p}/L^q = \boxed{N_0^{\,q/p}/V'}$

i.e. for new number of parameters q, N needs to be

$N_0^{\,q/p}$, or in our example, $N_0^{\,7/3}$ or 10,000,000 data points!

The Curse of Dimensionality… takeaways

- kNN (or any method that relies on distance metrics) will suffer in high dimensions.
  - Nearest neighbors are “far” away in high dimensions (even for d=10).
- A 10% neighborhood in a high dimensional unit hypercube requires a hypersphere with large radius.
  - Hyperspheres are weird in high dimensions…
- High dimensional data tends to be sparse; it’s easy to overfit sparse data.
  - It takes A LOT OF DATA to make up for increased dimensionality.

Parametric vs Non-parametric Models

Parametric models have a <u>fixed</u> number of learned parameters.
- Logistic regression is parametric.
- kNN is non-parametric.

Parametric models are more structured. The added structure often combats the curse of dimensionality... as long as the structure is derived from reasonable assumptions.

Alternate perspective: Parametric models are not distance based, so the curse doesn't apply!

Summary: kNN

Pros:

- super-simple
- training is trivial (store the data)
- works with any number of classes
- easy to add more data
- few hyperparameters:
  - *k*
  - *distance metric*

Cons:

- high prediction cost (especially for large datasets)
- high-dims = bad
  - we'll learn dimensionality reduction methods in two weeks!
- categorical features don't work well...

The Curse of Dimensionality (another perspective)

galvanize

Don't freak out...

$$\lim_{d\to\infty} \frac{V_{\text{sphere}}(R,d)}{V_{\text{cube}}(R,d)} = \lim_{d\to\infty} \frac{\frac{\pi^{d/2}R^d}{\Gamma(d/2+1)}}{(2R)^d} = \lim_{d\to\infty} \frac{\pi^{d/2}}{2^d\Gamma(d/2+1)} = 0$$

Euler's gamma function…
basically, it's the *factorial* function that can operate on fractional numbers

Factorial overtakes exponentiation in the limit…
e.g.

$$\lim_{x\to\infty} \frac{c^x}{x!} = 0$$

What does this mean?