Support Vector Machines

CSC 411 Tutorial

November (2nd, 3rd, 13th, 15th) 2017

Tutor: Bowen Xu

Many thanks to Jake Snell and Kevin Swersky for much of the following material.

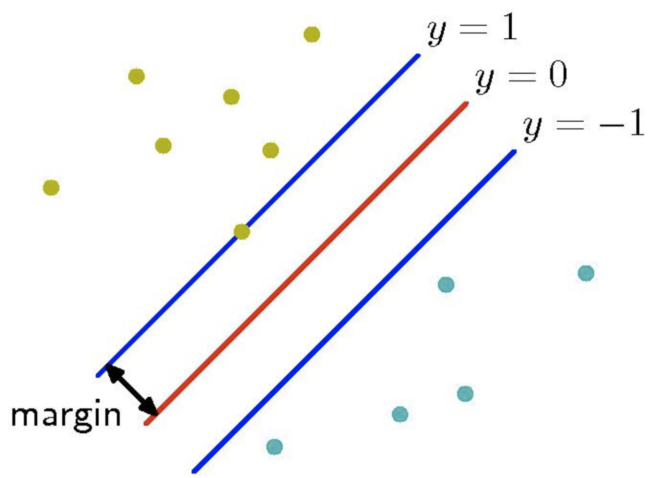
Brief Review of SVMS

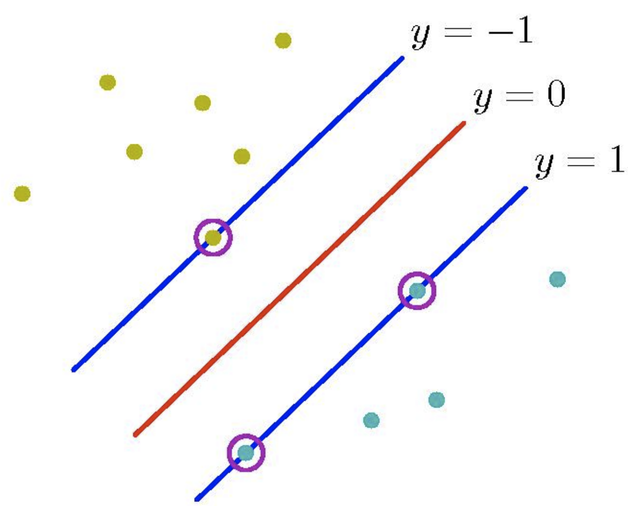
Geometric Intuition

In [15]:

```
side_by_side(im2html(Image("loose.png"), width=450),
             im2html(Image("tight.png"), width=450))
```

Out[15]:



