CMSC 510 – L09
Regularization Methods for Machine Learning

Instructor:
Dr. Tom Arodz

Recap: MLE for P(y|x)

- Let's say we have a probability distribution P(y|x) of a certain shape, and the distribution is parameterized by a vector w, so P(y|x)=P(y|x,w)
- How to estimate w?
  - We have a training set S with m samples
  - We could do maximum likelihood estimation of w

$$\max P(y_1, y_2, \ldots, y_m \,|\, x_1, x_2, \ldots, x_m, w)$$

    - For which **w** are the observed y's for x's most likely?
    - We don't need to know anything about probability of x's
      - we're not estimating P(x,y), just P(y|x)
  - Samples are i.i.d: they're independent, so:

$$P(y_1, y_2, \ldots, y_m \,|\, x_1, x_2, \ldots, x_m, w) = \Pi_{i=1}^{m} P_w(y_i \,|\, x_i, w)$$

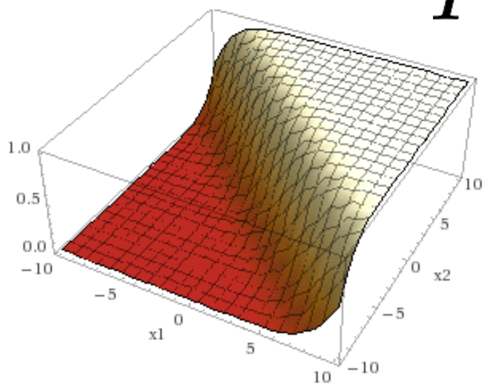
  - ML estimate of **w:**

    w that maximizes: $\boxed{\max \Pi_{i=1}^{m} P(y_i \,|\, x_i, w)}$

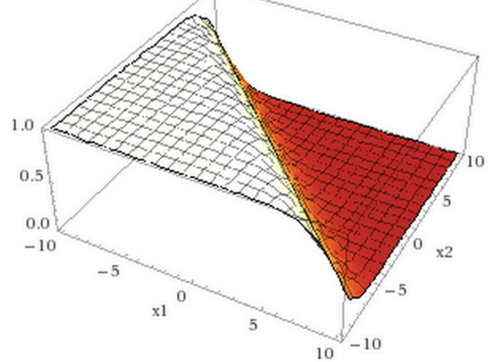
Recap: MLE / Logistic regression

- Under the assumption that class conditional probabilities depend on an unknown **w** in this way:

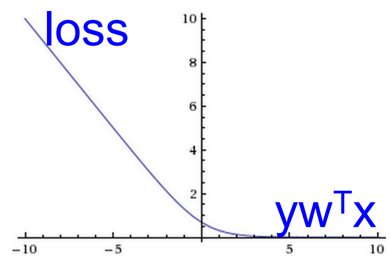
$$P(y_i\,|x_i, w\,) = a(y_i w^T x_i) = \frac{1}{1+\mathrm{e}^{-y_i w^T x_i}}$$



$P(+1\,|x_i, w\,)$



$P(-1\,|x_i, w\,) = 1 - P(+1\,|x_i, w\,)$



- Maximum likelihood estimate of w is:

$$\max \Pi_{i=1}^{m} P(y_i\,|x_i, w\,)$$

- Solve this instead:

$$\min -\frac{1}{m} \ln \Pi_{i=1}^{m} a(y_i w^T x_i)$$

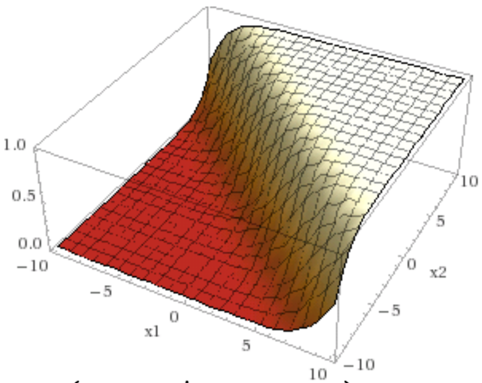
- Or this:

$$\min \frac{1}{m} \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})$$

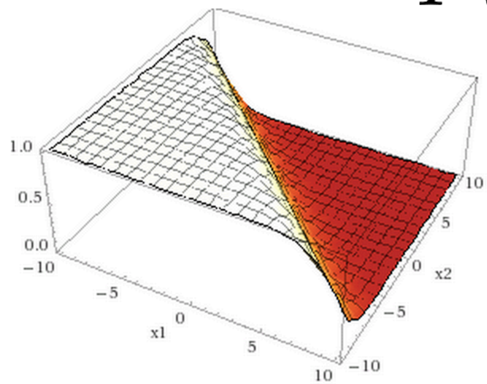
Back to our view of LR

- Under the assumption that class conditional probabilities depend on an unknown **w** in this way:

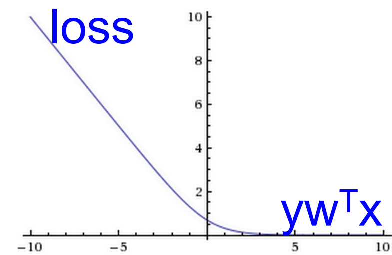
$$P(y_i \,|\, x_i, w) = \frac{1}{1+\mathrm{e}^{-y_i w^T x_i}}$$



$P(+1 \,|\, x_i, w)$



$P(-1 \,|\, x_i, w) = 1 - P(+1 \,|\, x_i, w)$



- Logistic regression minimizing the logistic loss: $\min \frac{1}{m} \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})$

- is a *maximum likelihood* estimate of **w** given the training set:

$$\max P(y_1, y_2, \ldots, y_m \,|\, x_1, x_2, \ldots, x_m, w)$$

Dealing with large weights

- Logistic regression is a *maximum likelihood* estimate of **w** given the training set:

$$\arg\max_{w} P(y_1, y_2, \ldots, y_m \,|x_1, x_2, \ldots, x_m, w\,)$$
$$\arg\max_{w} \Pi_{i=1}^{m} P(y_i \,|x_i, w\,)$$

- *Maximum a posteriori (MAP)* estimate of w is:

$$\arg\max_{w} P(y_1, y_2, \ldots, y_m \,|x_1, x_2, \ldots, x_m, w\,)P(w)$$
$$\arg\max_{w} P(w)\Pi_{i=1}^{m} P(y_i \,|x_i, w\,)$$

- We take into account our estimate of probability P(w) of each possible vector of weights w
  - Here, we can insert a belief that large w are unlikely: P(large w) is small, P(small w) is large

Dealing with large weights

- *Maximum a posteriori (MAP)* estimate of w is:

$$\arg\max_w P(y_1, y_2, ..., y_m \,|x_1, x_2, ..., x_m, w\,)P(w)$$
$$\arg\max_w P(w)\Pi_{i=1}^m P(y_i\,|x_i, w\,)$$

- Applying the standard log() trick:

$$\arg\max_w \ln P(w) + \sum_{i=1}^m \ln P(y_i\,|x_i, w\,)$$
$$\arg\min_w -\ln P(w) + \sum_{i=1}^m -\ln P(y_i\,|x_i, w\,)$$
$$\arg\min_w \ln \frac{1}{P(w)} + \sum_{i=1}^m \ln \frac{1}{P(y_i\,|x_i, w\,)}$$

