Guión das prácticas no IGFAE

Prácticas optativas de Grao · Curso 24-25
Universidade de Santiago de Compostela

Índice

1. Presentación

1.1. Obxectivos

O obxectivo destas prácticas vai ser de introducir técnicas de *machine learning* na análise de datos do detector *ACtive TARget and Time Projection Chamber* (ACTAR TPC). Máis concretamente, centrarémonos en identificar as partiículas de alta enerxía que superan dous muros de silicio. Trátase dunha técnica novel que nunca se implementou neste tipo de dispositivos, polo que non está garantido que funcione... Pero merece a pena probar e pode ofrecer unha formación moi interesante.

Abarcaremos varias áreas:

- **Programación**: Afondaremos na programación con `python`, utilizando módulos de *ML* como `tensorflow` ou `scikit-learn`. Melloraremos os coñecementos de `pandas` e `matplotlib`, así como unha introdución aos **histogramas** con `hist`.
- **Detectores**: ACTAR TPC é un detector gasoso moi recente que abre todo unha nova ventá de posibilidades no eido das reaccións nucleares. Introduciremos os conceptos básicos dun detector gasoso e complementarémolos cos de detectores de silicio, esenciais neste experimento.
- **Teoría**: Expoñeremos as leis básicas da cinemática que rixen calquera reacción entre partículas e introduciremos os fundamentos da interacción radiación-materia, básicos neste experimento.

Comezaremos estudando os fundamentos teóricos e implmentándoos manualmente para asegurar que se entenden ben. A segunda parte abarcará a parte computacional en si mesma, tentando chegar ao obxectivo final de identificar as partículas.

1.2. Motivación

Este ano levarase a cabo un experimento para estudar reaccións de transferencia co ^{11}Li, de xeito que se poda facer un estudo pormenorizado da súa estrutura nuclear. Estudos coma este son moi importantes, posto que o *modelo de capas nuclear* (moi parecido ao seu análogo atómico) sábese que falla a medida que nos afastamos da estabilidade e pasamos a outros núcleos máis exóticos ($N/Z \gg 1$), e faise necesario postular modelos teóricos máis amplos, válidos de forma xenérica.

As reaccións que imos estudar son:

- $^{11}\mathrm{Li} + \mathrm{d} \rightarrow \mathrm{p} + {}^{12}\mathrm{Li}$

  É unha reacción de engádega dun neutrón.

- $^{11}\mathrm{Li} + \mathrm{d} \rightarrow \mathrm{d} + {}^{11}\mathrm{Li}$

  É a difusión elástica ($\mathrm{E}_x = 0$) ou inelástica ($\mathrm{E}_x > 0$).

- $^{11}\mathrm{Li} + \mathrm{d} \rightarrow \mathrm{t} + {}^{10}\mathrm{Li}$

  Aquí, ao contrario, eliminamos un neutrón.

Todo isto é posíbel grazas á seguinte configuración do detector:

- **Gas**: O gas que encherá ACTAR TPC será unha mestura de D_2 e CF_4 (90 % e 10 %, respectivamente) a unha presión de 900 mbar. Será o D_2 o que provea os d da reacción.
- **Feixe**: O ^{11}Li terá unha enerxía de 7,5 AMeV (é dicir, 82,5 MeV), e será producido no acelerador ISAC de TRIUMF, en Canadá.

2. Principios físicos

2.1. Un detector gasoso

ACTAR TPC segue os principios de funcionamento dunha *time projection chamber*: as partículas ao pasaren polo gas ionizan os átomos, liberando electróns que **derivan** baixo aplicación dun campo eléctrico. Esta carga é recollida nun sensor amplamente fragmentado (*pad plane*), permitindo *seguir* as partículas no seu traspaso dentro do medio en 2D.

Pero é máis: pódese engadir a terceira coordenada coñecendo o tempo que lle leva aos electróns chegar ao *pad plane*, pois a **velocidade de deriva** é constante. Deste xeito, o seguimento das partículas faise en **3D**.