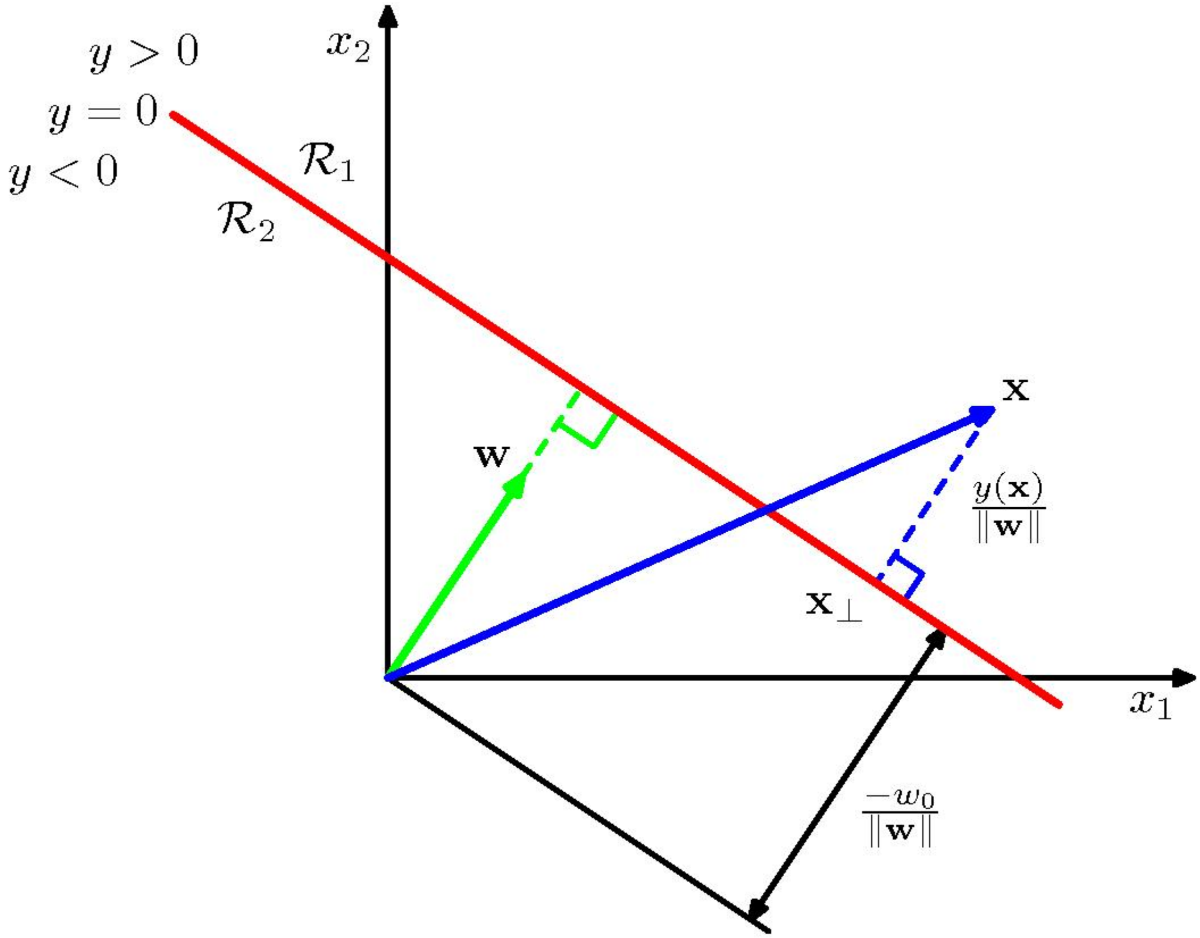


Margin Derivation

In [65]:

```
Image("http://research.microsoft.com/en-us/um/people/cmbishop/PRML/prmlfigs-jpg/Figure4.1.jpg")
```

Out[65]:



Margin Derivation

Compute the distance d_n of an arbitrary point x_n in the (+) class to the separating hyperplane.

$$\begin{aligned} w^T \left(x_n - d_n \frac{w}{||w||} \right) + b = 0 \\ w^T x_n - d_n \frac{w^T w}{||w||} + b = 0 \\ w^T x_n + b = d_n ||w|| \\ d_n = \frac{w^T x_n + b}{||w||} \end{aligned}$$

If we let $t_n \in \{1, -1\}$ denote the class of x_n, then the distance becomes

$$d_n = \frac{t_n (w^T x_n + b)}{||w||}$$

SVM Problem

But scaling $w \to \kappa w$ and $b \to \kappa b$ doesn't change $d_n = \frac{t_n(w^T x_n + b)}{||w||}$.

We can set $d_n = \frac{1}{||w||}$ for the point x_n closest to the decision boundary, leading to the problem:

$$\max \frac{1}{||w||}$$
$$\text{s.t. } t_n(w^T x_n + b) \geq 1, \text{ for } n = 1 \ldots N$$

or equivalently:

$$\min \frac{1}{2}||w||^2$$
$$\text{s.t. } t_n(w^T x_n + b) \geq 1, \text{ for } n = 1 \ldots N$$