Dealing with large weights

- Logistic regression was a *maximum likelihood* estimate of **w** given the training set, for a specific form of P(y|x,w): $P(y_i\,|x_i, w\,) = \frac{1}{1+\mathrm{e}^{-y_i w^T x_i}}$

$$\arg\min_w \sum_{i=1}^{m} \ln(1+\mathrm{e}^{-y_i w^T x_i})$$

- *Maximum a posteriori (MAP)* estimate of **w** for the same form of P(y|x,w) is:

$$\arg\min_w \left[\ln \frac{1}{P(w)}\right] + \sum_{i=1}^{m} \ln(1+\mathrm{e}^{-y_i w^T x_i})$$

Dealing with large weights

- Let's assume: $P(y_i \,|x_i, w) = \frac{1}{1 + e^{-y_i w^T x_i}}$
- *Maximum a posteriori (MAP)* estimate of **w** given the training set is:

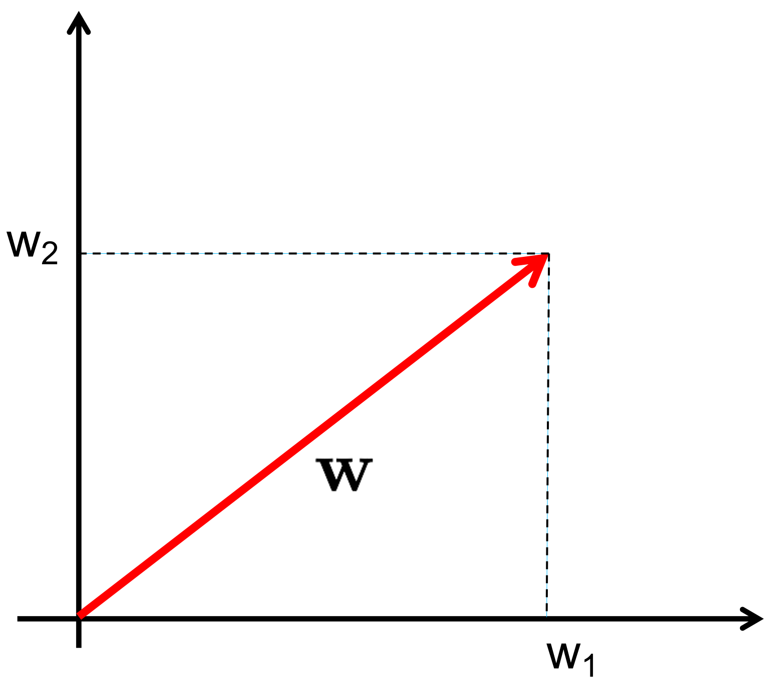
$$\arg\max_w P(w)\Pi_{i=1}^m P(y_i \,|x_i, w)$$

$$\arg\min_w \left[\ln \frac{1}{P(w)}\right] + \sum_{i=1}^m \ln(1 + e^{-y_i w^T x_i})$$

- P(w) = plausibility of this particular w
  - unless we have some evidence that large w are needed, we believe large w are unlikely to be truly necessary
    - More likely, they're artificial effect of correlated features
- P(w) should be: small for large w,
  large for small w

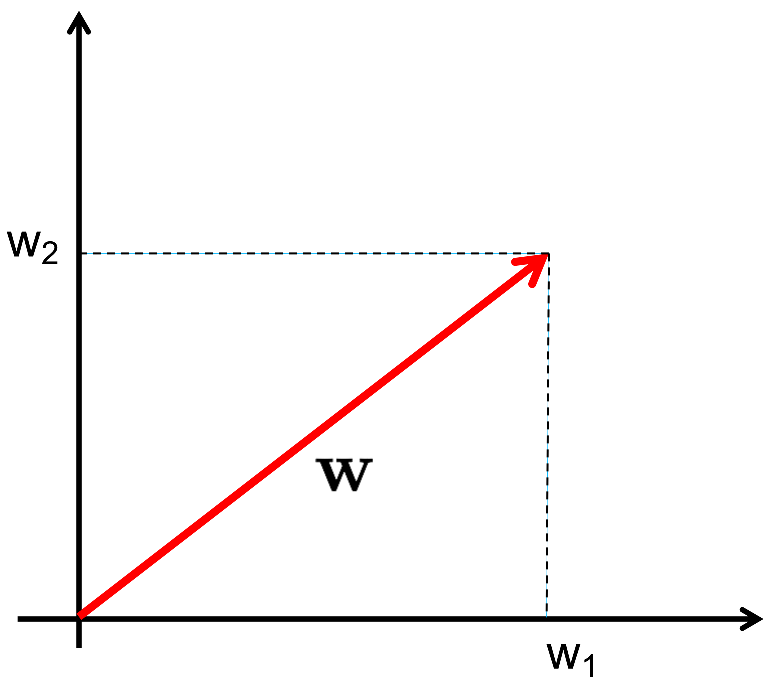
Dealing with large weights

- P(w) should be: small for large w,
  large for small w
- How to measure if w is large or small? $\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$



Dealing with large weights

- P(w) should be: small for large w,
  large for small w
- How to measure if w is large or small?



L_p norm:

$$\|w\|_p = \left(\sum_{f=1}^{F} |w_f|^p \right)^{1/p}$$

large w = large value of ||w||

we raise ||w|| to power p for convenience

$$\|w\|_p^p = \left(\sum_{f=1}^{F} |w_f|^p \right)$$

Dealing with large weights

- Let's assume: $P(y_i | x_i, w) = \frac{1}{1+\mathrm{e}^{-y_i w^T x_i}}$
- *Maximum a posteriori (MAP)* estimate of **w:**

$$\arg\min_w \left[\ln \frac{1}{P(w)}\right] + \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})$$

- P(w) should be: small for large L_p norm of w,
  large for small L_p norm of w

$$\|w\|_p^p = \left(\sum_{f=1}^{F} |w_f|^p\right)$$

- What form should P(w) take?
  - $P(w)=f(\ ||w||^p\ )$ where $0 \le ||w||^p \le \infty$
  - What would be a convenient choice for f()?

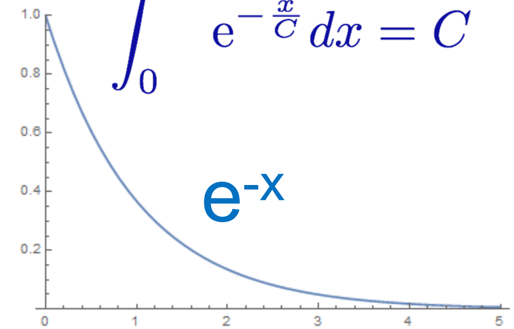
Dealing with large weights

- Let's assume: $P(y_i \,|x_i, w) = \frac{1}{1+e^{-y_i w^T x_i}}$
- *Maximum a posteriori (MAP)* estimate of **w:**

$$\arg\min_w \left[\ln \frac{1}{P(w)}\right] + \sum_{i=1}^{m} \ln(1 + e^{-y_i w^T x_i})$$