Finalmente, o principio de *active target* engade a vantaxe de ser o gas o propio albo da reacción: este actúa como medio de reacción e de detección. O seguinte esquema ilustra este modo de funcionamento.

Os datos que vas analizar proveñen dunha simulación na cal se implementaron os seguintes parámetros:

- O tamaño da zona de detección (Figura 2) é de $256 \times 256 \times 255\,\mathrm{mm}^3$.
- O criterio de dimensións é que o beam vai no eixo X, e a dimensión vertical é Z.
- Os detectores auxiliares poden poñerse arredor desta zona nos diferentes lados. Nós usaremos algo parecido ao da figura, cos detectores de Si (como os azuis) cara adiante.
- Teremos dous muros de silicio, un tras do outro. O primeiro está a 10 cm do *pad plane* e o segundo está separado do primeiro 3 cm. Cada muro constitúese de 12 unidades de tamaño $80 \times 50 \times 1{,}5\,\mathrm{mm}^3$.

A disposición xeométrica é a que se amosa a continuación.

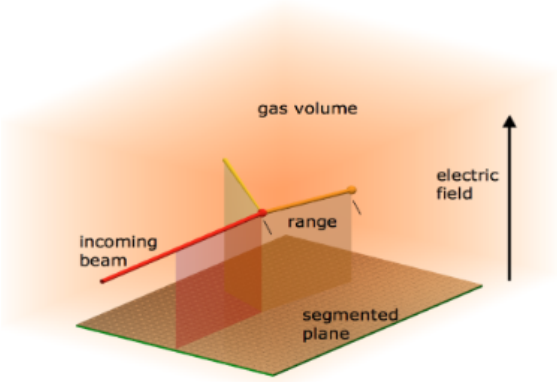

Figura 1: Esquema de funcionamento dunha TPC.

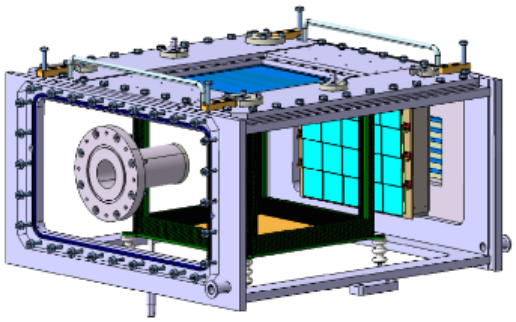

Figura 2: Deseño do detector ACTAR TPC. A rexión de detección é a parte negra interna.

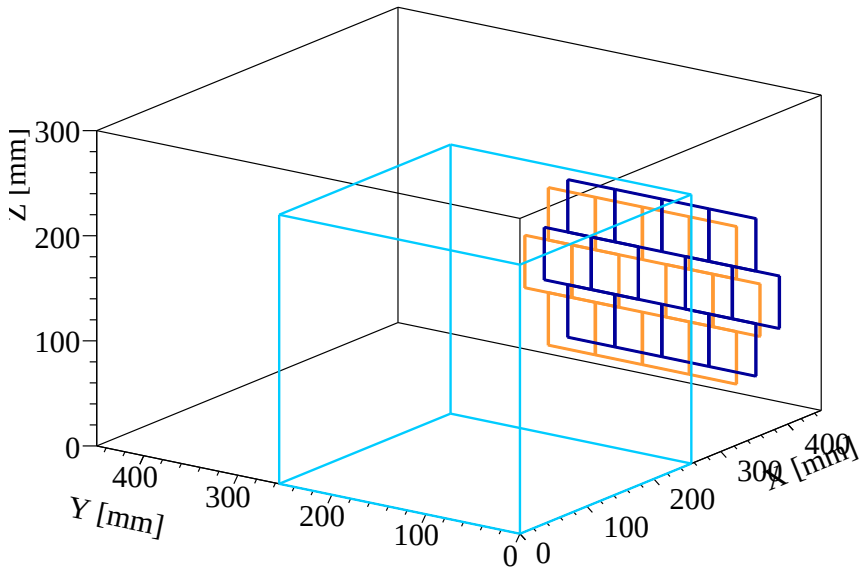

Figura 3: O código de cores é o seguinte: ACTAR TPC en azul, en escuro a primeira capa de silicios e en laranxa o segundo.