Non-linear SVMs

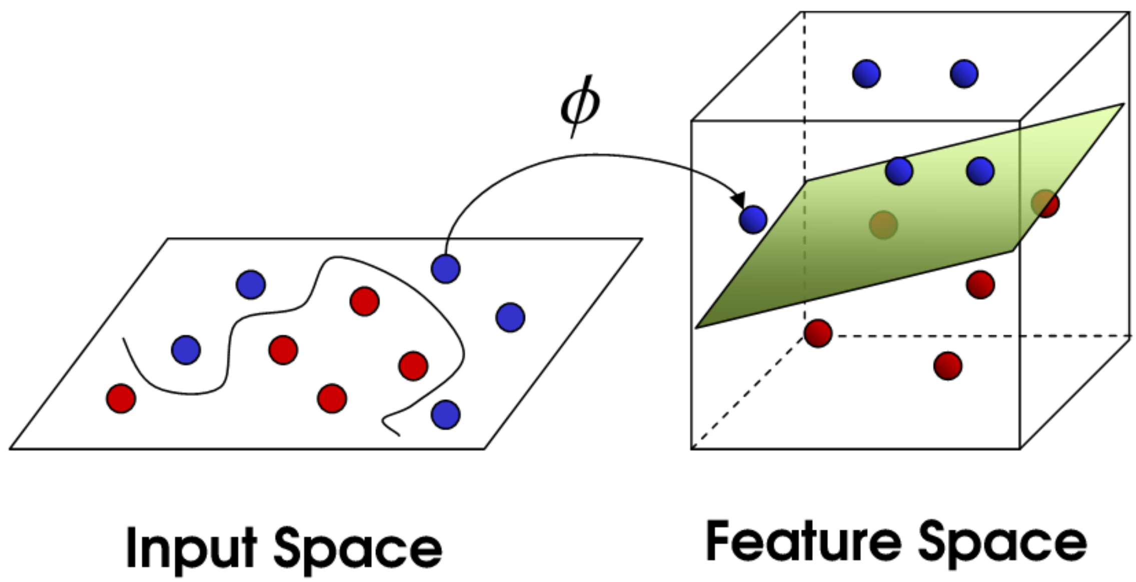
For a linear SVM, $y(x) = w^T x + b$.

We can just as well work in an alternate feature space: $\tilde{y}(x) = w^T \phi(x) + b$.

In [19]:
```
print "http://i.imgur.com/Wuxy0.png"
Image("http://i.imgur.com/Wuxy0.png")
```

```
http://i.imgur.com/Wuxy0.png
```

Out[19]:

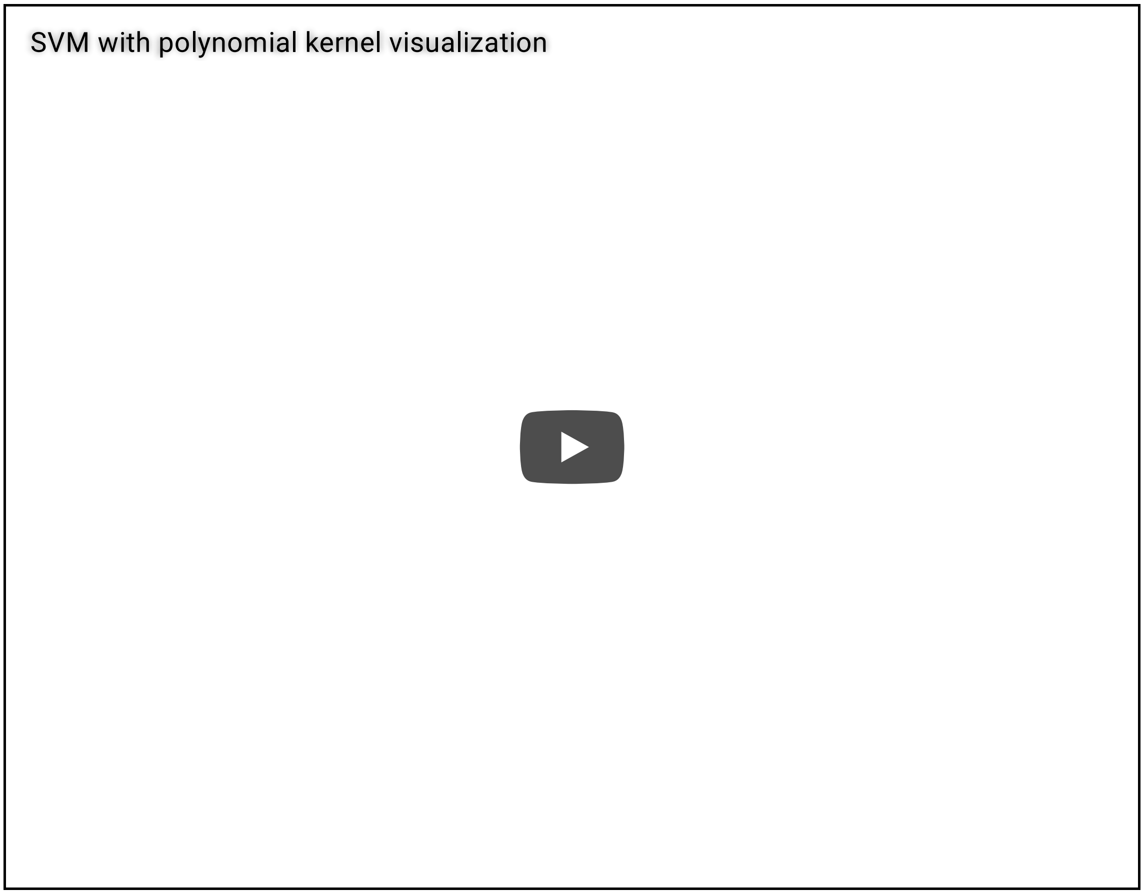


Non-linear SVMs

In [27]:
```
print "http://www.youtube.com/watch?v=3liCbRZPrZA"
YouTubeVideo("3liCbRZPrZA", width=900, height=600)
```

```
http://www.youtube.com/watch?v=3liCbRZPrZA
```

Out[27]:



SVMs vs Logistic Regression

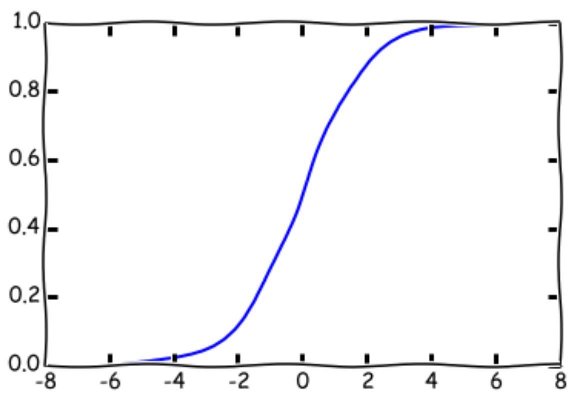
Logistic Regression

In [30]:

```
import matplotlib.pyplot as plt
plt.xkcd()
x = linspace(-8, 8)
y = 1/(1 + np.exp(-x))
plt.plot(x, y)
```

Out[30]:

```
[<matplotlib.lines.Line2D at 0x4558310>]
```



Logistic Regression

- Assign probability to each outcome

$$P(y = 1|x) = \sigma(w^T x + b)$$

- Train to maximize likelihood

$$\mathcal{L}(w) = \prod_{n=1}^{N} \sigma(w^T x_n + b)^{y_n} (1 - \sigma(w^T x_n + b))^{1-y_n}$$