- P(w) should be: small for large w, large for small w

$$\int_0^\infty e^{-\frac{x}{C}} dx = C$$



L_p norm

large w = large norm

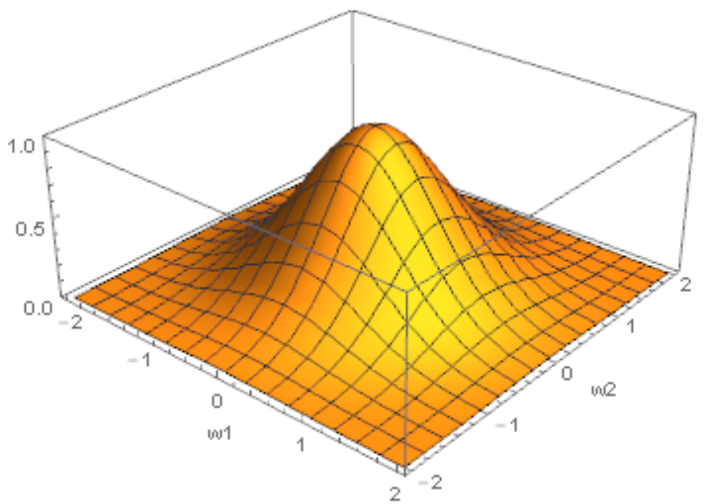
$$P(w) = \frac{1}{C} e^{-\frac{1}{C}||w||_p^p}$$

$$||w||_p^p = \left(\sum_{f=1}^{F} |w_f|^p\right)$$

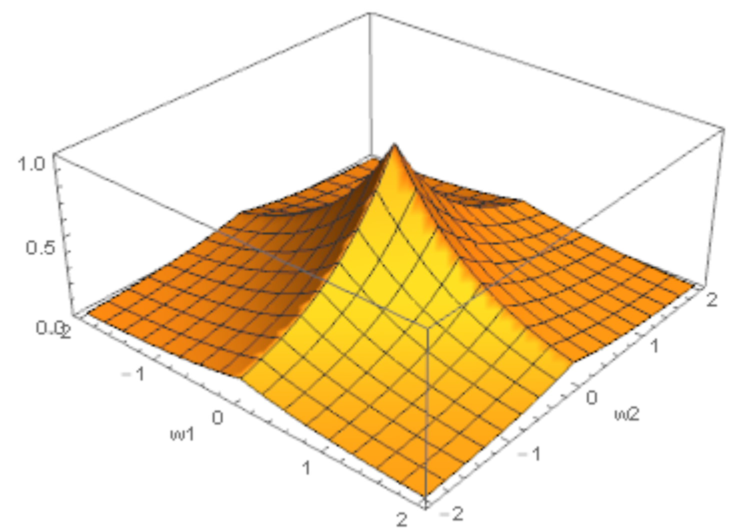
Dealing with large weights

- P(w) should be: small for large w,
  large for small w

$$\|w\|_p^p = \left(\sum_{f=1}^{F} |w_f|^p \right)$$



$$P(w) = \frac{1}{C} \mathrm{e}^{-\frac{1}{C}\|w\|_2^2}$$



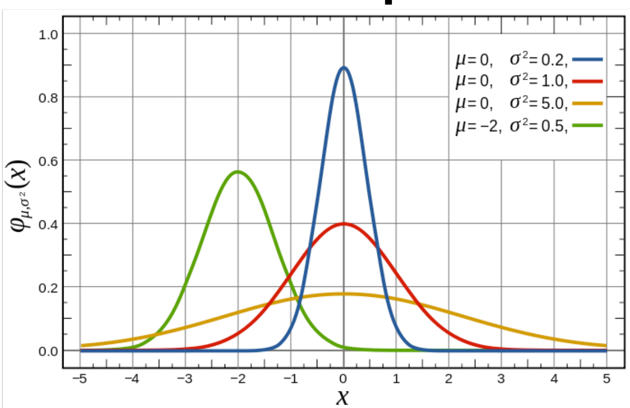
$$P(w) = \frac{1}{C} \mathrm{e}^{-\frac{1}{C}\|w\|_1}$$

Dealing with large weights

- Let's assume: $P(y_i \,|x_i, w\,) = \frac{1}{1+\mathrm{e}^{-y_i w^T x_i}}$
- *Maximum a posteriori (MAP)* estimate of **w:**

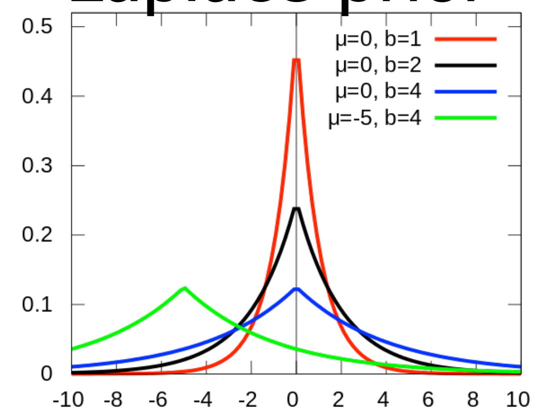
$$\arg\min_w \left[\ln \frac{1}{P(w)}\right] + \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})$$

- Gaussian prior



Constant C is standard deviation or its equivalent

Laplace prior



$$P(w) = \frac{1}{C}\mathrm{e}^{-\frac{1}{C}\|w\|_2^2}$$

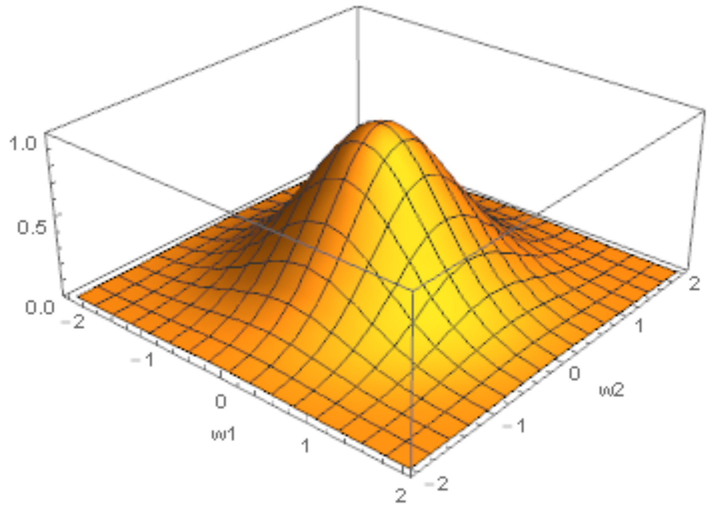
$$P(w) = \frac{1}{C}\mathrm{e}^{-\frac{1}{C}\|w\|_1}$$

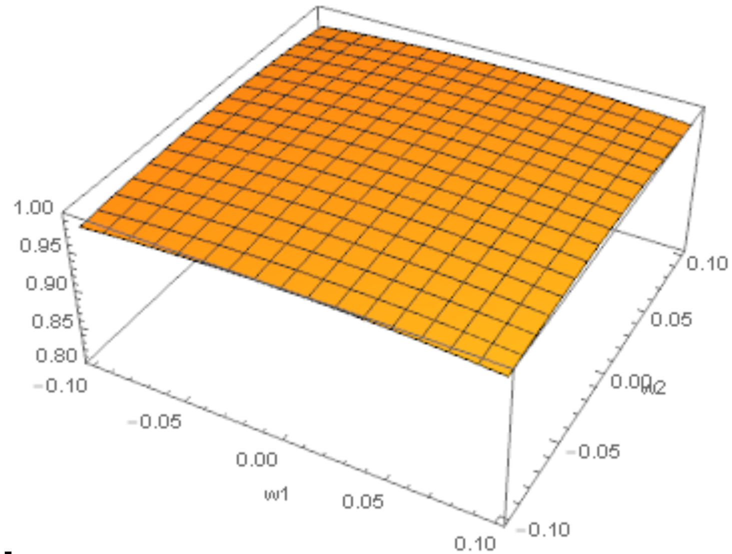
- Lower C => P(w) concentrated more around 0
- Lower C => promotes smaller feature weights w

