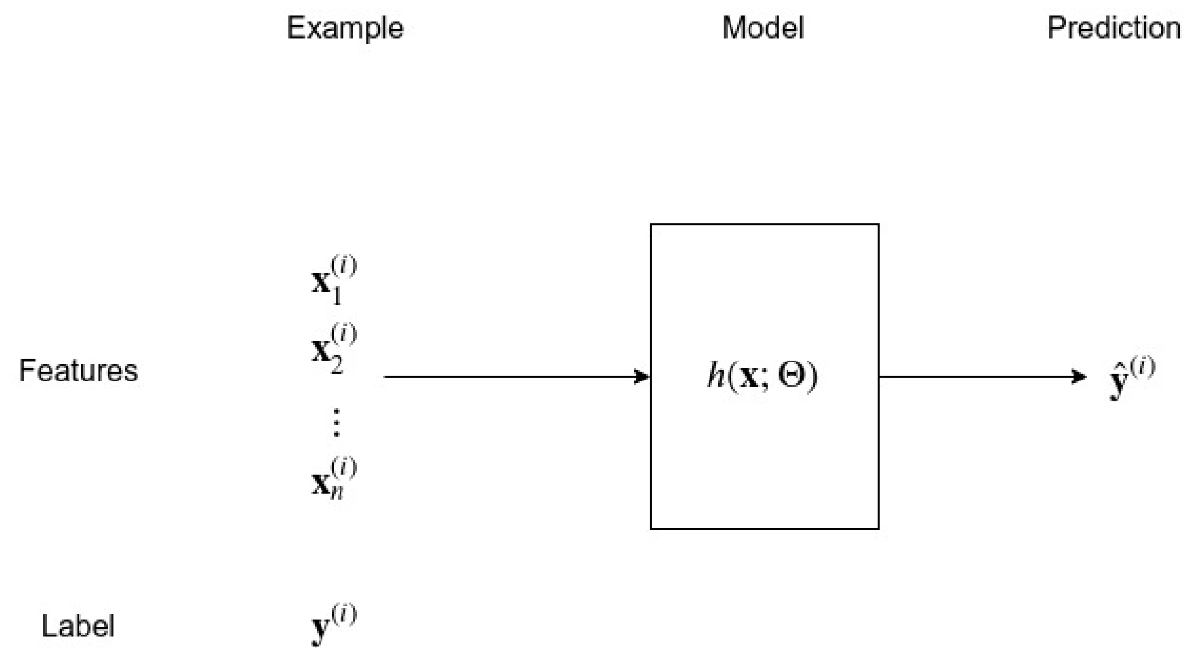
Notation

Training



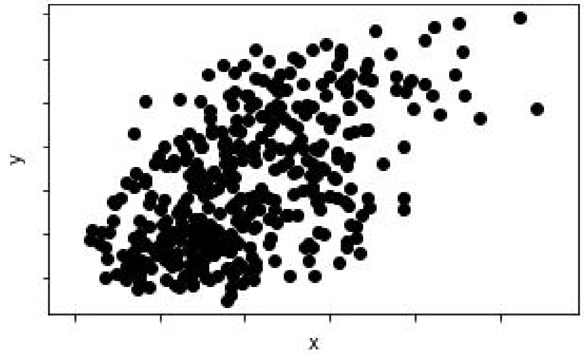
- Training

The key task of Machine Learning is finding the "best" values for parameters $\mathbf{\Theta}$.

The process of using training examples $\mathbf{X}$ to find $\mathbf{\Theta}$

- is called *fitting* the model
- is solved as an optimization problem (to be described)

Fitting a model



$\langle \mathbf{X}, \mathbf{y} \rangle$ Training examples

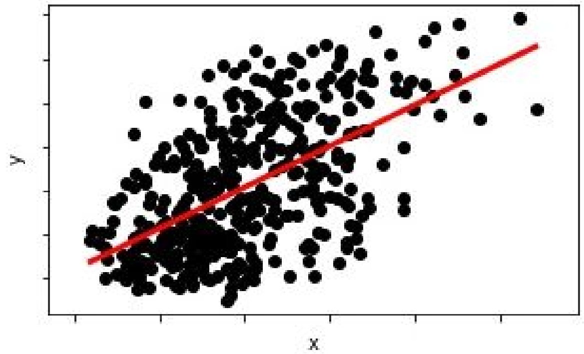
$\mathbf{X} : m \times n$
$\mathbf{y} : m \times 1$