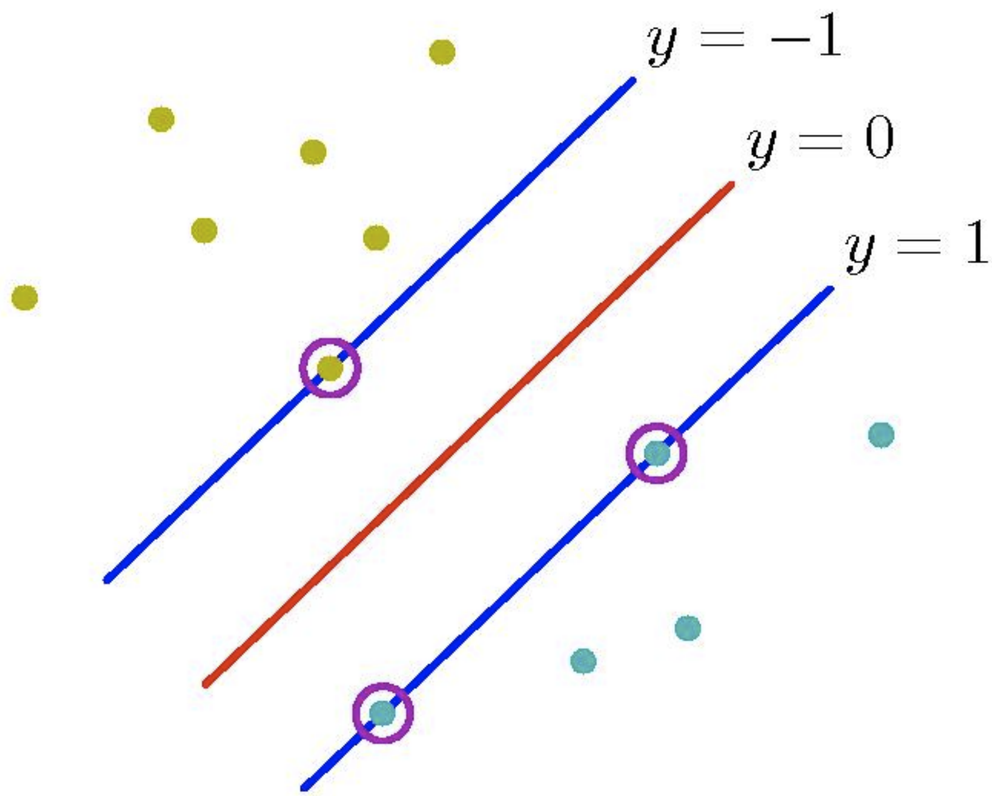
- Linear decision boundary

$$\hat{y} = I[w^T x + b \geq 0]$$

SVMs

In [31]:
```
Image("tight.png")
```

Out[31]:



SVMs

- Enforce a margin of separation
  $y_n(w^T x_n + b) \geq 1, \text{ for } n = 1 \dots N$

- Train to find the maximum margin
  $$\min \frac{1}{2}||w||^2$$
  $\text{s.t. } (2y_n - 1)(w^T x_n + b) \geq 1, \text{ for } n = 1 \dots N$

- Linear decision boundary
  $\hat{y} = I[w^T x + b \geq 0]$

Comparison

- **Logistic regression** wants to maximize the probability of the data.
  - The greater the distance from each point to the decision boundary, the better.

- **SVMs** want to maximize the distance from the closest points to the decision boundary.
  - Doesn't care about points that aren't support vectors.

A Different Take

Consider an alternate form of the logistic regression decision function:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x) \geq P(y=0|x) \\ 0 & \text{otherwise} \end{cases}$$

$$P(y=1|x) \propto \exp(w^T x + b)$$

$$P(y=0|x) \propto 1$$

A Different Take

Suppose we don't actually care about the probabilities. All we want to do is make the right decision.

We can put a constraint on the likelihood ratio, for some constant $c > 1$:

$$\frac{P(y=1|x_n)}{P(y=0|x_n)} \geq c$$

A Different Take

Take the log of both sides:

$$\log P(y=1|x_n) - \log P(y=0|x_n) \geq \log c$$

Recalling that $P(y=1|x_n) \propto \exp(w^T x_n + b)$ and $P(y=0|x_n) \propto 1$:

$$w^T x_n + b - 0 \geq \log c$$
$$w^T x_n + b \geq \log c$$

But c is arbitrary, so set it s.t. $\log c = 1$:

$$w^T x_n + b \geq 1$$

A Different Take

So now we have $(2y_n - 1)(w^T x_n + b) \geq 1, \text{for } n = 1 \ldots N$. But this may not have a unique solution, so put a quadratic penalty on the weights to make the solution unique:

$$\min \frac{1}{2} ||w||^2$$
$$\text{s.t. } (2y_n - 1)(w^T x_n + b) \geq 1, \text{ for } n = 1 \ldots N$$

By asking logistic regression to make the right **decisions** instead of maximizing the **probability** of the data, we derived an SVM.