L_1 vs L_2 regularization

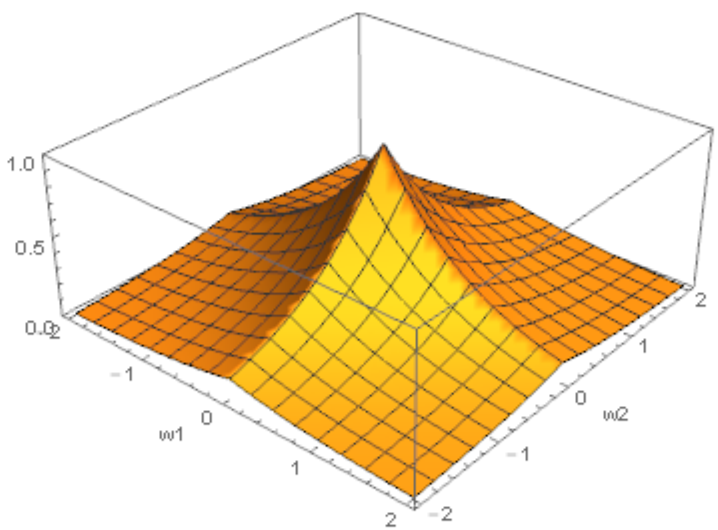
$$P(w) = \frac{1}{C} e^{-\frac{1}{C}\|w\|_p^p}$$

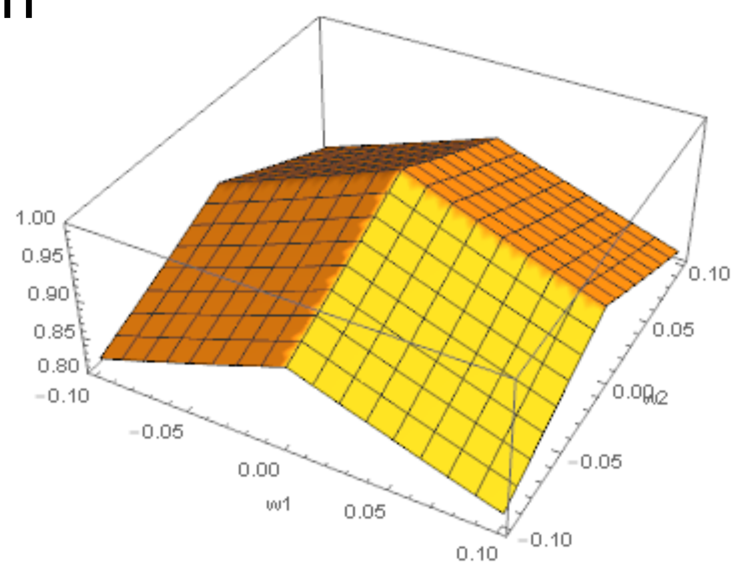
- P(w) for L_2 regularization





- P(w) for L_1 regularization





L_p regularization

- L_p regularization
$$\arg\min_{w} \left[\ln \frac{1}{P(w)} \right] + \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})$$
- What does the first term expand to?
- We have:
$$P(w) = \frac{1}{C} \mathrm{e}^{-\frac{1}{C} \|w\|_p^p}$$
- So:
$$\boxed{\ln \frac{1}{P(w)} = \frac{1}{C} \|w\|_p^p + \ln C}$$

Regularization

$$\boxed{\ln \frac{1}{P(w)} = \frac{1}{C}\|w\|_p^p + \ln C}$$

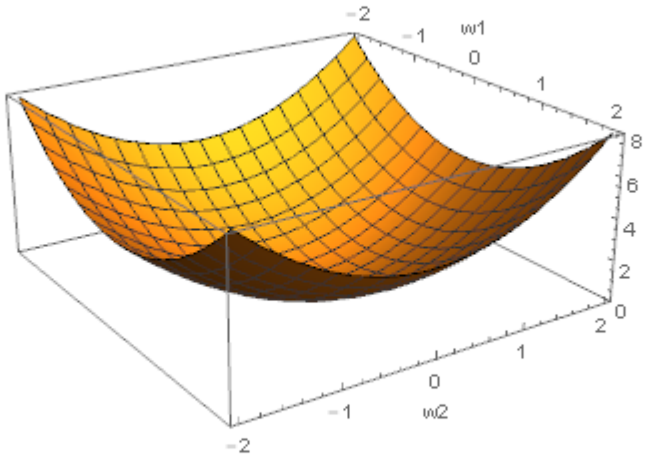
- Logistic regression with L_p regularization

$$\arg\min_w \left[\ln \frac{1}{P(w)}\right] + \sum_{i=1}^m \ln(1 + e^{-y_i w^T x_i})$$

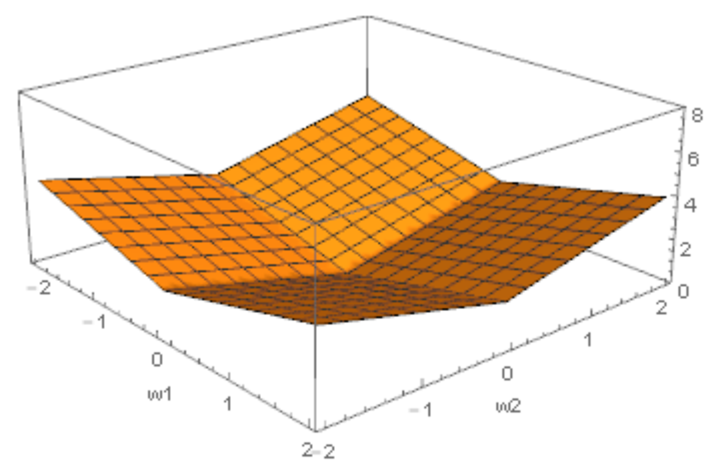
$$\arg\min_w \frac{1}{C}\|w\|_p^p + \sum_{i=1}^m \ln(1 + e^{-y_i w^T x_i})$$

- We're adding a penalty term $\frac{1}{C}\|w\|_p^p$
- The penalty term looks like this:

or this:



$$\|w\|_p^p = \left(\sum_{f=1}^F |w_f|^p\right)$$



$$\|w\|_2^2$$

$$\|w\|_1$$

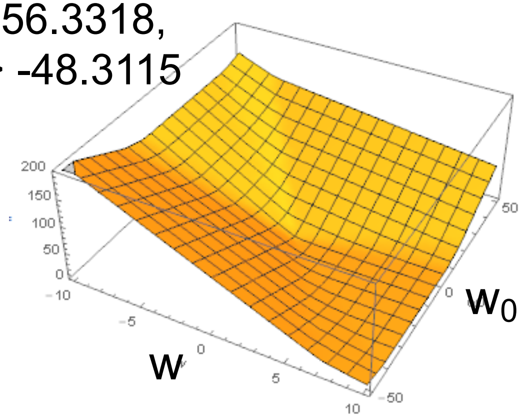
Regularization

- Logistic regression with L_p regularization

$$\arg\min_{w} \frac{1}{C}\|w\|_p^p + \sum_{i=1}^{m} \ln(1+\mathrm{e}^{-y_i w^T x_i})$$