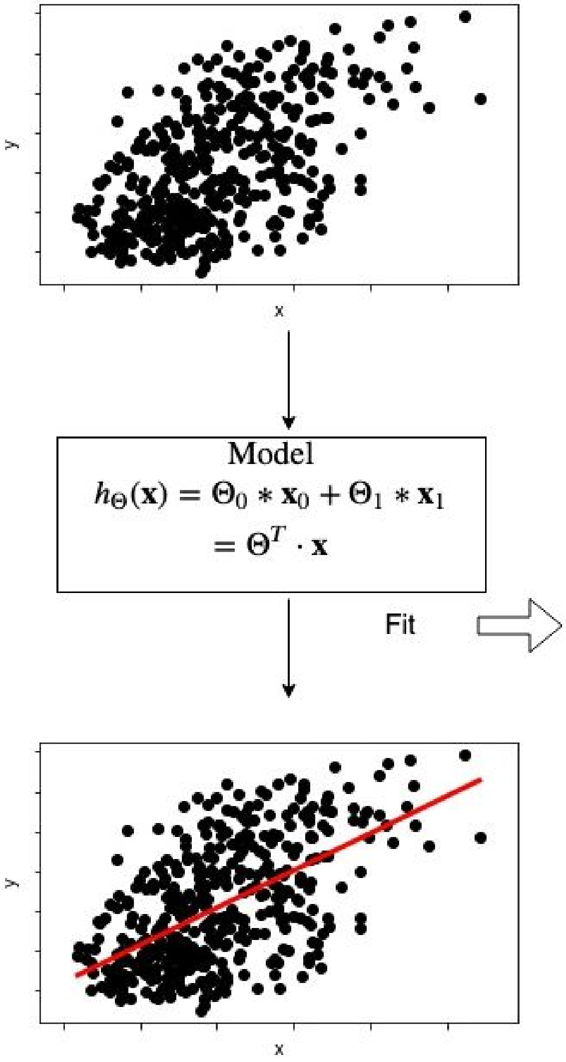
Model
$h_\Theta(\mathbf{x})$

Fitting, training

Fit $\Rightarrow$ Θ

$\Theta : n \times 1$





$\langle \mathbf{X}, \mathbf{y} \rangle$ Training examples

$\mathbf{X} : m \times n$
$\mathbf{y} : m \times 1$

Fitting, training

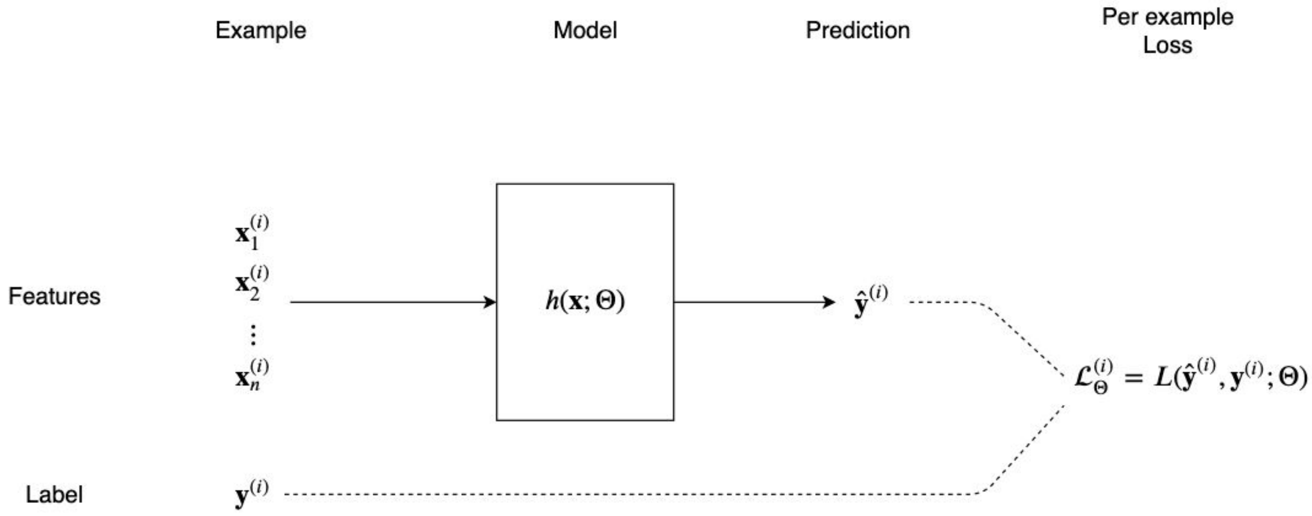
$\Theta = [\Theta_0, \Theta_1] = [\text{intercept, slope}]$

$\Theta : (n+1) \times 1$

$\mathbf{x}^{(i)} = [1, \mathbf{x}_1^{(i)}, \dots, \mathbf{x}_n] : (n+1) \times 1$

Training, optimization

Training Example



The Loss for the entire training set is simply the average (across examples) of the Loss for the example

$$\mathcal{L}_\Theta = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_\Theta^{(\mathbf{i})}$$