Likelihood Ratio

The **likelihood ratio** drives this derivation:

$$r = \frac{P(y=1|x)}{P(y=0|x)} = \frac{\exp(w^T x + b)}{1} = \exp(w^T x + b)$$

Different classifiers assign **different costs** to r.

LR Cost

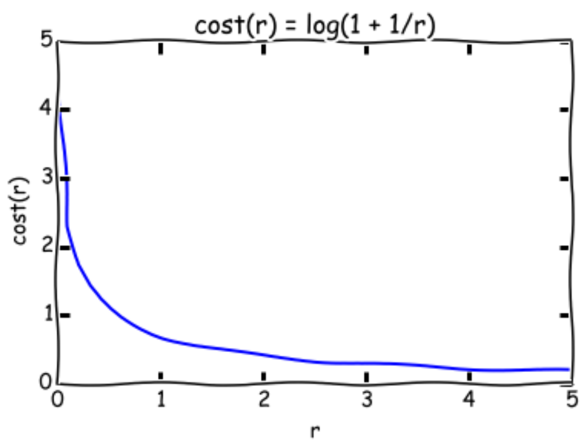
Choose $\text{cost}(r) = \log\left(1 + \frac{1}{r}\right)$

In [41]:

```
import matplotlib.pyplot as plt
plt.xkcd()
r = linspace(1e-2, 5)
cost_r = np.log(1 + 1/r)
plt.plot(r, cost_r)
plt.xlabel("r")
plt.ylabel("cost(r)")
plt.title("cost(r) = log(1 + 1/r)")
```

Out[41]:

```
<matplotlib.text.Text at 0x58bae10>
```



LR Cost

$$\begin{aligned}\log\left(1+\frac{1}{r}\right) &= \log\bigl(1+\exp(-(w^Tx+b))\bigr) \\ &= -\log\frac{1}{1+\exp(-(w^Tx+b))} \\ &= -\log\sigma(w^Tx+b)\end{aligned}$$

Minimizing $\text{cost}(r)$ is the same as minimizing the negative log-likelihood objective for logistic regression!

SVM with Slack Variables

If the data is not linearly separable, we can introduce slack variables.

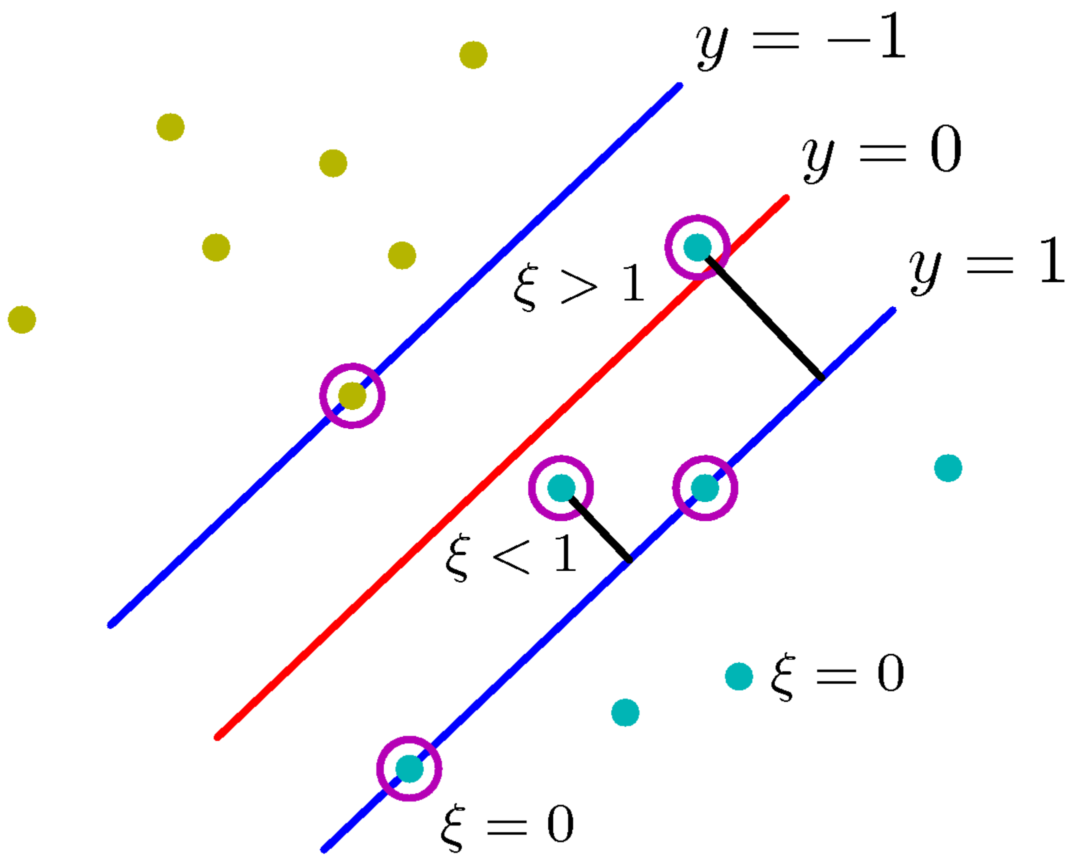
$$\min \frac{1}{2}||w||^2 + C\sum_{n=1}^{N}\xi_n$$
$$\text{s.t. } (2y_n - 1)(w^T x_n + b) \geq 1 - \xi_n, \text{ for } n = 1 \ldots N$$
$$\text{and } \xi_n \geq 0, \text{ for } n = 1 \ldots N$$