2.2. Perdas de enerxía na materia

As partículas ao propagarse a través do gas interaccionarán co campo de Coulomb dos átomos deste, perdendo enerxía e liberando electróns. Esta perda de enerxía en cada interacción é moi pequena, pero como é moi elevado o número de colisións no gas (é, polo tanto, un proceso estocástico), chega a ser moi importante. Nunha aproximación continua, está dada pola **ecuación de Bethe-Bloch**:

$$-\frac{dE}{dx} \cong \frac{4\pi e^4}{m_e}\left(\rho\frac{N_A}{M}\right)\frac{z^2 Z}{v^2}\ln\frac{2m_e v^2}{I}$$

Depende esencialmente da partícula a considerar (Z e A) e do gas (a través de ρ). É interesante definir:

- Rango (R): É o camiño percorrido por unha partícula ata que *case* se detén. É función da enerxía inicial e do material a atravesar. Existe unha **relación unívoca** $E \iff R$ que imos utilizar para estimar a perda da enerxía en función da distancia percorrida pola partícula.
- *Straggling*: Como a perda de enerxía é un **proceso estatístico**, dúas partículas coa mesma enerxía inicial non van depositar a mesma ΔE. O *straggling* mide, dalgunha forma, a σ desta distribución.

Esta información está tabulada nun programa que se coñece como ***SRIM*** e que foi empregado para obter os datos que se analizarán. Non obstante, vas familiarizarte con el para entender ben os conceptos da interación radiación-materia introducidos.

2.3. Detectores de silicio

Vas estudar os seus principios de funcionamento o próximo curso en *Electrónica Física*, pero aquí tes unhas nocións básicas sobre o seu funcionamento. Cada silicio funciona como unha **unión PN**, na cal algúns átomos da rede cristalina de Si se subsituíron por outros: na rexión *P*, por átomos deficitiarios en electróns (hai máis *ocos*), mentres que na *N* por outros ricos en e^-. Ao xuntalos créase unha zona interna de campo eléctrico pola diferenza de cargas, o que se coñece como *zona de baleirado* ou de *carga espacial*, e que ten un tamaño típico de µm.

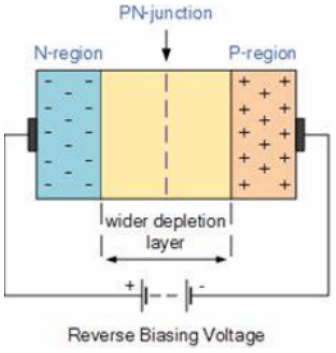

Figura 4: Esquema dunha unión PN polarizada en inversa

Baixo a aplicación dun potencial apropiado, nunha configuración que se coñece como *polarización inversa*, esta zona de baleirado faise moi grande, ocupando case todo o detector. E isto é o que nos interesa: as partículas cargadas, ao atravesar esta rexión, ionizan o medio liberando pares electrón-oco. Grazas ao campo eléctrico interno da unión PN podemos recollelos (os e^-) e medilos como correcte eléctrica. O sinal recollido é proporcional ao número de electróns liberados na ionización e, polo tanto, á enerxía incidente da partícula.

