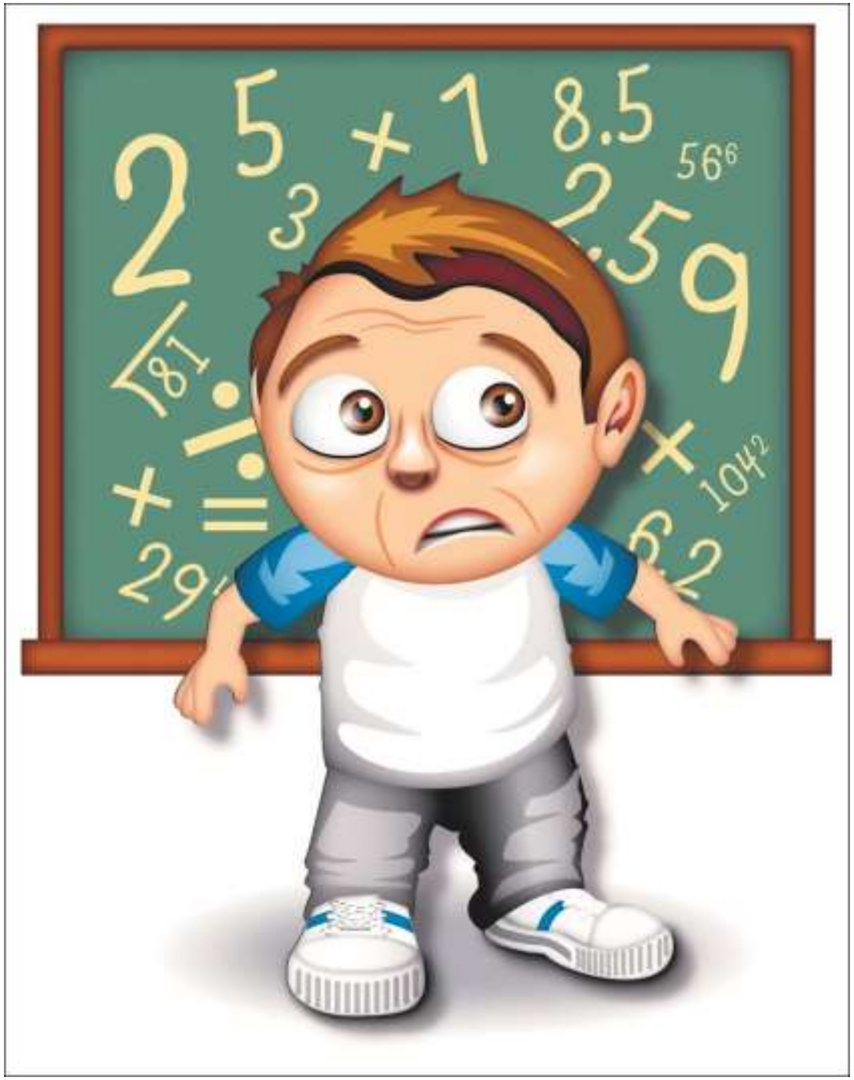
REGULARIZATION

David Kauchak
CS 451 – Fall 2013

Admin

Assignment 5

Math so far…



Model-based machine learning

1. pick a model

$$0 = b + \sum_{j=1}^{m} w_j f_j$$

2. pick a criteria to optimize (aka objective function)

$$\sum_{i=1}^{n} 1\left[y_i (w \cdot x_i + b) \le 0 \right]$$

3. develop a learning algorithm

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} 1\left[y_i (w \cdot x_i + b) \le 0 \right]$$

Find w and b that minimize the 0/1 loss

Model-based machine learning

1. pick a model

$$0 = b + \sum_{j=1}^{m} w_j f_j$$

2. pick a criteria to optimize (aka objective function)

$$\sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b))$$

use a convex surrogate loss function

3. develop a learning algorithm

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b))$$

Find w and b that minimize the surrogate loss

Finding the minimum

You're blindfolded, but you can see out of the bottom of the blindfold to the ground right by your feet. I drop you off somewhere and tell you that you're in a convex shaped valley and escape is at the bottom/minimum. How do you get out?