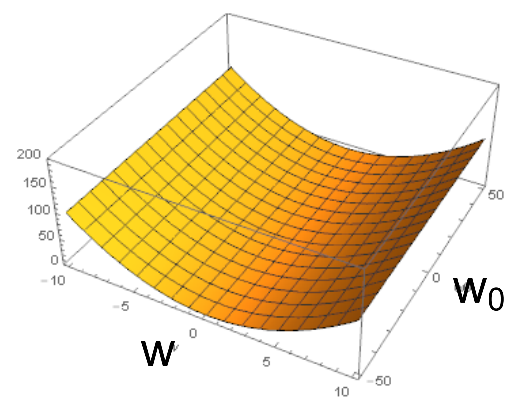
- We're adding a penalty term $\frac{1}{C}\|w\|_p^p$
- Algorithm minimizes:
  empirical risk of h() + penalty for complexity of h()

Minimum:
w -> 56.3318,
w_0 -> -48.3115



$$\sum_{i=1} \ln(1+\mathrm{e}^{-y_i w^T x_i})$$

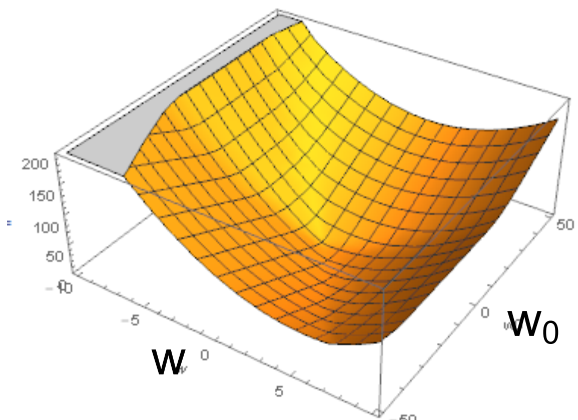
+



$$\|w\|_2^2$$

No penalty over w_0

=

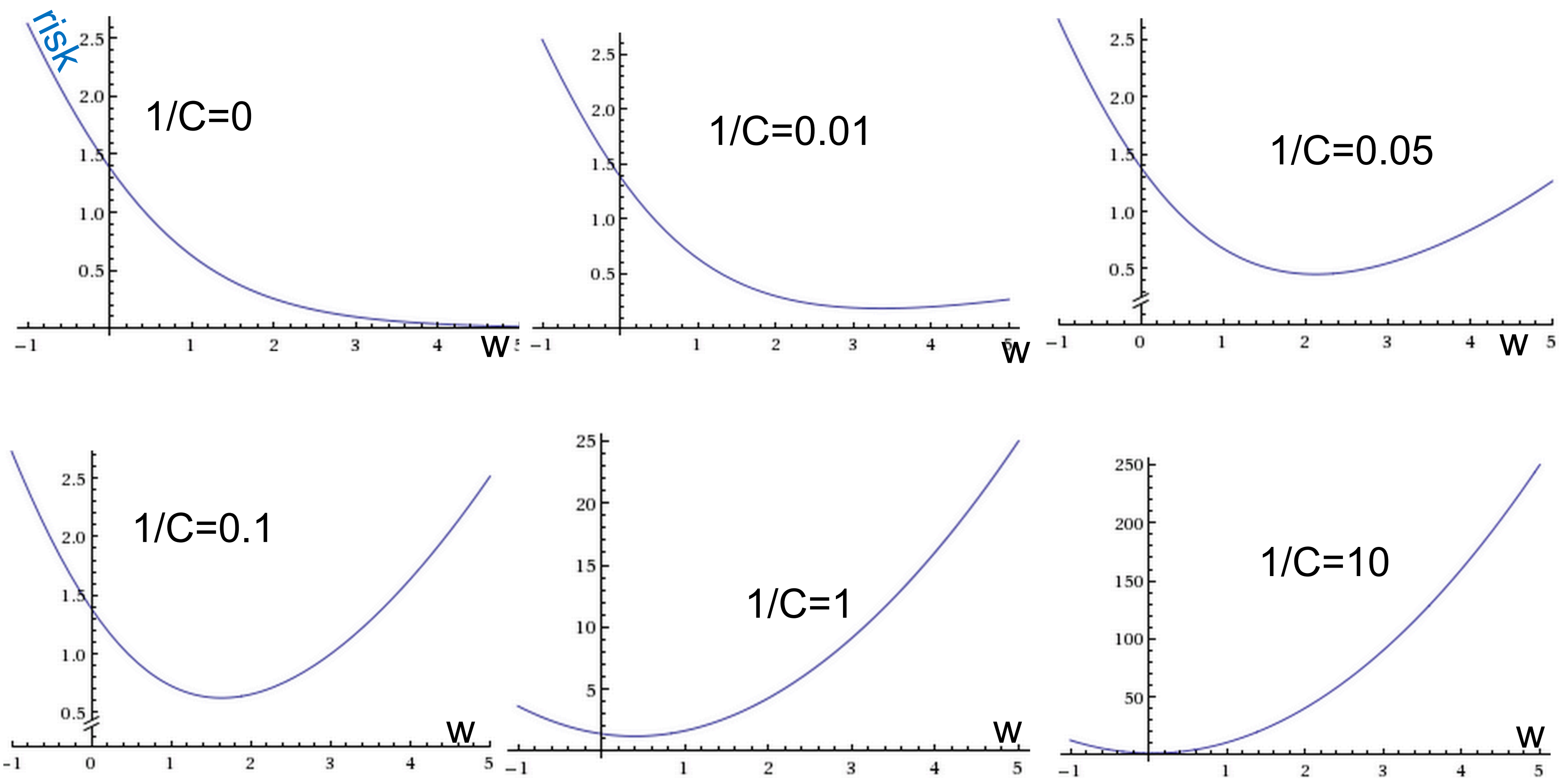


Minimum:
w -> 0.647365,
w_0 -> -1.14344

L_2 Regularization

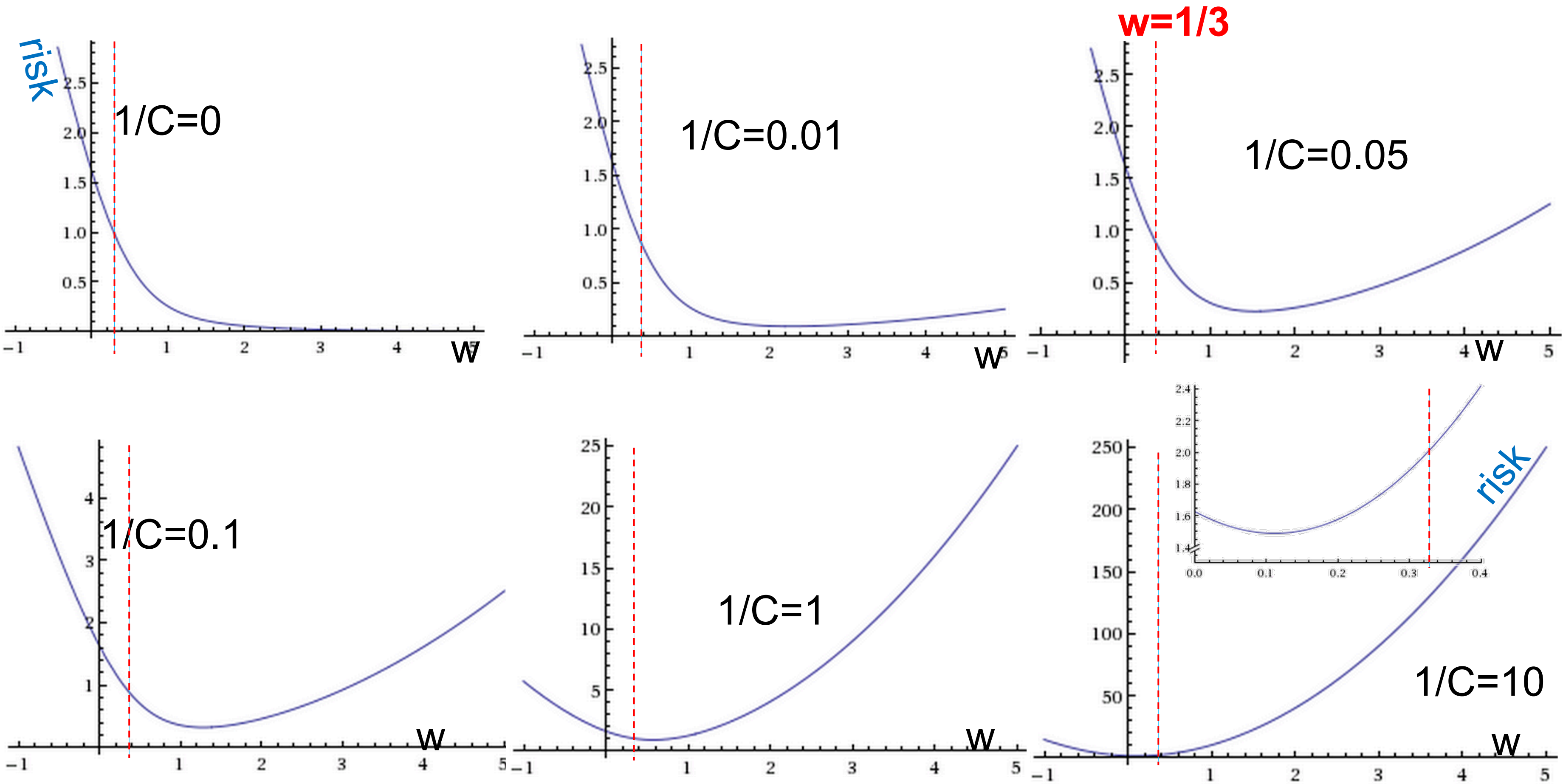
$$\arg\min_w \boxed{\frac{1}{C}} \|w\|_p^p + \sum_{i=1}^{m} \ln(1 + e^{-y_i w^T x_i})$$

- Logistic regression with L_2 regularization
- Higher 1/C => higher penalty for large norm of w
    - Example: two points (x=1,y=1), (x=-1,y=-1), w_0=0 (fixed value)



L_2 Regularization $\arg\min_w \frac{1}{C}\|w\|_p^p + \sum_{i=1}^m \ln(1+e^{-y_i w^T x_i})$

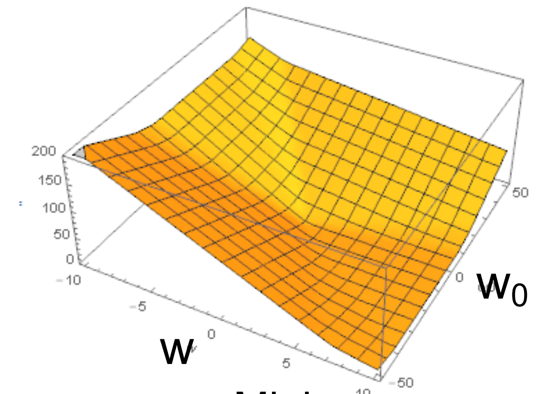
- Example: two points (x=3,y=1), (x=-1,y=-1), w_0=-1 (fixed value)
  - $wx+w_0$=3*w + (-1) > 0 => **w > 1/3 (condition for correct prediction)**
  - $wx+w_0$=-1*w + (-1) < 0 => w > -1 (condition for correct prediction)
- Too much regularization is bad: 1/C=10 => w=0.11 => wrong prediction



L_2 Regularization

- Logistic regression:

$$\arg\min_w \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})$$



Minimum:
w -> 56.3318,