The best (optimal) Θ is the one that minimizes the Average (across trainng examples) Loss

$$\Theta = \operatorname*{argmin}_{\Theta} \$\mathcal{L}_{\Theta}$$

K Nearest Neighbors

Test example

$\mathbf{x}$

Training examples

Training examples			Targets ordered in decreasing order of similarity of $\mathbf{x}^{(i)}$ to $\mathbf{x}$
$\mathbf{x}^{(1)}$	$\text{similarity}(\mathbf{x}, \mathbf{x}^{(1)})$	$\text{similarity}(\mathbf{x}, \mathbf{x}^{(i_1)})$	$\mathbf{y}^{(i_1)}$
$\mathbf{x}^{(2)}$	$\text{similarity}(\mathbf{x}, \mathbf{x}^{(2)})$	$\text{similarity}(\mathbf{x}, \mathbf{x}^{(i_2)})$	$\mathbf{y}^{(i_2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}^{(m)}$	$\text{similarity}(\mathbf{x}, \mathbf{x}^{(m)})$	$\text{similarity}(\mathbf{x}, \mathbf{x}^{(i_m)})$	$\mathbf{y}^{(i_m)}$

Compute similarity of $\mathbf{x}$ to each $\mathbf{x}^{(i)}$

Sort $\mathbf{x}^{(i)}$ in decreasing order of similarity to $\mathbf{x}$

Prediction: $\hat{\mathbf{y}} = \text{most frequent}(\mathbf{y}^{(i_1)}, \ldots, \mathbf{y}^{(i_k)})$

Here's an illustration of KNN in action:

- training example
$$\mathbf{x^{(i)}} = [\mathbf{x}_1, \mathbf{x}_2], \mathbf{y^{(i)}} \in \{0, 1\}$$
is plotted as a colored dot, with the color corresponding to $\mathbf{y^{(i)}}$
- we form many test (non-training) examples by creating arbitrary pairs of $\mathbf{x}_1, \mathbf{x}_2$ values in a grid
    - predict for each, fill the grid with a color corresponding to the predicted class

The line separating colors (classes) is called the *separating* or *decision* boundary.

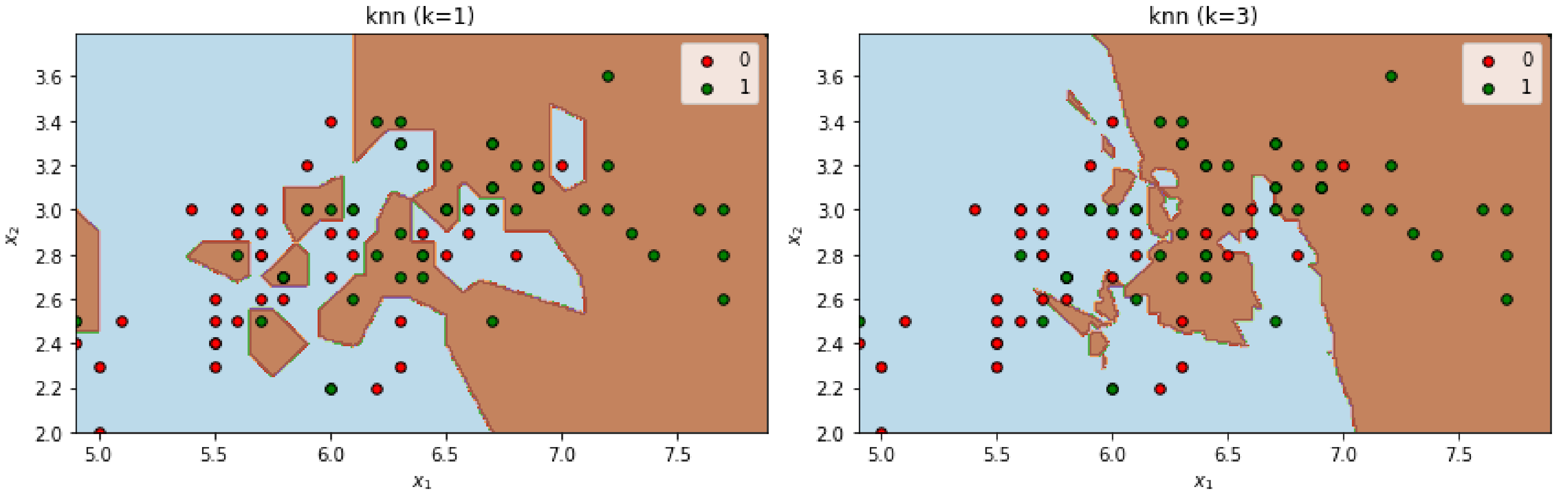
In [4]:

```
from sklearn.neighbors import KNeighborsClassifier

classifiers =  [ ("knn (k=1)", KNeighborsClassifier(1)),
                 ("knn (k=3)", KNeighborsClassifier(5))
               ]
svmh = svm_helper.SVM_Helper()
_= svmh.create_kernel_data(classifiers=classifiers)
fig, axs = svmh.plot_kernel_vs_transform(show_margins=False)
plt.close()
```

In [5]:
```
fig
```

Out[5]:



Useful tools to help with Markdown

A couple of great tools

- Detexify (http://detexify.kirelabs.org/classify.html)
    - hand-drawn symbols convert to TeX !
- Mathpix (https://mathpix.com/)
    - Screen-shot to markdown !

In [6]:

```python
print("Done")
```

```
Done
```