Gradient descent

- pick a starting point (w)
- repeat until loss doesn't decrease in all dimensions:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)

$$w_j = w_j - \eta \frac{d}{dw_j} loss(w)$$

Some maths

$$\frac{d}{dw_j} loss = \frac{d}{dw_j} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b))$$

$$= \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) \frac{d}{dw_j} - y_i(w \cdot x_i + b)$$

$$= \sum_{i=1}^{n} -y_i x_{ij} \exp(-y_i(w \cdot x_i + b))$$

Gradient descent

- pick a starting point (w)
- repeat until loss doesn't decrease in all dimensions:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)

$$w_j = w_j + \eta \sum_{i=1}^{n} y_i x_{ij} \exp(-y_i(w \cdot x_i + b))$$

What is this doing?

Perceptron learning algorithm!

repeat until convergence (or for some # of iterations):

for each training example (f_1, f_2, …, f_m, label):

$$prediction = b + \sum_{j=1}^{m} w_j f_j$$

if *prediction* * *label* ≤ 0: // they don't agree

for each w_j:

$w_j = w_j + f_j$*label

$b = b$ + label

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i(w \cdot x_i + b))$$

or

$$w_j = w_j + x_{ij} y_i c \quad \text{where} \quad c = \eta \exp(-y_i(w \cdot x_i + b))$$

The constant

$$c = \eta \exp(-y_i(w \cdot x_i + b))$$

learning rate   label   prediction

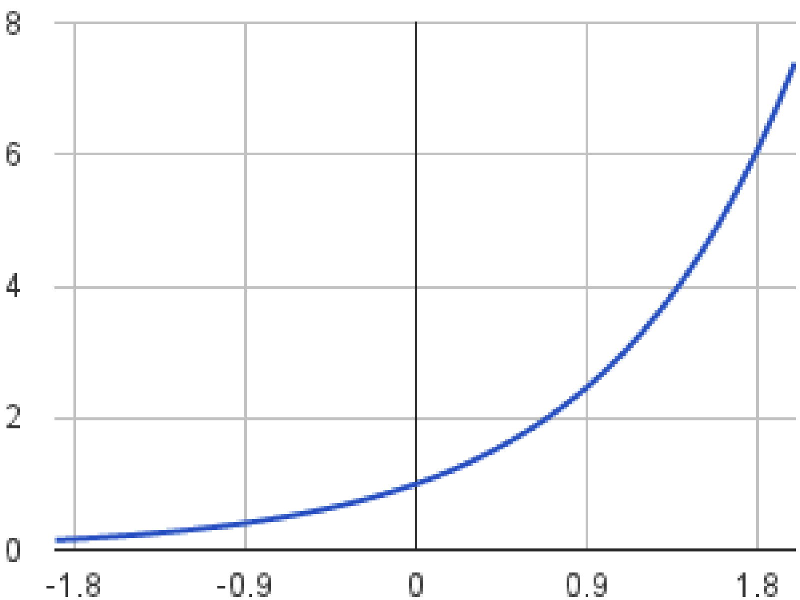
When is this large/small?

The constant

$$c = h\exp(- y_i(w \times x_i + b))$$

label

prediction

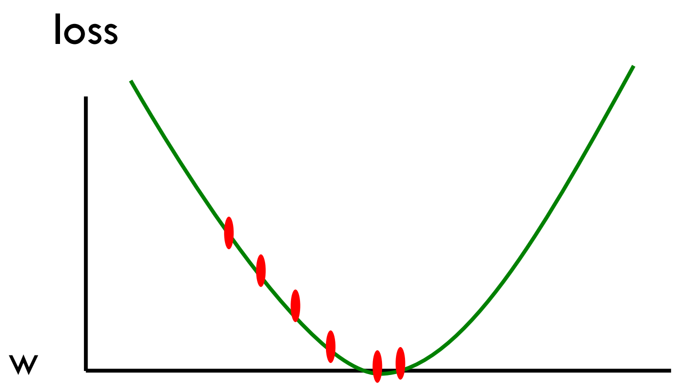


If they're the same sign, as the predicted gets larger there update gets smaller

If they're different, the more different they are, the bigger the update

One concern

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b))$$



What is this calculated on?