w_0 -> -48.3115

- Solved through gradient descent:

$$w_{t+1} = w_t - \frac{\partial \left(\sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i}) \right)}{\partial w}$$

$$w_{t+1} = w_t - \sum_{i=1}^{m} \frac{\partial \ln(1 + \mathrm{e}^{-y_i w^T x_i})}{\partial w}$$

- Weights may grow and grow, gradient never going to 0

L_2 Regularization

- Logistic regression with L_2 regularization:

$$\arg\min_w \frac{1}{C}\|w\|_2^2 + \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})$$

- Solved through gradient descent:

$$w_{t+1} = w_t - \frac{\partial\left(\frac{1}{C}\|w\|_2^2 + \sum_{i=1}^{m} \ln(1 + \mathrm{e}^{-y_i w^T x_i})\right)}{\partial w}$$

$$w_{t+1} = w_t - \frac{2}{C} w_t - \sum_{i=1}^{m} \frac{\partial \ln(1 + \mathrm{e}^{-y_i w^T x_i})}{\partial w}$$

- **Weight decay!**
  - In each iteration, old weights are reduced by a fraction
    - Before we add something new, from gradient
  - Weights don't grow to be large

Coding classification methods

Thankfully, we do not have to act like in HW1

In the last couple of years, a number of libraries for automating gradient descent became popular

Tensorflow, PyTorch

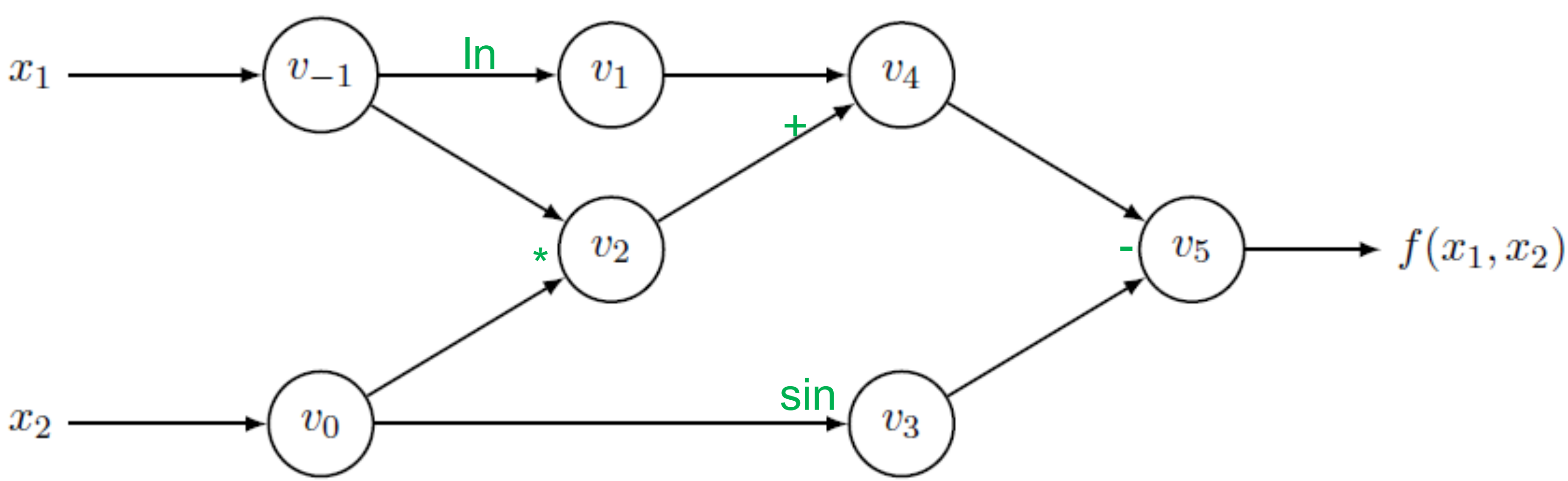
Automatic differentiation

Tensorflow/PyTorch are libraries for performing calculations and derivatives on a *computational graph*