2.4. Incertezas experimentais

Como xa comentamos, os datos que se analizarán non proveñen dun experimento real senón dunha simulación. Non obstante, esta ten en conta os defectos experimentais implementables nun computado e que son determinantes en moitos casos do éxito ou non dun experimento. Aquí unha lista:

- O *straggling* en enerxía da partícula ao propagarse no gas.
- O mesmo pero ao propagarse no silicio.
- A **resolución** dos detectores de silicio. Na Sección anterior comentamos que a corrente medida é proporcional ao número de electróns de ionización xerados no proceso. Non obstante, a estatística gaussiana asígnalle unha interteza $u = \sqrt{N_e}$, o cal quere dicir que unha partícula coas mesmas características que atravese o Si non vai necesariamente rexistrar a mesma ΔE. Experimentalmente mídese e estímase en 50 keV de incerteza a unha enerxía incidente de 5,5 MeV.

2.5. Cinemática

Todo proceso de interacción entre partículas podes escribirse en termos de variables cinemáticas (enerxías e ángulos) usando as **leis de conservación** de enerxía e de momento. Esta pode ser descrita no sistema LAB ou no CM (mediante unha transformación Lorentz a un sistema con 3-momento nulo en ambas canles de entrada e saída) do seguinte xeito:

$$\text{No LAB: } p_1 = (E_1, \boldsymbol{p_1})\,;\ p_2 = (m_2, \boldsymbol{0})\,;\ p_3 = (E_3, \boldsymbol{p_3})\,;\ p_4 = (E_4, \boldsymbol{p_4})$$
$$\text{No CM: } p_1' = (E_1', \boldsymbol{p_1'})\,;\ p_2' = (E_2', -\boldsymbol{p_1'})\,;\ p_3' = (E_3', \boldsymbol{p_3'})\,;\ p_4' = (E_4', -\boldsymbol{p_3'})$$

É estándar asumir unha partícula *target* (a 2) en repouso. Aínda que a simulación xa está feito, penso que é interesante que comprendas como se xeran os eventos. Para iso, vas resolver as ecuacións de cinemática para obter a representación T vs θ da partícula 3 que mediremos nós. Para iso:

- Partiremos da enerxía cinética T_1 do feixe (1) no LAB.
- Calcularás a transformación Lorentz da canle de entrada ao CM.
- Repartiremos a E_{CM} (enerxía total no centro de masas) entre as partículas de saída en función de θ_{CM} e ϕ_{CM}.
- Finalmente, recuperaremos as enerxías de saída de ambas partículas no LAB coa transformación inversa.
- Tamén necesitaremos os ángulos no laboratorio, θ_{Lab} e $\phi_{Lab} = \phi_{CM}$.

Algunhas fórmulas que che poden ser interesantes para unha transformación Lorentz en X (o feixe móvese nesa dirección; é un convenio):

$$\begin{aligned} E' &= \gamma(E - \beta p_x), & p'_x &= \gamma(p_x - \beta E) \\ E &= \gamma(E' + \beta p'_x), & p_x &= \gamma(p'_x + \beta E') \end{aligned}$$

Onde as variables primadas indican que son no CM. Cando transformes ao final do CM ao LAB terás en conta θ_{CM}: $p'_{3,x} = |\boldsymbol{p'_3}| \cdot \cos(\theta_{CM})$. O ángulo ϕ_{CM} non aparece de momento posto que é transversal á transformación.

3. Para comezar

3.1. Instalación de paquetes

Para poder realizar esta análise imos necesitar unha serie de paquetes de *python*:

1. Crea un entorno virtual para esta análise, así non modificaremos nada da túa instalación actual. Segue as instrucións de aquí.
2. Unha vez dentro do teu *venv*, instala os paquetes básicos: `numpy`, `matplotlib`, `pandas`
3. Instala `uproot`. É un paquete que nos vai permitir ler os ficheiros de simulación creados en formato `.root`
4. Instala o paquete `PyPhysics` que che ofrecerá moitas clases de interese (cinemática e perdas de enerxía). Clona o repositorio e modifica o teu PYTHONPATH.

3.2. Github

Git é unha ferramenta de control de versións que che permitirá manter un historial do teu código, podendo volver atrás cando sexa necesario. Github é unha web para almacenar repositorios Git. Vamos utilizala para manter o código seguro nun servidor e volver cara atrás cando sexa necesario.