Is this what we want to optimize?

Perceptron learning algorithm!

repeat until convergence (or for some # of iterations):

for each training example (f_1, f_2, …, f_m, label):

$$prediction = b + \sum_{j=1}^{m} w_j f_j$$

~~if *prediction* * *label* ≤ 0: // they don't agree~~

for each w_i:

$w_i = w_i + f_i$*label

$b = b$ + label

Note: for gradient descent, we always update

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i(w \cdot x_i + b))$$

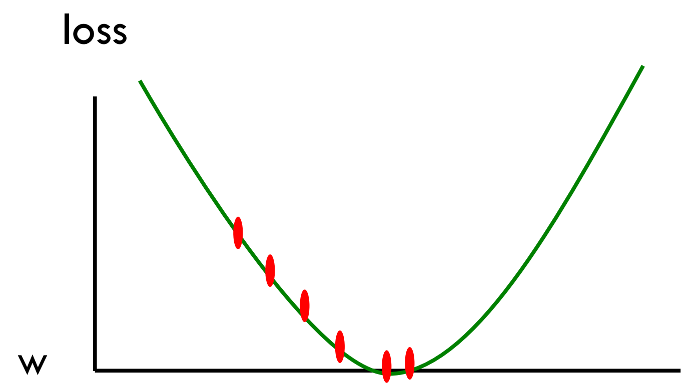
or

$$w_j = w_j + x_{ij} y_i c \quad \text{where} \quad c = \eta \exp(-y_i(w \cdot x_i + b))$$

One concern

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b))$$

We're calculating this on the **training set**



We still need to be careful about overfitting!

The min w,b on the training set is generally NOT the min for the test set

How did we deal with this for the perceptron algorithm?

Overfitting revisited: regularization

A regularizer is an additional criteria to the loss function to make sure that we don't overfit

It's called a regularizer since it tries to keep the parameters more normal/regular

It is a bias on the model forces the learning to prefer certain types of weights over others

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} loss(yy') + \lambda\ regularizer(w,b)$$

Regularizers

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

Should we allow all possible weights?

Any preferences?

What makes for a “simpler” model for a linear model?

Regularizers

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

Generally, we don't want huge weights

If weights are large, a small change in a feature can result in a large change in the prediction

Also gives too much weight to any one feature

Might also prefer weights of 0 for features that aren't useful

How do we encourage small weights? or penalize large weights?

Regularizers

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

How do we encourage small weights? or penalize large weights?

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} loss(yy') + \lambda \; regularizer(w,b)$$

Common regularizers

sum of the weights

$$r(w,b) = \sum_{w_j} |w_j|$$

sum of the squared weights

$$r(w,b) = \sqrt{\sum_{w_j} |w_j|^2}$$

What's the difference between these?

Common regularizers

sum of the weights

$$r(w,b) = \sum_{w_j} \left| w_j \right|$$

sum of the squared weights

$$r(w,b) = \sqrt{\sum_{w_j} \left| w_j \right|^2}$$

Squared weights penalizes large values more

Sum of weights will penalize small values more

p-norm

sum of the weights (1-norm)

$$r(w,b) = \sum_{w_j} \left| w_j \right|$$

sum of the squared weights (2-norm)