SVM with Slack Variables

In [42]:

```
Image("http://research.microsoft.com/en-us/um/people/cmbishop/PRML/prmlfigs-png/Figure7.3.png")
```

Out[42]:



SVM Cost

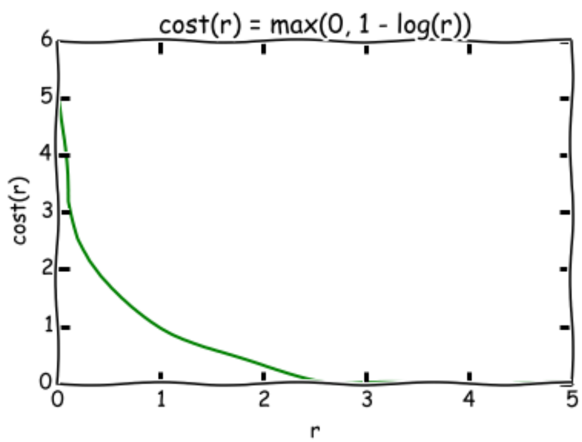
Choose $\text{cost}(r) = \max(0, 1 - \log(r)) = \max(0, 1 - (w^T x + b))$

In [49]:

```
import matplotlib.pyplot as plt
plt.xkcd()
r = linspace(1e-2, 5)
cost_r = 1 - np.log(r)
cost_r[cost_r < 0] = 0
plt.plot(r, cost_r, 'g')
plt.xlabel("r")
plt.ylabel("cost(r)")
plt.title("cost(r) = max(0, 1 - log(r))")
```

Out[49]:

```
<matplotlib.text.Text at 0x624fed0>
```



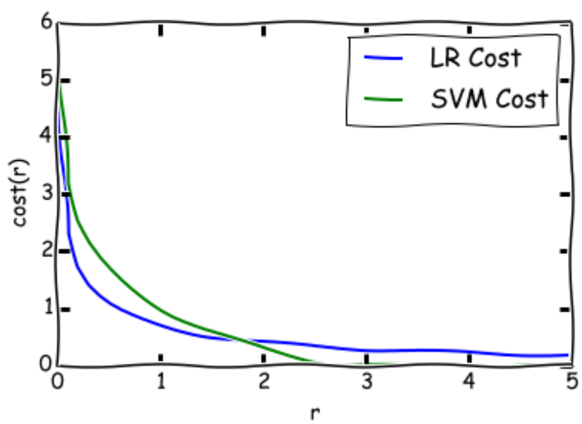
Plotted in terms of r

In [57]:

```
import matplotlib.pyplot as plt
plt.xkcd()
r = linspace(1e-2, 5)
lr_cost_r = np.log(1 + 1/r)
svm_cost_r = 1 - np.log(r)
svm_cost_r[svm_cost_r < 0] = 0
plt.plot(r, lr_cost_r, 'b', label="LR Cost")
plt.plot(r, svm_cost_r, 'g', label="SVM Cost")
plt.xlabel("r")
plt.ylabel("cost(r)")
plt.legend(loc="best")
```