Para iso, o primeiro é ter unha conta en `www.github.com`. Tras unha configuración inicial un pouco tediosa, o plan de traballo é o seguinte, que se debe executar cando fagas cambios importantes ou cando remates a túa sesión de traballo:

1. `git add .` vai engadir todos os cambios dende o anterior *commit*
2. `git commit` abrirá un editor de texto no que crear unha mensaxe para informar dos cambios. Péchase con `Ctrl+S`, `Ctrl+X`
3. `git push` envía os cambios á nube

No tocante ao paquete `PyPhysics`, probablemente teremos que actualizalo ao longo das prácticas. Para recibir os cambios do servidor, vai á carpeta na que está descargado e executa `git pull`. Desta forma teralo actualizado.

4. Plan de traballo

Este é o plan de traballo das próximas semanas.

- **Primeira semana**: Instalación de paquetes e familiriziación co entorno. Representación de todas as variables da simulación e das súas correlacións. Estudo da cinemática e das perdas de enerxía.
- **Segunda semana**: Preparación dos datos para entrenar unha rede neuronal. Comprensión do funcionamento destas e implementación dunha simple.
- **Terceira semana**: Engédega da rede neuronal convolucional 1D á anterior para analizar os perfís de carga. Ver se isto mellora os resultados. *Fine-tuning* dos hiperparámetros.
- **Cuarta semana**: Análises complementarias e propostas doutras solucións en caso de que non funcione. Redacción da memoria de prácticas.

4.1. Primeira semana

Imos comezar por representar todas as variables que nos dá a simulación. Para iso empregaremos histogramas, que podes construír usando o módulo `hist`.

```
import hist
## histograma 1D
h1d = hist.Hist.new.Reg(nbinsx, xmin, xmax, label="X axis").Double()
## histograma 2D: hai que especificar ambos eixos
h2d = hist.Hist.new.Reg(nbinsx, xmin, xmax, label="X axis")
                   .Reg(nbinsy, ymin, ymax, label="Y axis").Double()
```

Xunta todos os ficheiros nun único dataframe e representa:

- A cinemática: T_3 vs θ_3. deberías ver 3 liñas ben diferenciadas
- As perdas de enerxía nos silicios: ΔE_0 vs ΔE_1. É o que se coñece como *Particle IDentification, (PID).*
- A coordenada X do punto de reacción `RPx`
- Colle un perfil de carga de cada reacción e compáraos

O seguinte paso vai ser entender o que vemos:

1. Fai os cálculos de cinemática para obter a liña téorica para as tres reaccións: ^{11}Li(d, p), (d, d), (d, t). Podes intentar facer unha **clase** en `python`.
2. Empregando a clase `EnergyLoss` de `PyPhysics` fai propagar p, d, t unha mesma distancia no gas para distintas enerxías. Estuda o que vés e intenta explicalo a partir da fórmula de Bethe-Bloch.
3. Explica porque hai moitas máis contas de `RPx` cara ao final da TPC.

4.2. Segunda semana

Agora imos aprender o que é unha rede neuronal e os conceptos básicos do *machine learning*.

1. Usa o Capítulo 2 de *Hands-on Machine Learning with Scikit-Learn and Tensorflow* para ter unha idea de como debes tratar os datos. Aquí van algunhas suxestións:
   - Crea un único dataframe con *labels* indicando se é p, d, t
   - Downsample os datos de forma que haxa o mesmo número de eventos de cada reacción
   - Aleatoriza o `df`
   - Decide que *features* debes eliminar: a T_3 claramente, pois se non permitiría unha distinción absoluta entre as partículas. Pensa que experimentalmente non é accesible sen identificar antes a partícula medida. Por outra banda, os perfís de carga deixarémolos para máis adiante.