$$r(w,b) = \sqrt{\sum_{w_j} \left| w_j \right|^2}$$

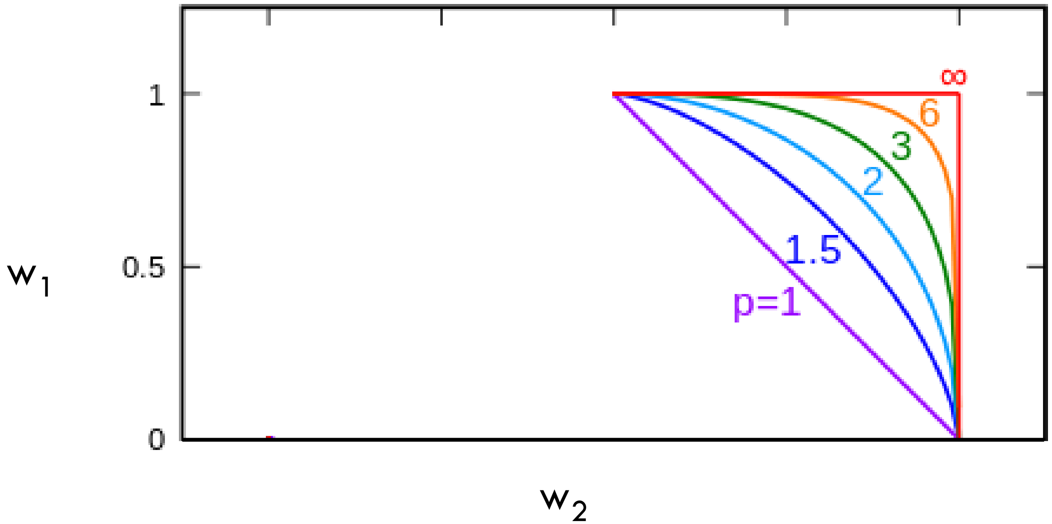
p-norm

$$r(w,b) = \sqrt[p]{\sum_{w_j} \left| w_j \right|^p} = \|w\|^p$$

Smaller values of p ($p < 2$) encourage sparser vectors

Larger values of p discourage large weights more

p-norms visualized

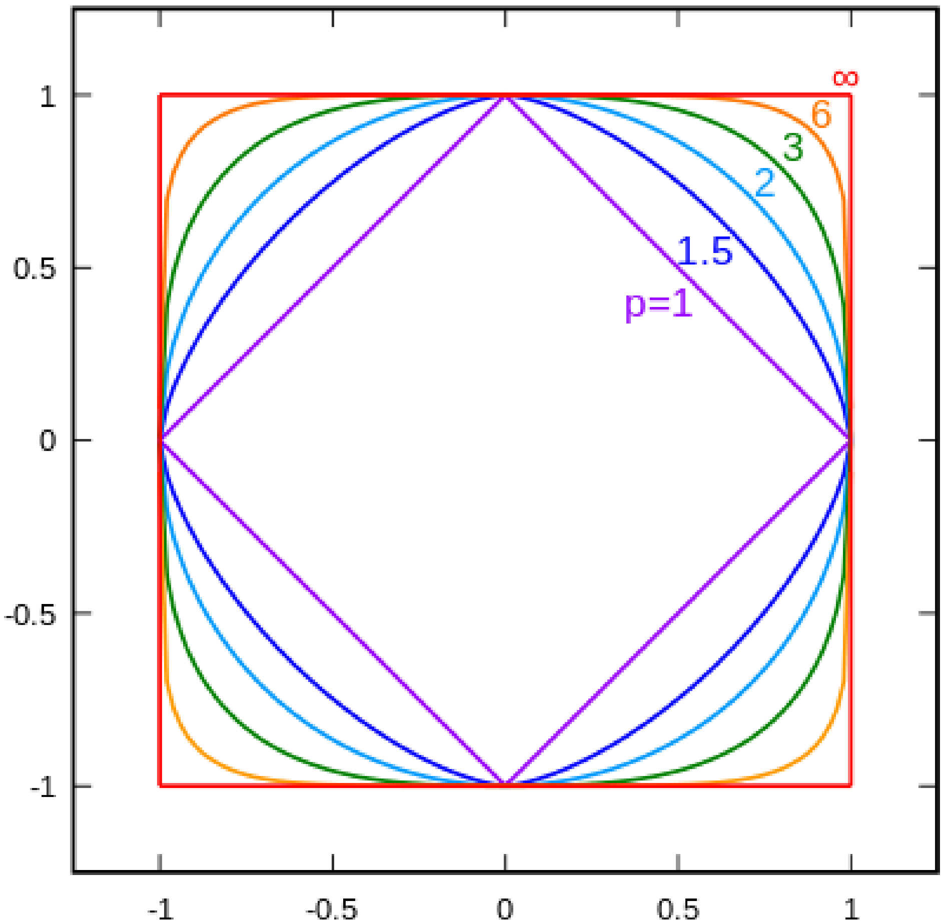


lines indicate penalty = 1

For example, if $w_1 = 0.5$

p	w_2
1	0.5
1.5	0.75
2	0.87
3	0.95
∞	1

p-norms visualized



all p-norms penalize larger weights

p < 2 tends to create sparse (i.e. lots of 0 weights)

p > 2 tends to like similar weights

Model-based machine learning

1. pick a model

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

2. pick a criteria to optimize (aka objective function)

$$\sum_{i=1}^{n} loss(yy') + \lambda\, regularizer(w)$$

3. develop a learning algorithm

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} loss(yy') + \lambda\, regularizer(w)$$

Find w and b that minimize

Minimizing with a regularizer

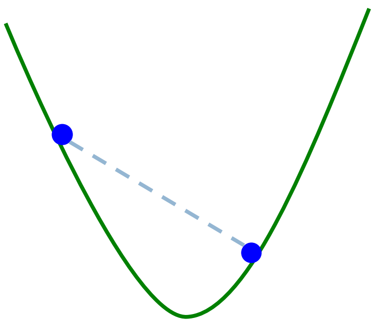
We know how to solve convex minimization problems using gradient descent:

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} loss(yy')$$

If we can ensure that the loss + regularizer is convex then we could still use gradient descent:

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \underbrace{loss(yy') + \lambda\, regularizer(w)}_{\text{make convex}}$$

Convexity revisited



One definition: The line segment between any two points on the function is *above* the function

Mathematically, *f* is convex if for all x_1, x_2:

$$f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2) \quad \forall \quad 0 < t < 1$$

the value of the function at some point between x_1 and x_2

the value at some point on the **line segment** between x_1 and x_2

Adding convex functions

Claim: If *f* and *g* are convex functions then so is the function z=*f*+*g*