Example: $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$

We want to compute y and dy/dx_1 for:

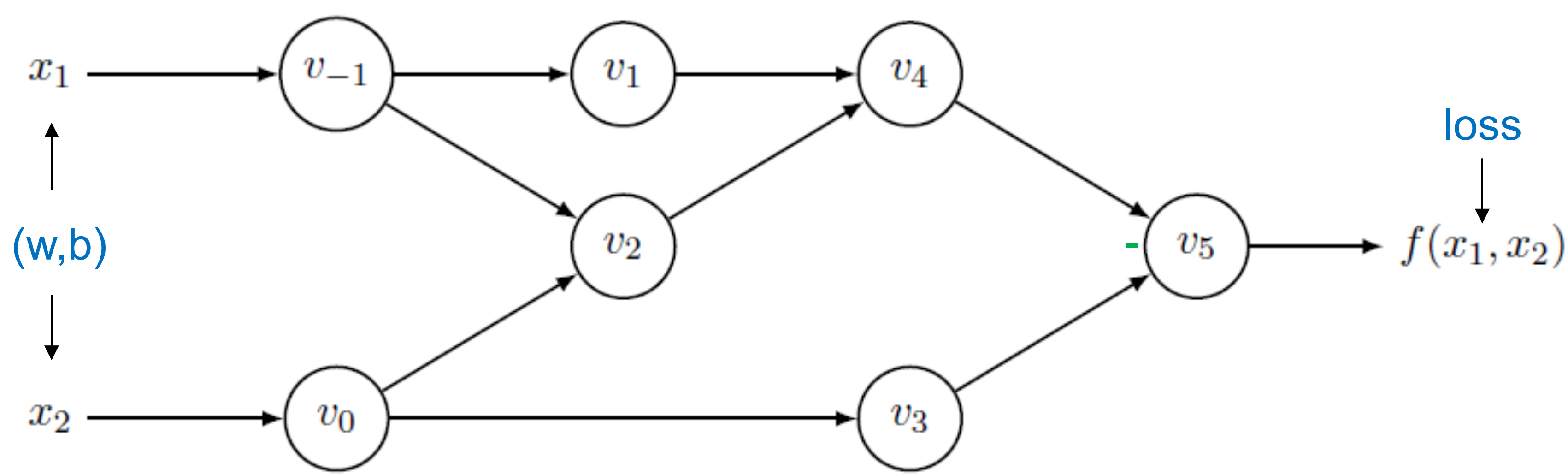
$$(x_1, x_2) = (2, 5)$$



Automatic differentiation

In machine learning, we rarely want to calculate gradient of loss w.r.t. feature values x_i

Instead, we will have model weights (w,b) as the starting variables for the computational graph (training samples x,y would just be constants somewhere in the graph)

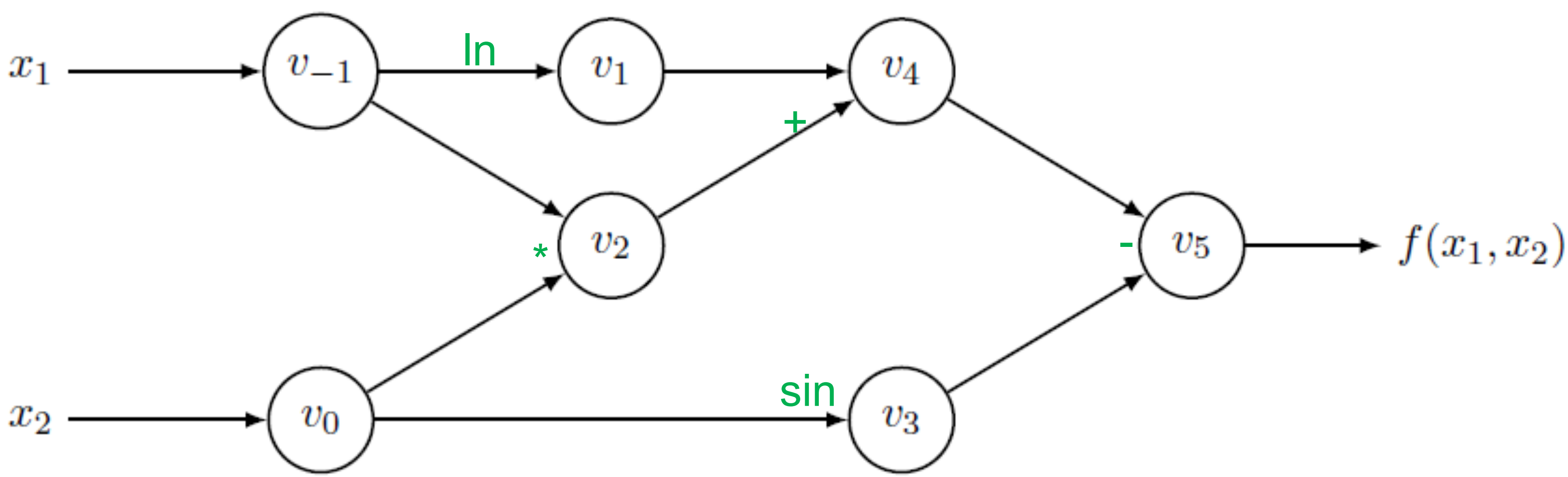


Automatic differentiation (AD)

Forward Primal Trace

$$
\begin{array}{lll}
v_{-1} & = x_1 & = 2 \\
v_0 & = x_2 & = 5 \\
\hline
v_1 & = \ln v_{-1} & = \ln 2 \\
v_2 & = v_{-1} \times v_0 & = 2 \times 5 \\
v_3 & = \sin v_0 & = \sin 5 \\
v_4 & = v_1 + v_2 & = 0.693 + 10 \\
v_5 & = v_4 - v_3 & = 10.693 + 0.959 \\
\hline
y & = v_5 & = 11.652
\end{array}
$$

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \quad (x_1, x_2) = (2, 5)$$



AD

Dot over a variable represents a derivative
Over what? Below, over x_1
There will be a similar trace for x_2