- Normaliza as *features*
- Encodifica as variables de clasificación: de strings a enteiros
- Dívideo en *train* e *validation*

2. Le as *Lectures* da U. Cornell para entender o que é unha rede neuronal e en que consiste o proceso de entrenamento. A idea coa que debes reter sobre unha rede neuronal é que é un conxunto de funcións as cales, a partir duns parámetros de entrada (*features*), aplican unha serie de transformacións (*weights* + *activation function*) para devolver unha saída. No noso caso, esta é unha clasificación: probabilidade de que sexa p, d, t.
3. Cando metamos a rede convolucional estaría ben que leses os capítulos 6 e 7 das *Lectures* da U. Pennsylvania. A nosa rede convolucional será 1D (un histograma) en lugar de 2D (unha imaxe), pero os conceptos son os mesmos.

De seguido imos construír o noso primeiro modelo de rede neuronal. Na seguinte Figura tes un exemplo.

```
model = Sequential([
    Input(shape=(X_scaled.shape[1],)),
    Dense(64, activation="relu"),
    Dense(64, activation="relu"),
    Dense(32, activation="relu"),
    Dense(len(encoder.classes_), activation="softmax")
])

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
model.summary()
```

Figura 5: Modelo básico de rede neuronal para clasificación.

Algunhas apreciacións:

- Usamos activación *softmax* na layer de saída para clasificación non binaria (hai 3 clases de saída)
- A función a minimizar é a *cross entropy*, tamén adecuada para estas tarefas

Axusta o modelo e plotea a fit accuracy por epoch, así como a matrix de confusión. Podes representar outras métricas, como a precision ou a recall. Le o Capítulo 3 do libro para máis información.

Podes probar outras arquitecturas da rede, con outras funcións de activación ou optimizadores (Capítulo 11, *Optimizers* do libro) a ver se atopas unha combinación que ofreza unha mellor exactitude.

Por último, no plot do PID superimpón os eventos mal identificados e que ideas se che ocorren para explicalo.

4.3. Terceira semana

Finalmente, intentaremos mellorar a rede neuronal introducindo os perfís de carga, pois como xa viches son característicos de cada partícula, polo que poderían axudar moito na clasificación. Para iso deberemos combinar a rede neuronal simple anterior cunha convolucional 1D, pois non se trata dunha imaxe senón dunha serie de valores de dE/dx.

Na anterior Subsección tes os documentos base que che permitirán entender o seu funcionamento. O modelo que entrenamos vaise ter que modificar, pois agora imos *concatenar* os resultados de dúas redes: a DNN + a CNN1d. A estrutura é a seguinte:

- Un modelo secuencial exactamente igual ao anterior para a parte non convolucional.
- Un modelo secuencial para a parte convolucional. Podes probar co da Figura 6 para comezar.

  No código, `filters` é o número de patróns que vai intentar aprender a CNN; `kernelsize` o número de posicións do vector que se teñen en conta en cada filtro e `strides` o número de posicións que se

```python
# Plain
plain_in = Input(shape=(X_scaled.shape[1],), name="plain_input")
plain_mod = Sequential([
    Dense(64, activation="relu"),
    Dense(64, activation="relu"),
    Dense(32, activation="relu")
], name="plain_mod")
plain_out = plain_mod(plain_in)

# Vector (Conv1D) branch
conv_in = Input(shape=(X_prof.shape[1], X_prof.shape[2]), name="conv_input")
conv_mod = Sequential([
    Conv1D(filters=32, kernel_size=3, strides=1, activation="relu", padding="same"),
    Conv1D(filters=16, kernel_size=3, strides=3, activation="relu", padding="same"),
    Flatten(),
    Dense(16, activation="relu")
], name="conv_mod")
conv_out = conv_mod(conv_in)

# Concatenate outputs
combined = Concatenate()([plain_out, conv_out])
x = Dense(32, activation="relu")(combined)
output = Dense(len(encoder.classes_), activation="softmax")(x)

model = Model(inputs=[plain_mod.input, conv_mod.input], outputs=output)
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
model.summary()
```

Figura 6: Exemplo de rede CNN1d.

despraza o *kernel* entre iteracións. Podes xogar metendo máis layers CNN, cambiando o número de filtros, o tamaño do *kernel*, etc. Tamén podes probar engadindo unha capa de *pooling*. A layer `Flatten` é necesaria para adecuar as dimensións para a seguinte `Dense`, que necesitamos para ter a mesma saída que o modelo DNN.