Prove:

$$z(tx_1+(1-t)x_2) \le tz(x_1)+(1-t)z(x_2) \quad \forall \quad 0<t<1$$

Mathematically, *f* is convex if for all x_1, x_2:

$$f(tx_1+(1-t)x_2) \le tf(x_1)+(1-t)f(x_2) \quad \forall \quad 0<t<1$$

Adding convex functions

By definition of the sum of two functions:

$$z(tx_1 + (1-t)x_2) = f(tx_1 + (1-t)x_2) + g(tx_1 + (1-t)x_2)$$

$$\begin{aligned} tz(x_1) + (1-t)z(x_2) &= tf(x_1) + tg(x_1) + (1-t)f(x_2) + (1-t)g(x_2) \\ &= tf(x_1) + (1-t)f(x_2) + tg(x_1) + (1-t)g(x_2) \end{aligned}$$

Then, given that:

$$f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2)$$

$$g(tx_1 + (1-t)x_2) \le tg(x_1) + (1-t)g(x_2)$$

We know:

$$f(tx_1 + (1-t)x_2) + g(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2) + tg(x_1) + (1-t)g(x_2)$$

So: $z(tx_1 + (1-t)x_2) \le tz(x_1) + (1-t)z(x_2)$

Minimizing with a regularizer

We know how to solve convex minimization problems using gradient descent:

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} loss(yy')$$

If we can ensure that the loss + regularizer is convex then we could still use gradient descent:

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \underbrace{loss(yy') + \lambda\, regularizer(w)}_{\text{convex as long as both loss and regularizer are convex}}$$

p-norms are convex

$$r(w,b) = \sqrt[p]{\sum_{w_j} \left|w_j\right|^p} = \left\|w\right\|^p$$

p-norms are convex for p >= 1

Model-based machine learning

1. pick a model

$$0 = b + \sum_{j=1}^{n} w_j f_j$$

2. pick a criteria to optimize (aka objective function)

$$\sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

3. develop a learning algorithm

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

Find w and b that minimize

Our optimization criterion

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

Loss function: penalizes examples where the prediction is different than the label

