SVM for Regression

SVM's can be used not only for Classification but for Regression as well.

For Classification, the Loss function tries to fit all training examples *outside* the buffer zone.

For Regression, the Loss function tries to fit all training examples *within* the buffer zone.

- This is analogous to forcing training examples to be close to the fitted line

Importance of normalization of $\mathbf{x}$

Imagine that the various components of $\mathbf{x}$ are on vastly different scales

- suppose $\mathbf{x}_1$ is in "billions of units" (so $\mathbf{x}_1$ is small) but $\mathbf{x}_2$ is in single units (large)

Then parameter Θ_1 (which multiplies $\mathbf{x}_1$) will likely be much smaller than Θ_2.

The Margin Penalty, which tries to reduce the value of elements of Θ will thus be more likely to reduce Θ_2 than Θ_1 for no reason other than the disparate scales.

As we saw in our lecture on transformations, this is a danger with loss/cost functions that include a penalty for parameters magnitude.

The disparate scales of features in $\mathbf{x}$ can also affect some transformations.

For the Gaussian RBF, the 2-norm will be dominated by the larger dimension, almost to the exclusion of smaller dimensions.

Thus, if the chosen ϕ is sensitive to scale (such as the Gaussian RBF) it's important to normalize the training set (across each dimension of features) so that all features (elements of $\mathbf{x}$) are on similar scale.

SVM Drawbacks

Complexity

The SVM may utilize up to m landmarks from the vectors among the training set.

These landmarks are referred to as "support vectors".

As m may be very large

- SVM models may consume a lot of memory (in the test phase, not just when being fit).

The choice of the complexity of a kernel depends on the relationship between m, the number of training examples, and n, the number of features.

- When m is large (and large relative to $n, m \gg n$) there is a lot of data on which to fit a complicated kernel.
    - however, when m is really large, this can be expensive
- When m is small, the lack of data for fitting suggests a simple, linear kernel or logistic regression
- when n is small, simpler Kernels are Logistic Regression is suggested

Aside

- The SVC Optimization problem is a type of Quadratic Programming problem
    - In the "primal" form of the equation: there are m (number of examples) constraints
    - In the "dual" form of the equation, there are n (number of features) constraints
- Kernels work on the dual form of the equation
    - So if n', the number of transformed features, is much bigger than m, kernels may cost more than pre-processing transformations

SVM in `sklearn`

`sklearn` has several SVM algorithms (using different optimization techniques) that, mathematically, should yield equivalent results.

The difference is that some optimize for time (faster solution) versus space (ability to handle bigger training sets) versus being more general solvers.

According to Geron (page 194), the following should be equivalent

- `LinearSVC(C=1, loss="hinge")`
- `SVC(C=1,kernel="linear")`
- `SGDClassifier(loss="hinge", alpha=1/(m*C))`

Final words

The SVM is very powerful, but it has a lot of moving parts

- Hinge Loss
- Large Margin Classification
- Kernel

The power of the SVM comes in the tight integration of all these parts.

Main take-aways

- The SVM offers you built-in transformations to help make the data set closer to being Linearly Separable
- You can use the kernel function even without understanding the transformation ϕ that corresponds to it
- Some kernel functions have hyper-parameters
    - The SVM can solve for the best hyper-parameter
- SVM's can also be used for Regression tasks

Preview of Deep Learning

Two SVM features

- The "max" function of the Hinge Loss
- The semi-automatic construction of synthetic features (kernels)

will be key features of a different set of models using Deep Learning that are the focus of the second half of this course.

```
In [4]: print("Done")
```

```
Done
```