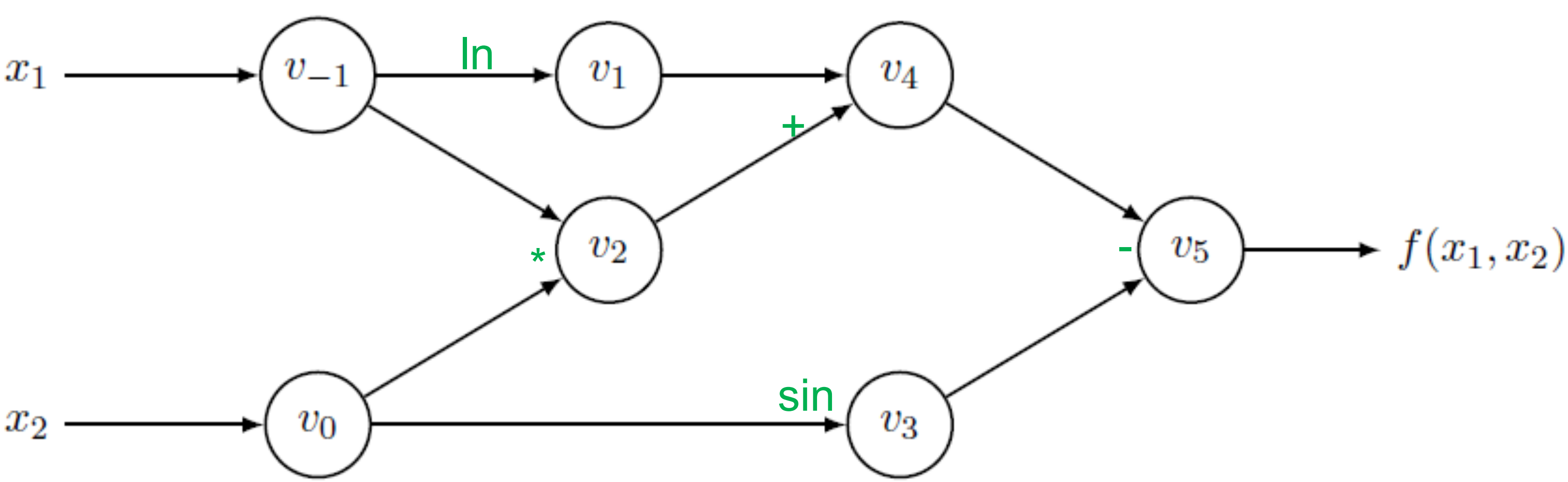
Forward Primal Trace

$$
\begin{array}{lll}
v_{-1} & = x_1 & = 2 \\
v_0 & = x_2 & = 5 \\
\hline
v_1 & = \ln v_{-1} & = \ln 2 \\
v_2 & = v_{-1} \times v_0 & = 2 \times 5 \\
v_3 & = \sin v_0 & = \sin 5 \\
v_4 & = v_1 + v_2 & = 0.693 + 10 \\
v_5 & = v_4 - v_3 & = 10.693 + 0.959 \\
\hline
y & = v_5 & = 11.652
\end{array}
$$

Forward Tangent (Derivative) Trace

$$
\begin{array}{lll}
\dot{v}_{-1} & = \dot{x}_1 & = 1 \\
\dot{v}_0 & = \dot{x}_2 & = 0 \\
\hline
\dot{v}_1 & = \dot{v}_{-1}/v_{-1} & = 1/2 \\
\dot{v}_2 & = \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1} & = 1 \times 5 + 0 \times 2 \\
\dot{v}_3 & = \dot{v}_0 \times \cos v_0 & = 0 \times \cos 5 \\
\dot{v}_4 & = \dot{v}_1 + \dot{v}_2 & = 0.5 + 5 \\
\dot{v}_5 & = \dot{v}_4 - \dot{v}_3 & = 5.5 - 0 \\
\hline
\dot{y} & = \dot{v}_5 & = \mathbf{5.5}
\end{array}
$$

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \quad (x_1, x_2) = (2, 5)$$



AD

Dot over a variable represents a derivative
Over what? Below, over x_1
There will be a similar trace for x_2

Forward Primal Trace

v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

Forward Tangent (Derivative) Trace

$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
$\dot{v}_0$	$= \dot{x}_2$	$= 0$
$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
$\dot{y}$	$= \dot{v}_5$	$= \mathbf{5.5}$

$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2) \quad (x_1, x_2) = (2, 5)$$

We do not get derivatives in a form of mathematical formulas

We just get the value of the derivative for specific x_1, x_2

That's what we need for gradient descent!

AD

https://arxiv.org/pdf/1502.05767.pdf

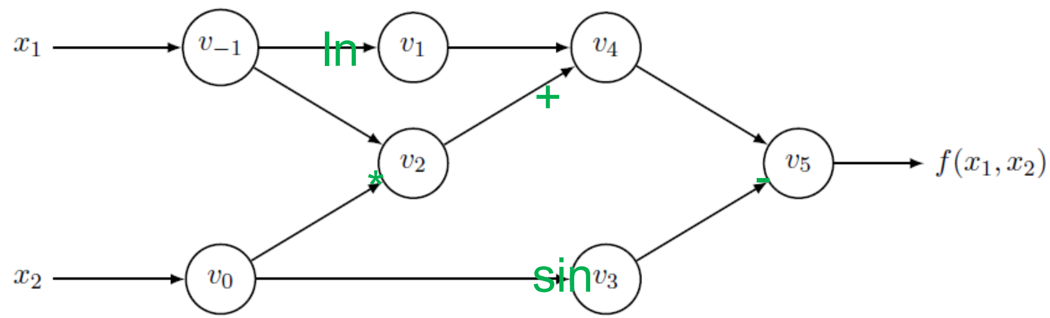
Forward Primal Trace		
v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

Forward Tangent (Derivative) Trace		
$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
$\dot{v}_0$	$= \dot{x}_2$	$= 0$
$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
$\dot{\boldsymbol{y}}$	$= \dot{\boldsymbol{v}}_5$	$= \mathbf{5.5}$

Above, we had forward differentiation

Calculate d v_i / d x for increasing i

Until we get to d v_5 / d x = d y / d x

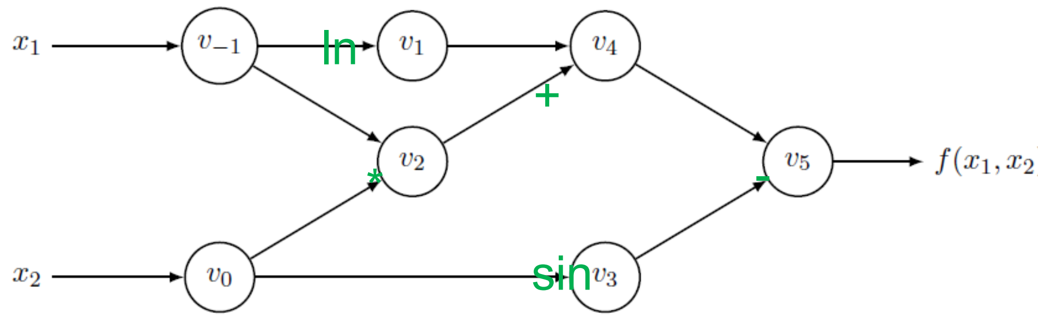


AD

There is an alternative way, closer to backpropagation

Calculate d y / d v_i for decreasing i

Until we get to d y / d v_0 = d y / d x



Forward Primal Trace

$v_{-1} = x_1$		$= 2$
$v_0 = x_2$		$= 5$
$v_1 = \ln v_{-1}$		$= \ln 2$
$v_2 = v_{-1} \times v_0$		$= 2 \times 5$
$v_3 = \sin v_0$		$= \sin 5$
$v_4 = v_1 + v_2$		$= 0.693 + 10$
$v_5 = v_4 - v_3$		$= 10.693 + 0.959$
$y = v_5$		$= 11.652$

Reverse Adjoint (Derivative) Trace alternative way (faster)

$\bar{x}_1 = \bar{v}_{-1}$		$= 5.5$
$\bar{x}_2 = \bar{v}_0$		$= 1.716$
$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0 = \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1} = \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5 = \bar{y}$	$= 1$	