Regularizer: penalizes large weights

Key: this function is convex allowing us to use gradient descent

Gradient descent

- pick a starting point (w)
- repeat until loss doesn't decrease in all dimensions:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)

$$w_i = w_i - \eta \frac{d}{dw_i}(loss(w) + regularizer(w, b))$$

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

Some more maths

$$\frac{d}{dw_j} objective = \frac{d}{dw_j} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \frac{\lambda}{2}\|w\|^2$$

$$\vdots \qquad \text{(some math happens)}$$

$$= -\sum_{i=1}^{n} y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) + \lambda w_j$$

Gradient descent

- pick a starting point (w)
- repeat until loss doesn't decrease in all dimensions:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)

$$w_i = w_i - \eta \frac{d}{dw_i}(loss(w) + regularizer(w, b))$$

$$w_j = w_j + \eta \sum_{i=1}^{n} y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) - \eta \lambda w_j$$

The update

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) - \eta \lambda w_j$$

learning rate

direction to update

constant: how far from wrong

regularization

What effect does the regularizer have?

The update

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) - \eta\lambda w_j$$

learning rate

direction to update

constant: how far from wrong

regularization

$$\left.\begin{array}{l}\text{If } w_i \text{ is positive, reduces } w_i \\ \text{If } w_i \text{ is negative, increases } w_i\end{array}\right\} \text{ moves } w_i \text{ towards } 0$$

L1 regularization

$$\operatorname{argmin}_{w,b} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \|w\|$$

$$\frac{d}{dw_j} objective = \frac{d}{dw_j} \sum_{i=1}^{n} \exp(-y_i(w \cdot x_i + b)) + \lambda \|w\|$$

$$= -\sum_{i=1}^{n} y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) + \lambda\, sign(w_j)$$

L1 regularization

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) - \eta\lambda\, sign(w_j)$$

learning rate

direction to update

regularization

constant: how far from wrong

What effect does the regularizer have?

L1 regularization

$$w_j = w_j + \eta y_i x_{ij} \exp(-y_i(w \cdot x_i + b)) - \eta\lambda\, sign(w_j)$$

learning rate

direction to update

constant: how far from wrong

regularization

If w_i is positive, reduces by a constant
If w_i is negative, increases by a constant

moves w_i towards 0 ***regardless of magnitude***

Regularization with p-norms

L1:

$$w_j = w_j + \eta(loss\_correction - \lambda\, sign(w_j))$$

L2:

$$w_j = w_j + \eta(loss\_correction - \lambda\, w_j)$$

Lp:

$$w_j = w_j + \eta(loss\_correction - \lambda\, c w_j^{p-1})$$

How do higher order norms affect the weights?

Regularizers summarized

L1 is popular because it tends to result in sparse solutions (i.e. lots of zero weights)

- However, it is not differentiable, so it only works for gradient descent solvers

L2 is also popular because for some loss functions, it can be solved directly (no gradient descent required, though often iterative solvers still)

Lp is less popular since they don't tend to shrink the weights enough

The other loss functions

Without regularization, the generic update is:

$$w_j = w_j + h y_i x_{ij} c$$

where

$$c = \exp(-y_i(w \times x_i + b)) \qquad \text{exponential}$$

$$c = 1[yy' < 1] \qquad \text{hinge loss}$$

$$w_j = w_j + h(y_i - (w \times x_i + b) x_{ij}) \qquad \text{squared error}$$

Many tools support these different combinations

Look at scikit learning package:

http://scikit-learn.org/stable/modules/sgd.html

Common names

(Ordinary) Least squares: squared loss

Ridge regression: squared loss with L2 regularization

Lasso regression: squared loss with L1 regularization

Elastic regression: squared loss with L1 AND L2 regularization

Logistic regression: logistic loss

Real results