- Concatenamos ambos modelos e superpoñemos máis capas densas para combinar as saídas de ambos modelos. A output é a mesma que anteriormente, coa activación *softmax*. Deberías apreciar unha mellora do 10 % respecto ao modelo anterior.

4.4. Cuarta semana

O último paso é amosar unha alternativa ás redes neuronais presentadas. Esta consiste en reconstruír a enerxía cinética de cada partícula no vértice asumindo os tres posibles casos (*p, d, t*) para cada evento e ver cal que hipótese é máis válida dacordo coa cinemática.

Para iso:

1. Colle só os eventos que fan punchthrough na segunda capa de silicios (os que queremos identificar).
2. Sabendo o espesor dos silicios de 1500 mm, crea unha interpolación para cada partícula: coa función `EnergyLoss.Slow()` garda a ΔE_1 para cada E_{ini} na entrada do silicio. Podes empregar unha `scipy.CubicSpline` en lugar dunha interpolacion lineal.
3. Selecciona uns poucos eventos protón. Reconstrúe a súa enerxía no vértice de reacción. Para iso usa a *spline* obtina antes para unha partícula:
   - Avalíaa na ΔE_1, obtendo a enerxía á entrada do Si 1.
   - Utiliza `EnergyLoss` no **gas** para obter a enerxía de saída da partícula no silicio 0 tras percorrer a distancia `tl_01`. A función `EnergyLoss.EvalInitialEnergy` fai xusto o contrario de `Slow`.
   - A enerxía á entrada do silicio 0 será a suma $\Delta E_0 + E_{\text{AfterSil0}}$ que acabas de recuperar.
   - Por último, calcula a enerxía no vértice (que é onde representas ti a cinemática). Aplica `EvalInitialEnergy` á enerxía anterior pero na distancia entre o silico e o vértice `tl_gas0`.
4. Con isto obterás un punto (T, θ) que podes representar sobre un gráfico de cinemática.

5. Fai o mesmo asumindo que o evento é d e t. Represéntaos sobre o mesmo gráfico de cinemática.

6. Deberías ver como, para un evento p, asumir d ou t leva a un punto que se separa moito de calquera liña cinemática. Só o valor reconstruído como p ten sentido.

7. Deste xeito tes unha identificación. O mesmo aplicará para eventos de deuterio e tritio.

8. Hai unha aproximación aquí: a interpolación que calculas non depende do ángulo, pero en realidade si que o debería facer, pois a perda de enerxía no silico dependerá do ángulo co que a partícula o atravese (a maior ángulo maior é a distancia que percorre, $d = 1500\,\mu\text{m}/\cos\theta$). Non é necesario que fagas o cálculo tendo en contas isto pero debes sabelo.

9. Tamén hai unha simplificación: $E_x = 0\,\text{MeV}$ en todo momento. Nun experimento real, o núcleo final ^{10}Li quedará nun estado con enerxía de excitación E_x que fará que a cinemática da partícula lixeira non sexa unha liña senón unha *distribución* no plano. Isto pode complicar a sinxeleza deste método.

5. Referencias

1. *Hands-on Machine Learning with Scikit-Learn & Tensorflow*, A. Géron (2017)

   É o libro de referencia. A maioría de conceptos están aí definidos, tanto para redes neuronais densas como convolucionais.

2. *Introduction to Neural Networks*, Y. Zhang *et al.* University of Cornell (2021). Unhas diapositivas interasentes para ter un resumo das NN se non queres ler o libro.

3. *Neural Networks*, D. Roth *et al.*. University of Pennsylvania (2016). Os capítulos 6 e 7 explican algo máis detalladamente as CNN.