Out[57]:

```
<matplotlib.legend.Legend at 0x6dfe390>
```



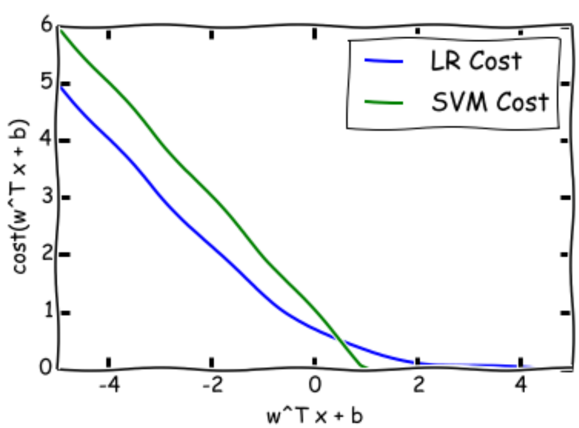
Plotted in terms of $w^T x + b$

In [64]:

```
plt.xkcd()
x = linspace(-5, 5)
lr_cost_x = np.log(1 + 1/x)
lr_cost_x = - np.log(1 / (1 + np.exp(-x)))
svm_cost_x = 1 - x
svm_cost_x[svm_cost_x < 0] = 0
plt.plot(x, lr_cost_x, 'b', label="LR Cost")
plt.plot(x, svm_cost_x, 'g', label="SVM Cost")
plt.xlabel("w^T x + b")
plt.ylabel("cost(w^T x + b)")
plt.xlim([-5,5])
plt.legend(loc="best")
```

Out[64]:

```
<matplotlib.legend.Legend at 0x7b36c90>
```



Exploiting the Connection between LR and SVMs

Kernel Trick for LR

In the dual form, the SVM decision boundary is

$$y(x) = w^T \phi(x) + b = \sum_{n=1}^{N} \alpha_n t_n K(x, x_n) + b = 0$$

We could plug this into the LR cost:

$$\log\left(1 + \exp\left(-\sum_{n=1}^{N} \alpha_n t_n K(x, x_n) - b\right)\right)$$

Multi-class SVMS

Recall multi-class logistic regression

$$P(y = i|x) = \frac{\exp(w_i^T x + b_i)}{\sum_k \exp(w_k^T x + b_k)}$$

Multi-class SVMS

Suppose that we just want the decision rule to satisfy

$$\frac{P(y=i|x)}{P(y=k|x)} \geq c, \text{ for } k \neq i$$

Taking logs as before,

$$(w_i^T x + b_i) - (w_k^T x + b_k) \geq 1, \text{ for } k \neq i$$

Multi-class SVMS

Now we have the quadratic program for multi-class SVMs.

$$\min \frac{1}{2}||w||^2$$
$$\text{s.t. } (w_{y_n}^T x_n + b_{y_n}) - (w_k^T x_n + b_k) \geq 1, \text{ for } n = 1 \dots N, k \neq y_n$$

LR and SVMs are closely linked

- Both can be viewed as taking a probabilistic model and miminizing some cost associated with the likelihood ratio.
- This allows use to extend both models in principled ways.

Which to Use?

Logistic regression

- Gives calibrated probabilities that can be interpreted as confidence in a decision.
- Unconstrained, smooth objective.
- Can be used within Bayesian models.

SVMs

- No penalty for examples where the correct decision is made with sufficient confidence, which can lead to good generalization.
- Dual form gives sparse solutions when using the kernel trick, leading to better scalability.

In [ ]:

```

```